Text Entry in Virtual Reality; Implementation of FLIK method and Text Entry Test-Bed

By

Eduardo Soto

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Information Technology: Digital Media

In

Department of Information Technology

Carleton University
Ottawa, Ontario

Abstract

We present new testing software for text entry techniques. Text entry is the act of entering text via some interaction technique into a machine or monitor, common text entry techniques are the typical QWERTY keyboard, smartphone touch virtual keyboards, speech to text techniques, and others. The purpose of this text entry test-bed software is to provide a flexible and reusable experiment tool for text entry studies, in such a way to include studies from a variety of sources, more specifically to this thesis, from Virtual Reality text entry experiments. On top of the proposed text entry test-bed, we conduct two studies comparing different text entry techniques in virtual reality as a way of validating the text entry test-bed, as well as contributing the results of these studies to the pool of research related to text entry in virtual reality.

Acknowledgements

I would like to thank my supervisor, Professor Robert Teather, for his assistance during this research. His advices and suggestions have consistently helped to steer this work in a productive direction and have greatly contributed to the quality of this thesis.

I would also like to thank Luis Soto and Elena Martin for their unconditional support throughout my academic journey. This could not have been possible without your encouragement.

Table of Contents

List of Tables

List of Figures

List of Appendices

This page lists all of the appendices.

Abbreviations

VR - Virtual Reality
AR – Augmented Reality
MSD – Minimum String Distance
KSPC – Keystrokes per Character
WPM – Words per Minute
FLIK – Fluid Intersection Keyboard
NASA-TLX – NASA Task Load Index
HMDs - Head Mounted Displays
API - Application Program Interface
DoF – Degrees of Freedom

Chapter 1: Introduction

According to Ko and Wobbrock, text is a ubiquitous form of verbal communication [13]. Text entry refers to the process of creating messages composed of characters, numbers, and symbols using an interface between a user and a machine. Text entry has been extensively studied in the field of human-computer interaction (HCI) over the past few decades, especially for use in desktop and mobile contexts. Efficient text entry is important because it directly impacts the user's ability to write documents, send messages, and communicate effectively when verbal communication is not available.

Virtual reality (VR) is defined as the use of computer technology to create simulated environments. It often employs head-mounted displays (HMD) along with spatially tracked controllers or hand trackers as input devices. VR has become increasingly popular in recent years; increased consumer demand means the interaction effectiveness of VR systems affects more people than ever. This thesis focuses on text entry in VR, such as entering text in search bars or sending text messages. In such use cases, it is important to include an effective option to enter text in VR systems as well as an effective method to evaluate novel text entry techniques and compare with existing techniques.

Sutherland's *Sword of Damocles*, is widely considered as the first virtual reality head-mounted display [4] despite it being an augmented reality (AR) display. Since Sutherland, extensive research has been dedicated to 3D interaction. This includes the definition of the fundamental canonical 3D interaction tasks, including selection, manipulation, travel, and system control [14]. These 3D interactions take advantage of

the degrees of freedom (DoF) provided by VR, meaning the number of basic ways a rigid object can move through 3D space. In particular, 3D systems (including VR) offer 6 DoF interaction, three degrees of freedom corresponding to rotation (pitch, yaw, and roll), and the other three correspond to the translational movement along the x, y, and z axes. The focus of this thesis is on text entry in VR, in particular, through the use of selection and system control methods, both of which can be applied to text entry in VR. Selection tasks mainly focus on the acquisition of targets, e.g., by pointing a ray at an object and pressing a button to indicate selection. This can be applied in text entry through the selection of keys on virtual keyboards. System control can include the use of physical controllers, graphical menus, voice commands, gestural commands, and others to operate menus, all of which can be employed for text entry in VR. While selection and system control do not strictly apply only to text entry concepts, they can be applied within this context to find new and effective text entry techniques in VR.

Research in VR text entry dates as far back as the 1990s, such as the Virtual Notepad exploring handwriting in immersive VR [21], as well as Maggioni's work on a novel gestural input device for VR [19], and the neural-interface Glove-Talk 2 [6]. It wasn't until the recent adoption of consumer-level VR devices that the need for an efficient and standard method for entering text in VR became a priority. With the rise in popularity, competition, and consumer acceptance of virtual reality head-mounted displays (HMDs), it is increasingly important to devise standards for interaction in VR. While designers can draw inspiration from existing practice such as mobile or desktop interaction and text entry techniques, adapting such methods to VR is not necessarily straightforward. Fundamental interaction in virtual reality is drastically different from

computer systems that came before, in that it supports much more direct interaction rather than using techniques or devices which yield layers of separation between the user and their task. Consider, for example, the difference between selecting an icon on a desktop using a mouse to indirectly control a cursor, versus directly picking up an object representing an icon in a VR system using one's hands. With VR applications in gaming, entertainment, simulation, therapy, training, and more [10, 20, 23], users need to be able to convey information in written form as efficiently and quickly as possible. While a user is immersed in a virtual environment, it is desirable to reduce the amount of distraction or breaks in the momentum of a particular experience by having to enter some text with an inefficient tool.

1.1 Motivation

To type long-form text in a VR environment is not yet feasible or likely even desirable. However, for quick messages or notes, this thesis makes contributions to text entry in VR. For example, consider playing a multiplayer online VR game, where communicating in written form may be preferable to speech communication. In such a scenario, players need to enter text in a fast and accurate way that does not get in the way of game play. Another scenario could be a VR conference where you would like to take notes and potentially send back-channel messages to other attendees. For gaming scenarios, one might find the need to type a quick SMS to a friend inviting them to come online, without interrupting the gameplay. In scientific fields or even in architecture, it might be useful to be immersed in a 3D environment and be able to annotate different components of the environment. There are cases where typing in VR might be preferable to voice recognition, such as being immersed in VR whilst in a loud environment, or

when you need to maintain quiet. Speech-to-text could work well for specific purposes, however, VR use has increased in areas such as office work, collaboration, and training and education. For these applications, inputting text is an essential part of these experiences and more often than not, precise text entry is required rather than using a speech-to-text entry technique. Modification of existing text is unreliable and inaccurate as compared to more refined and direct text entry techniques.

Accessibility is also a factor behind the purpose of having an efficient text entry technique for VR. As mentioned, it is not always desirable or even possible to use a speech-to-text solution. People with speech impediments or similar disabilities might rely solely on written or typed form of communication. On top of this, to make a system as accessible as possible, it is necessary to consider as many forms of interaction as needed and search for a feasible and standard method of employing these interactions. Text entry is one of the interactions that benefits from having a standard method of interaction, as can be seen from desktop and laptops using the standard keyboard as their text entry technique. It is necessary to go through many iterations of techniques of any interaction method, in our case for text entry, in order to optimize, refine, and ultimately find the standard method of interaction.

On the topic of standard techniques, the current standards for text entry in VR are sub-optimal, and have not seen much refinement or modifications since their first iterations. It is our goal to provide contributions that aid in the discovery of a standard text entry technique, as well as provide empirical study data that adds to current research in the field of text entry in VR.

1.2 Contribution

The first contribution of this thesis is a text entry testbed. A common issue for text entry research is consistency and better standardization of methodological practice in evaluating text entry systems [2]. Methodological consistency is important in text entry research since usually, analysis is comparative, i.e., involves empirically comparing text entry techniques to determine which is most efficient. Employment of similar methods and metrics to ensure comparability between studies is motivating factor of the text entry testbed. The text input testbed is a tool for conducting text entry studies which supports desktop, mobile, VR, and other text input methods, that adheres to standard practice in HCI text input research. The testbed provides a consistent platform to perform text entry comparisons regardless of the techniques used. The testbed was developed using the popular game engine software Unity 3D. With minimal Unity 3D experience required, and future work focusing on eliminating the need for any Unity 3D experience, researchers can implement a text entry technique to add key logging functions that serve as individual character inputs. The testbed is described in Chapter 3.

The second contribution is a novel VR text entry technique called Fluid Interaction Keyboard (FLIK). FLIK uses two controllers with 6DoF (e.g., Oculus Touch controllers) together with a simple interface for fast and intuitive text entry. It operates by selecting characters from a virtual soft keyboard by directly touching the characters. When eliminating physical button presses on the controllers, users are enabled to type faster using this technique and achieve fluid hand motions to increase the speed of character selection. FLIK was inspired by another text entry technique, Cutie Keys [3, 31] and was designed to improve on existing practice in VR text entry.

The third and final contribution is a formal evaluation of FLIK, comparing it to existing VR text entry techniques through two user studies. The first study also validates the use of the evaluation testbed. It compares four text entry techniques in a typical text entry experiment where users transcribe phrases presented to them using the different techniques. Two of the techniques are industry standard text entry techniques for virtual reality: Controller Pointing, and continuous cursor selection. The other two techniques are FLIK, and physical keyboard typing on a tracked keyboard using Logitech's Bridge SDK [28], which was added for its high performance.

The second study employs the testbed, and compares Controller Pointing and FLIK across two aids intended to further facilitate fast text entry: BigKey [31] and Word Disambiguation. BigKey resizes keys to offer faster selection depending on the probability that the next letter will be typed, where the selection is based on common digraph and trigraph frequencies. Word Disambiguation behaves similar to common mobile phone text entry and makes word suggestions based on what was already typed. The results of these studies are provided along with a discussion of performance, errors, and user questionnaires.

1.3 Outline of Thesis

The thesis is divided into six chapters:

Chapter 1 introduces the problem with text entry for virtual reality and the importance of having fast, accurate, and efficient text entry in virtual reality.

Chapter 2 provides an overview of related work in text entry research. Covering previous work on text entry error and performance analysis, text entry techniques, and text entry techniques in the realm of virtual reality. It provides insight and the "big picture" of the state of text entry to bring the reader up-to-date with the literature to follow along with the rest of the thesis.

Chapter 3 describes the design and implementation of the text entry test-bed, the FLIK text entry technique, BigKey for VR, Word Disambiguation, Logitech Bridge API, and the other text entry techniques used.

Chapter 4 is dedicated to the first user study, where Controller Pointing, continuous cursor selection, physical keyboard, and FLIK are tested and compared.

Chapter 5 is dedicated to the second user study, where only Controller Pointing and FLIK are compared, except that now they use BigKey and Word Disambiguation as conditions.

Chapter 6 draws conclusions from both user studies and discusses both qualitative and quantitative results, suggests design considerations, and provides insight into future studies and improvements.

Chapter 2: Related Work

Text entry is a subject that has been studied for decades. In typical text entry experiments, the time to enter a phrase is recorded while transcribing text to provide a measure of entry speed, while the transcribed text is compared with the original text to measure errors. Entry speed is a key metric for text entry since the goal of text entry techniques is most commonly to offer fast ways of entering text into a system. However, entry speed cannot be looked at by itself; error rate (e.g., number of mistyped characters) is another metric that is used in text entry studies along with entry speed to find a balance between fast text entry speed and acceptable error rates.

Methodology for conducting text entry studies is detailed by Mackenzie [16]; he focuses on the evaluation of text entry techniques and laid out strategies in order to do so. A typical text entry study starts with a working prototype. Users must be able to enter text, and have it displayed as a result. Once this prototype is implemented, substantial pre-experimental testing can begin. The first step is to get a rough idea of the entry speed possible with the technique in question. Next is to decide what the evaluation task will consist of. For text entry studies, the task is commonly to transcribe a set of phrases as quickly and accurately as possibly using some text entry technique as the typing method. Different tasks will have advantages over others, so it is up to the researcher to choose the best task in order to achieve results that make sense in the context of what is being compared. During a text entry experiment, data is collected to be able to assess performance metrics such as entry speed and accuracy. Data for text entry studies can include the time it takes to transcribe a single phrase, the final transcribed phrase, the set of keystrokes (including spacebar, delete, modifier keys…), and error rate.

For the methodology described above, a set of phrases is needed to conduct the study. Mackenzie provides such a phrase set for text entry experiments [18] which provides a consistent metric for text entry research independent of technique used, technology involved, or even the main researcher conducting a study. Mackenzie's phrase set consists of 500 phrases to be used in such studies; this provides easy access to much needed source material to be used in text entry. One way of performing text entry studies is to have participants freely enter text with the text entry method in question; however, this causes problems since there is no metric to measure accuracy without a source to compare with. Phrases in Mackenzie's phrase set have a mean length of 28.61 characters, which makes them ideal for text entry experiments for not being too short or too long. They also provide probabilities of letters and words that might be typed next which is useful when implementing word disambiguation systems; moreover, these probabilities are consistent with normal English. Overall, using this phrase set balances internal and external validity of text input experiments.

2.1 Measuring Text Entry Speed and Accuracy

Wobbrock et al. [2] describe measures of text entry performance in which they include words per minute as a text entry speed measure as well as the minimum string distance as an error rate measure. We now summarize key metrics of text entry performance, several of which we employ in our experiments.

Entry speed is calculated using words per minute as seen in Equation 1 described by Boletsis et al. [3].

$$wpm = \frac{|T| - 1}{S} \times 60 \times \frac{1}{5}$$

Equation 1- Words per minute formula

where S is the time in seconds from the first to the last key press and $|T|$ is the number of characters in the transcribed text (i.e., the phrase length). The constant '60' corresponds to the number of seconds in a minute, and the factor of one fifth corresponds to how many characters compose an average word, which is defined to be 5 characters in text entry experiments. This definition was used to make the results more generalizable with previous studies. We subtract 1 from the length since we start timing the participants as soon as they enter the first key, which means the first key is not timed, so we need to take away 1.

An important metric for measuring text entry accuracy is the **minimum string distance** (MSD) [17, 24, 26, 25]. While text entry speed is a common metric is relatively simple to calculate, accuracy, however, is more complex. Consider that a simple character-by-character comparison of an entered string to a target phrase may not accurately capture user errors. For example, suppose a user meant to enter the word "*quickly*" but mistakenly entered "*qucehkly*". The first two characters are correct, but one might consider everything after the letter 'u' to be incorrect due to displacement of all the letters typed; however, the final three letters 'kly' are correct. A character-by-character comparison would yield seven errors, which clearly does not give a realistic idea of how many mistakes the user actually made.

MSD instead is based on an algorithm used in DNA analysis, phylogenetics, spelling correction, and linguistics for measuring the distance between two strings. As an example, consider two strings 'abcd' and 'acbd'. To calculate the error between these two

strings you could take the transcribed string and delete the c, then, insert a c after the b. This requires two actions to make the strings identical, so MSD = 2. Minimum String Distance is denoted MSD(A, B) where A is the presented text and B is the transcribed text. With this in mind, MSD(A, B) = 0 means A and B are identical. Further refinement for error analysis [17] notes the difficulty of analyzing errors in text entry systems, in that the displacement of letters in certain text entry errors might make correctly typed letters incorrect (see, e.g., the example of *quickly*" vs. "*qucehkly*" mentioned earlier).

The use of MSD for text entry evaluations was pioneered by Mackenzie and Soukoreff [26] by using the Levenshtein distance between two strings, which refers to the minimum number of edits required to transform one into the other. These edits can be adding a letter, removing a letter, or changing a letter to match the target string. To finalize their work, they summarize their efforts along with providing insights in what they coined pathological error correcting [25]. This consists of users noticing they have made a mistake, but they have already typed more letters, instead of using the cursor pointer to correct the error in the proper location, they proceed to hit the backspace key multiple times.

2.2 Text Entry Techniques

Text entry techniques are the input methods for entering text into a system. These can include physical keyboards, touchpads, voice recognition, and even pencils.

There has been extensive research done in the area of text entry using physical keyboards [8, 9, 12]. This type of work typically focuses on conducting text entry experiments using standard keyboards to enter text into a computer system to compare entry speed and accuracy measures. The main focus lies on physical keyboard text entry

studies in the context of VR use. Walker et al. introduced Decoder-Assisted Typing using an HMD and a physical keyboard [29], where it is stated that using a real keyboard in virtual reality presents a key issue which is occlusion (not being able to see the keyboard you are typing on). Occlusion significantly degrades the performance of typing with a physical keyboard in VR for all but touch typists, who are able to type without looking at the keyboard they are typing on. Walker et al. employed an experiment consisting of three conditions, visible typing, occluded typing, and HMD typing. Visible typing outperformed the other techniques, and HMG typing performed the worst, proving that there is a real issue when it comes to occlusion. For their follow-up study [30], an extra feature was added which is displaying a virtual keyboard which presents visual feedback on the key pressed. They performed a study with conditions: Desktop, Desktop Assistant, HMD, and HMD Assistant. The HMD Assistant condition performed better than standalone HMD, which shows that there are viable solutions to occlusion.

Other researchers studied hand visualizations when typing on physical keyboards [8, 9, 12], comparing different hand representations in VR text entry. Knierim et al. [12] employed retro reflective markers to accurately track fingers, with the ability to track fingers they proceeded to conduct a user study to compare different hand representations along with three levels of opacity; full opacity, 50% opacity, and invisible, see Figure 1. The hand representations included realistic hands, abstract skeleton hands, finger tips only, and real world (no VR). It was found that rendering of hands in VR has a significant effect on the typing performance measured using the WPM for inexperienced users in VR, however, the hand representation type had no effect on significance.

Grubert et al. [9] made use of the Logitech Bridge SDK [28]. This method was

employed to be able to track the physical keyboard and hands and display a digital hand representation in VR. The hand representation included no hands, realistic hands, finger tips only, and VideoHand, where a video pass-through allows the user to see their hands in VR. They found no significant results on entry speed; however, participants reported that their preferred technique was VideoHand.

Figure 1 Hand Representations for Text Entry in VR Using a Physical Keyboard [12]

Figure 2 Effects of Hand Representations for Typing in VR [8]

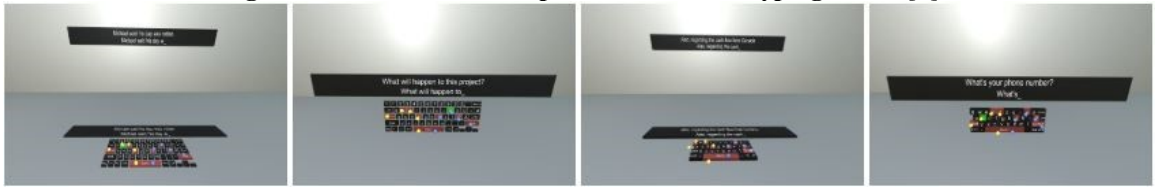


Figure 3 On left side: Physical Keyboard 1 to 1 mapped and repositioned. On Right: Tablet keyboard 1 to 1 mapped and repositioned [9]

Most text entry techniques in VR do not use a physical keyboard, due to the restricted mobility of using a standard keyboard, when VR often requires users to physically move around. The two most common techniques are controller pointing and bimanual text entry. Controller pointing requires spatially tracked controllers to be able to

point and select keys on a soft keyboard. Bimanual text entry presents a soft keyboard split into two sides and requires two touchpads or joysticks to move the cursor on each side of the keyboard to select keys, as can be seen in Figure 4. Bimanual entry commonly employs two joysticks that are moved by each thumb to control one of two cursors on the corresponding side of the virtual keyboard [22]. In a typical text entry experiment, using bimanual entry, the offered entry speeds are at best 15.6 WPM, but on average 8.1 WPM.

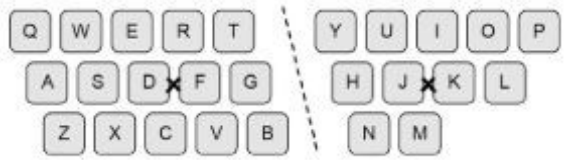


Figure 4 Bimanual Text Entry with Game Controllers

Similar techniques have been included in several studies, which consistently report low text entry speeds between 5.3 and 10 WPM [3, 27]. Previous studies compared bimanual entry with other techniques such as:

- Controller Pointing,
- A drum-like keyboard (CutieKeys [31]), where users hit the keys on a soft keyboard using drumming like motion
- Head-directed input [3], where the orientation of the head defines the selection of keys via a ray cast from the head position
- Freehand [27], where hand tracking is employed to select keys directly
- Controller tapping [27] which is similar to freehand except users select keys by tapping with the controller

Figure 5 and Figure 6 depict examples of these techniques. The drum-like keyboard (Figure 5b) offered the best performance, which shows potential for this kind of text entry technique for VR.

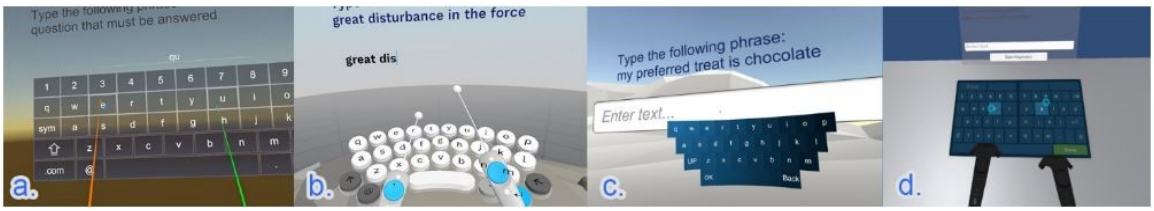


Figure 5 Controller-based VR text-input techniques evaluated and implemented in this study: a. Controller Pointing, b. Drum-like keyboard, c. Head-directed input, and d. Split keyboard [3].

Figure 6 Selection-based Text Entry in VR [27]

We finish our exploration of VR text entry techniques by mentioning BigKey by Faraj et al. [1]. BigKey resizes keys on the virtual keyboard in real-time. It uses the probability of a letter being typed *next* to increase the size of the predicted next key. For example, upon typing the character 't', the character 'h' is likely to be the next key due to the frequency of the "th" digraph in English. Similarly, upon typing 'q', the 'u' key is almost certain to follow, while other keys like 'z' for instance, are almost certain *not* to follow. In the former case, BigKey would increase the relative size of the 'h' key after striking the 't' key, and increase the size of the 'u' key after striking the 'q' key (while decreasing the size of keys like 'z'). According to Fitts' law, larger targets (at otherwise equal distance) are faster to select [15]. An example of BigKey in action is seen in Figure 7.

Figure 7 BigKey virtual keyboard while selecting the word "the" [2]

In this section, we discussed key works in text entry. Our work builds off of these by employing and further developing the text entry techniques discussed. In the next chapter we present our fluid interaction keyboard which is based on CutieKeys, as well as adapting BigKey to controller pointing and the fluid interaction keyboard techniques. We also take advantage of previous work in text entry measures for entry speed and accuracy as described by Equation 1 and MSD, which we use to analyze results from our studies. Finally, we use Mackenzie's phrase set [18] as the text to be transcribed in our studies' task, which is further detailed in Chapter 3.

Chapter 3: Text Entry in Virtual Reality

This section describes the implementation of our text entry testbed, BigKey, word disambiguation, and the input techniques evaluated in the studies.

3.1 Text Entry Testbed

Noting the need for an experimental framework to conduct text entry studies in VR, we first developed a text entry experiment testbed. The design and implementation of the text entry testbed was targeted as a flexible text entry experiment framework developed with Unity 3D 2018.3.7f1. It was developed for conducting user studies on text entry in VR for the purpose of this thesis. However, it doubles as a general purpose text entry experiment tool since it can be adapted for use with any (non-VR) text entry technique. Researchers can develop their text entry techniques and easily integrate it with the text entry testbed for initial performance test, up to full scale user studies.

The design and development of the testbed is considered the first contribution of this thesis. The testbed is available for download at http://sotocodes.com/text-entry-test-bed/ and its source code is also available at https://github.com/sotoea/text-input-test-bed. Detailed instructions on how to use and contribute to the development of the testbed are available at the above links. We made the testbed publicly available in hopes to continuously make improvements to the experiment software and make it more accessible and functional over time.

The following subsections detail the sequence of events of a text entry experiment employing the text entry testbed.

3.1.1 Experimental Setup Screens

Upon starting the testbed, the experimenter is presented with the welcome screen (Figure 8). The experimenter enters a participant ID, the number of phrases that will be presented to the participant in each block, and the number of blocks (i.e., repetitions of all experiment conditions). For example, if 10 phrases are chosen, and 3 blocks are chosen, 30 phrases will be typed overall split into 3 blocks.

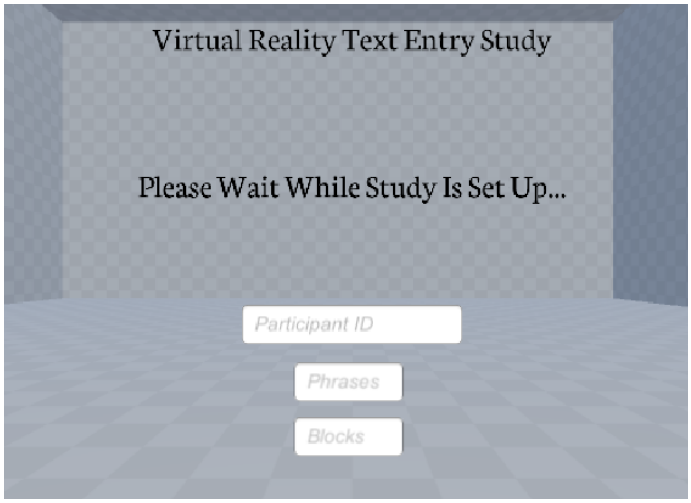


Figure 8. Participant VR view of the text entry study welcome screen. Initial screen where the experimenter selects Participant ID, number of phrases, and number of blocks. At this stage, the only components displayed to the participant are the title and "Please Wait" labels in the background.

Upon proceeding, the experimenter is then prompted to select a text entry method on the next screen (Figure 9). The input technique selection screen presents a list of options which is customized by the experimenter depending on what text entry techniques they want to include in the evaluation. If the study requires multiple levels of independent variables to choose from, additional options will appear allowing the experimenter to choose each condition (Figure 9). Once everything is set and an initial text entry method to test is chosen, the system will start with the first block of the chosen method.

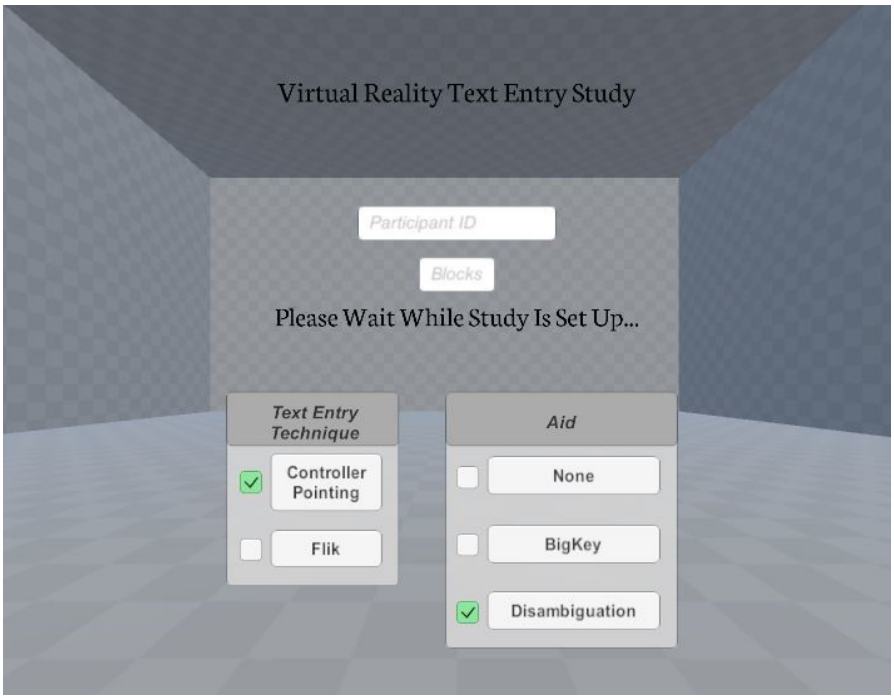


Figure 9 Text Entry Technique Selection Screen. The experimenter selects the desired text entry condition and aid using the mouse. This screen is seen by the experimenter while the participant only sees the label 'Please wait while study is set up'.

In the background, the software contains a singleton variable which holds all relevant information of the current experiment setup, including Participant ID, number of phrases, current phrase, number of blocks, current block, and experiment condition. This is all the necessary data for a study.

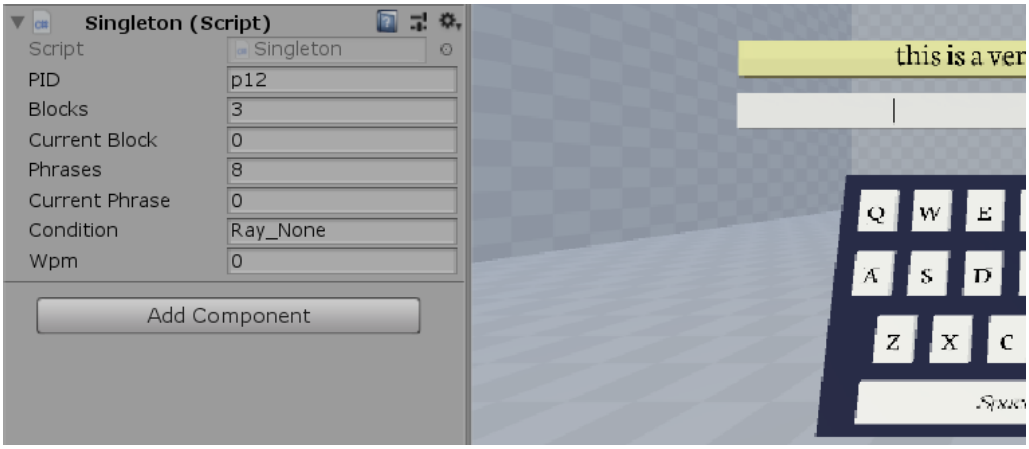


Figure 10 Text entry Test-Bed Singleton window

Once the experiment has started, the singleton (Figure 28) contains the necessary data to keep track of the experiment and record data to log files.

3.1.2 Starting the Experiment

Upon starting the first block and any subsequent block, the current Unity scene will get information from the setup singleton to create a new unique log file. In the scene, there are three main components apart from the singleton: The phrase to be transcribed, the text entered so far, and the text entry technique are the main components that the participant is able to visualize in the scene. The testbed currently uses Mackenzie's phrase set of 500 phrases as the default phrases, which are commonly used in text entry experiments [18], but these can be changed by replacing a file called 'phrases.txt' with any other phrase set as long as it follows the same format as Mackenzie's phrase set text file.

As each text entry trial commences, the participant sees the target phrase (i.e., the phrase to be transcribed), the text they have entered so far in a separate text field, and any components that need to be visualized and controlled (e.g., rays, virtual keyboards, etc.) required for a given text entry technique. See Figure 11. The testbed randomly selects a phrase from the phrase set; this is the "target" phrase, and it is presented to the participant in the phrase to transcribe section and will wait for the participant to enter the first letter, this is repeated for the number of phrases selected by the experimenter for each block. When the participant enters the first character, a timer starts. The timer stops once the participant hits ENTER or an equivalent key on the virtual keyboard being used. The timer measures how long it took the participant to enter the phrase. While the participant is entering text, the system records the raw input stream, which corresponds to all

characters selected in order; this includes space, backspace, and modifiers.

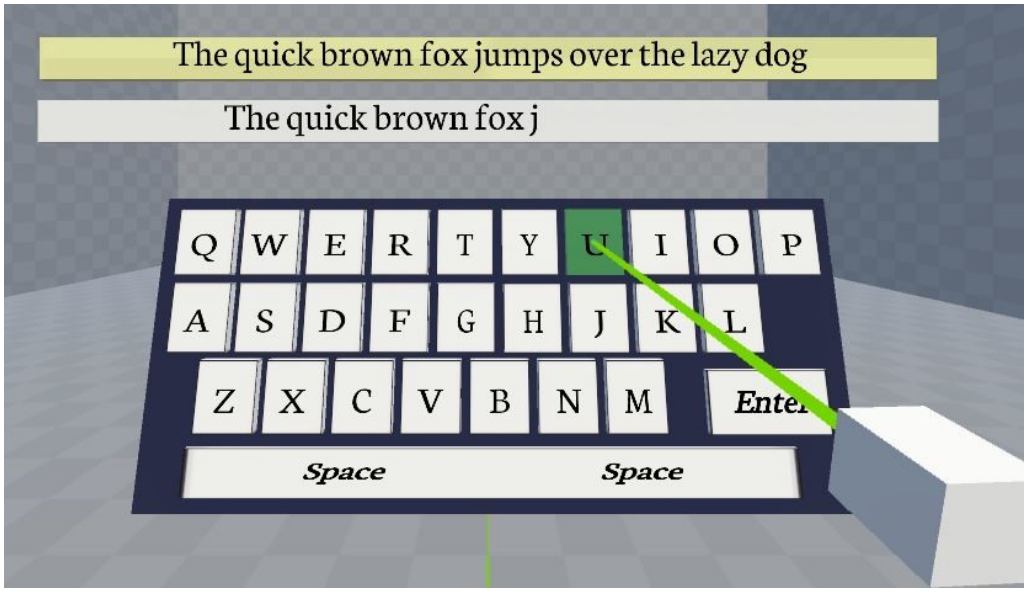


Figure 11 Elements presented to participants during text entry experiment

When the participant completes a phrase and hits ENTER, the testbed records all relevant details to a log file, which is structured as seen in Figure 12. In the log file, each trial appears as a single row, and includes experiment setup details such as the participant ID, text entry technique, current condition, block number, the phrase to be transcribed, the text that is actually transcribed by the participant, the raw input stream, time, the calculated entry speed in words per minute, and error rate in minimum string distance [2].

Participant ID	Text Entry Tecnique	Condition	Block	Phrase	Transcribed	Keystrokes	Time	MSD	Length Phrase	Length Trans	wpm	Error Rate %
1	Ray	None	1	bad for the environment	bad for the environment	bad-for-the-environment	16.1202	0	23	23	16.37696803	0.00%
1	Ray	None	1	we park in driveways	we paek in driveways	we-paek-in-driveways	18.9295	1	20	20	12.04469215	5.00%
1	Ray	None	1	what a monkey sees a monkey will do	what a monkey sees a monkey will do	what-a-monkey-sees-a-monkey-will-do	29.9984	0	35	35	13.60072537	0.00%
1	Ray	None	1	world population is growing	world population is growing	world-population-is-growing	21.6486	0	27	27	14.41201741	0.00%
1	Ray	None	1	what a lovely red jacket	what al lovely red jacket	what-al-lovely-red-jacket	17.1598	1	24	25	16.78341239	4.00%
1	Ray	None	1	an excellent way to communicate	an excellent way to communicate	an-excellent-way-to-communicate	24.3003	0	31	31	14.81463192	0.00%
1	Ray	None	1	he is still on our team	he is still on our team	he-is-still-on-our-team	17.5646	0	23	23	15.03023126	0.00%
1	Ray	None	1	not quite so smart as you think	not quite so smart as you think	not-quite-so-smart-as-you-think	27.0893	0	31	31	13.28937994	0.00%
1	Ray	None	2	my bike has a flat tire	my bike has a flat tire	my-bike-has-a-flat-tire	15.3452	0	23	23	17.20407684	0.00%
1	Ray	None	2	effort is what it will take	effort is what it will take	effort-is-what-it-will-take	22.0915	0	27	27	14.12307901	0.00%
1	Ray	None	2	she wears too much makeup	she wears too much makeup	she-wears-too-much-makeup	18.5272	0	25	25	15.54471264	0.00%
1	Ray	None	2	this equation is too complicated	this equation is too complicated	this-equation-is-too-complicated	20.7602	0	32	32	17.91890252	0.00%
1	Ray	None	2	I took the rover from the shop	I took the rover from the shop	I-took-the-rover-from-the-shop	20.9074	0	30	30	16.64482432	0.00%
1	Ray	None	2	join us on the patio	join us on the patio	join-us-on-the-patio	19.6099	0	20	20	11.62678035	0.00%
1	Ray	None	2	an inefficient way to heat a house	an inrifficient way tobheat a house	an-inrifficient-way-tobheat-a-house	28.6977	2	34	34	13.79901525	5.88%
1	Ray	None	2	meet tomorrow in the lavatory	meet tomorrow in the lavatory	meet-tomorrow-in-the-lavatory	27.8705	0	29	29	12.05575788	0.00%
1	Ray	None	3	companies announce a merger	companies announce a merger	companies-announce-a-merger	22.102	0	27	27	14.11636956	0.00%
1	Ray	None	3	the stock exchange dipped	the stock exchange dipped	the-stock-exchange-dipped	21.308	0	25	25	13.51605031	0.00%
1	Ray	None	3	this is a very good idea	this is a very good idea	this-is-a-very-good-idea	17.0346	0	24	24	16.20231764	0.00%
1	Ray	None	3	so you think you deserve a raise	so you think you deserve a raise	so-you-think-you-deserve-a-raise	20.5906	0	32	32	18.06649636	0.00%
1	Ray	None	3	look in the syllabus for the course	look in the syllabus for the course	look-in-the-syllabus-for-the-course	26.7766	0	35	35	15.23718471	0.00%
1	Ray	None	3	protect your environment	protect your environment	protect-your-environment	14.0793	0	24	24	19.60324732	0.00%
1	Ray	None	3	be persistent to win a strike	be persistent to win a strike	be-persistent-to-win-a-strike	19.4224	0	29	29	17.29961282	0.00%
1	Ray	None	3	a big scratch on the tabletop	a big scratch on the tabletop	a-big-scratch-on-the-tabletop	20.7252	0	29	29	16.21214753	0.00%

Figure 12 Sample log file for one participant covering 3 blocks with 8 phrases per block using one text entry technique using a particular modifier condition.

3.1.3 Intermission Screen

Once the participant completes all phrases in a given block, the log file is closed and is ready to be processed, the system will then move on to an intermission screen where the participant can take a break. This intermission also provides a chance for the experimenter to ask for any feedback or to fill out any questionnaires that might be needed. Once ready, the next block begins.

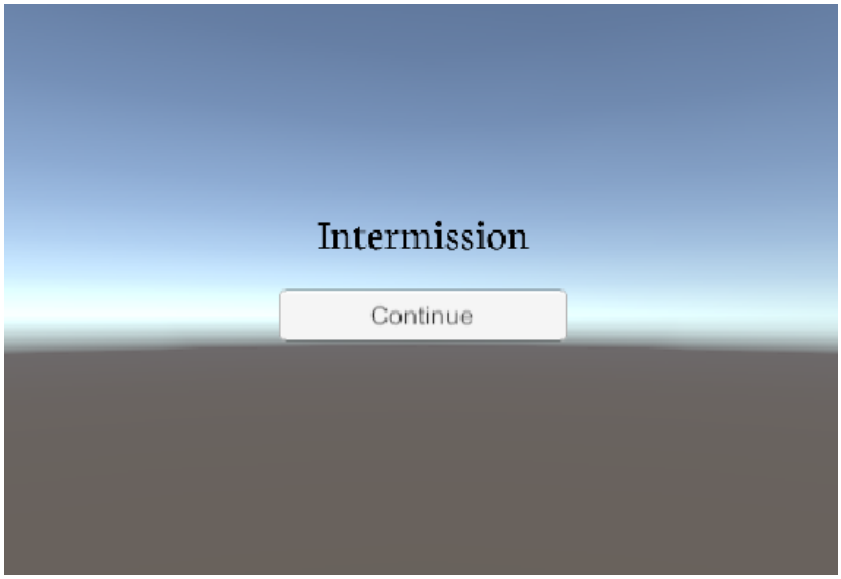


Figure 13 Intermission screen shown in between blocks. This is the screen where participants can take a short break before starting on the next block. Viewed by participants.

3.1.4 Finishing the Experiment

Upon completing all blocks for a given condition, the system returns to the welcome screen. The experimenter can then continue with the next experimental condition, or provide participants with a questionnaire, or complete the experiment as appropriate.

3.2 Text Entry Techniques

This subsection describes the main four text entry techniques of the two

experiments described in Chapters 4 and 5 of this thesis. These techniques include controller pointing, continuous cursor selection, physical keyboard, and fluid interaction keyboard (FLIK), as well as the two "aids" (BigKey and word disambiguation). These text entry techniques as well as the aids were included not only to empirically compare the techniques, but to also validate the text entry testbed itself. The techniques are chosen due to their standard use in the industry; Controller Pointing and continuous cursor selection being widely used as the main technique in the VR industry, physical keyboard due to its high performance in non-VR applications, and fluid interaction keyboard as a novel technique to compare. The two experiments, detailed in the following chapters, were designed to necessitate modification to the text entry testbed to support incorporating different techniques and aids, as a way to assess the effectiveness and ease of use of the testbed as an experimental tool. Overall, the objective was to support multiple text entry techniques and the evaluation of various combinations of text entry techniques and aids, while simultaneously collecting performance data on VR text entry methods. In the following subsections, we describe the text entry techniques used, which are defined as the method in which you enter text into a computer system, as opposed to keyboard layout, which is the layout individual keys are presented to the user on physical or virtual keyboards. For example, entering text input on a smart phone using traditional tap keyboard versus using a swipe method are both considered text entry techniques, whereas QWERTY and DVORAK are two different keyboard layouts a given text entry technique can adopt. For our studies, we implement the QWERTY keyboard layout on our text entry techniques since it is more commonly used in text entry environments, as well as keyboard layouts being beyond the scope of this thesis.

3.2.1 FLIK (Fluid Interaction Keyboard)

One of our main contributions to this thesis is the FLIK text entry technique. This text entry technique is inspired by CutieKeys [31]. CutieKeys employs a drumstick metaphor, where users enter text by swinging the VR controller like a drumstick to hit pads corresponding to keys organized in the standard QWERTY keyboard layout. However, unlike CutieKeys, FLIK supports contacting keys from both top and bottom directions. In our implementation of FLIK, the virtual keyboard presents every key as a sphere, positioned far enough from its neighbor to prevent overlap or mistaken keystrokes due to accidental collisions between the cursor and the key sphere. FLIK uses bimanual 3D tracked controllers. A selection cursor sphere is attached to the tip of each controller; these selection spheres are what the user manipulates, and intersects with the key spheres to issue a specific keystroke. Intersection between the selection cursor and the key sphere simultaneously selects *and* confirms the key input; no further button presses are required with FLIK. Upon intersecting a key sphere, the user can move in arcs in both direction, and loop back through the keyboard to the desired key (unlike CutieKeys, which would require they move back (upwards) from below upon striking a key, then move back down *again* from above to strike the next key). Although the difference between FLIK and CutieKeys is subtle, we anticipate that this minor difference will improve entry rates due to the continuous movement of the hands. In contrast, with CutieKeys, the user has to tap on a key, and then reverse the direction of movement to stop selecting the key. See Figure 14.

Figure 14 Fluid Intersection Keyboard (FLIK). (A) Shows the phrase to be transcribed before the user has entered any text. (B) The user selects the character 'n' by moving his right hand through it so that the red selection cursor touches and goes through the character as seen in (C) where the sphere cursor is now behind (and partially occluded by) the 'n' character key sphere. (D) Again, shows the same right hand having looped around the bottom of the keyboard and selecting the 'o' character key sphere from beneath. (E) As the right hand exits the character 'o', the left hand now moves through the spacebar to select a space character. (F) User continues in this manner selecting the next character.

3.2.2 Controller Pointing

Controller pointing requires using two 6DOF controllers. A selection ray is emitted from each controller, which allows a participant to bimanually (using both controllers, one in each hand) remotely point at a virtual (i.e., soft) keyboard presented in front of them. Individual keystrokes are issued upon pressing the primary thumb button on the controller currently being used for selection. For extra visual feedback, the key currently intersected by the selection ray changes colour. The controllers also vibrate to confirm selection when the button is pressed. Auditory feedback in the form of a "click"

sound is also provided. Both rays pointing at the same key would not have any side-effects. See Figure 15.

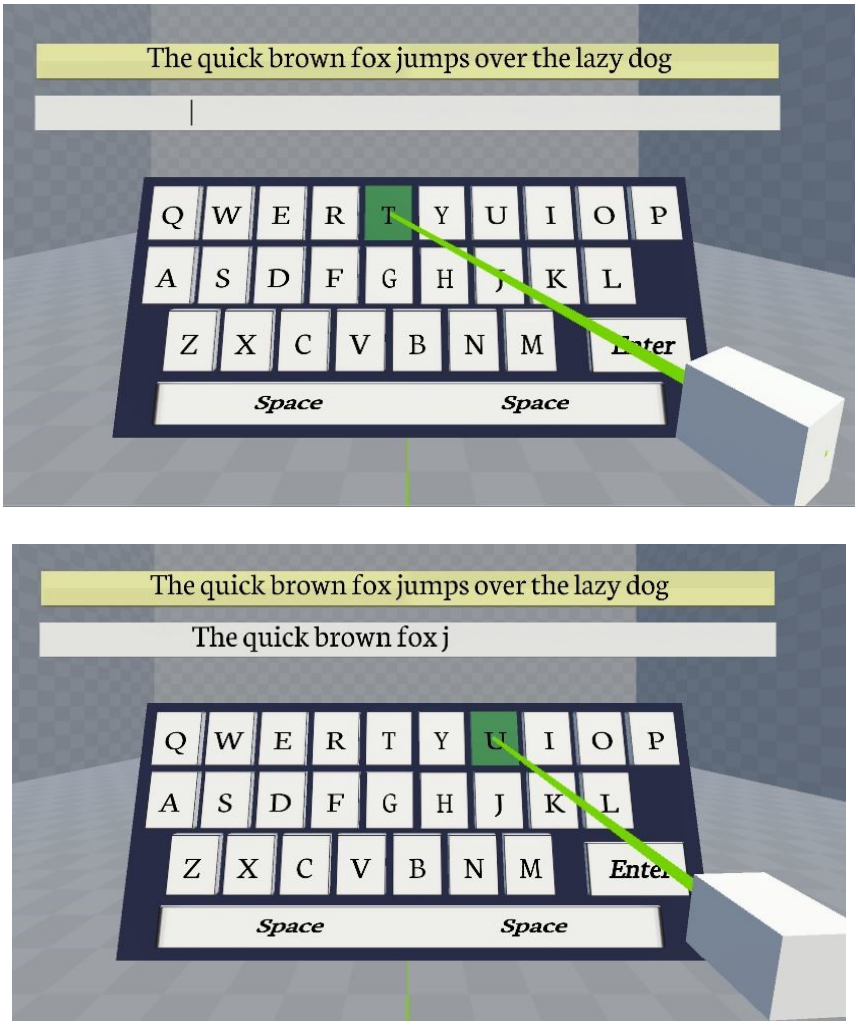


Figure 15 Images showing the Controller Pointing keyboard.

3.2.3 Continuous Cursor Selection

With continuous cursor selection, the virtual keyboard is divided in left and right halves to support bimanual entry. Keys on the right half of the keyboard are selected via a cursor controlled by the right hand and vice versa. A 2D selection cursor is initially placed at the midpoint on each keyboard half. These selection cursors are controlled by moving the corresponding (right or left) joystick (e.g., with an Oculus Touch controller)

or sliding a finger/thumb on the controller's touchpad (e.g., on an HTC Vive controller). The cursors move in a continuous motion in the direction of joystick or touchpad movement. Once the cursor hovers over the desired key on the virtual keyboard (half), a button press on the corresponding controller confirms selection and issues the keystroke. The same feedback mechanisms described above for the controller pointing technique are employed with this technique as well. For the purpose of this thesis, feedback mechanisms are not treated as an independent variable but instead are part of the text entry technique as whole. However, with minimal modification to the testbed, feedback mechanisms could be included as a separate independent variable if desired. See Figure 16.

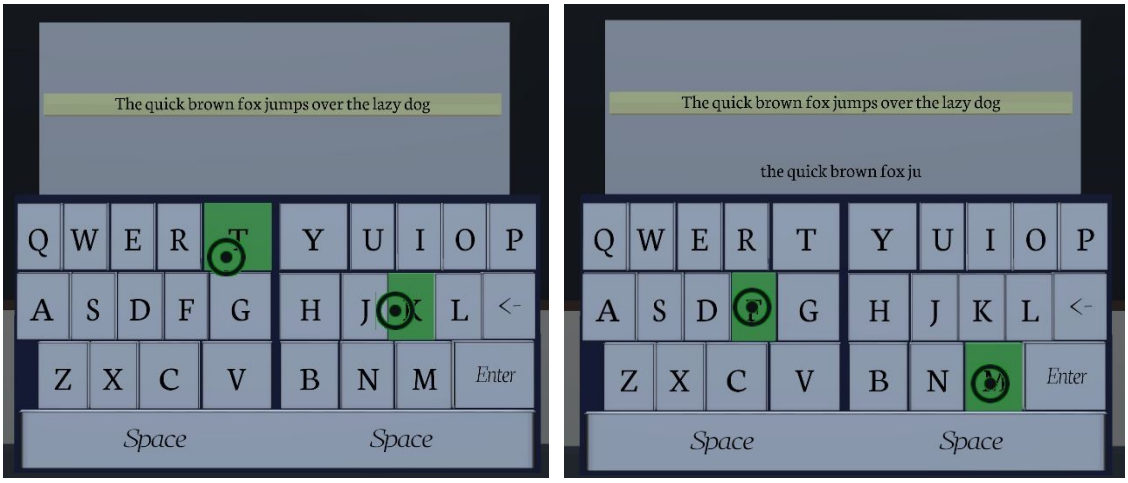


Figure 16 Continuous Cursor Selection. By dragging each thumb on their respective touchpads, the cursors move on their respective half in order to select characters from the virtual keyboard.

3.2.4 Physical Keyboard

Lastly, there is a tracked physical keyboard with virtual hand representations by using Logitech Bridge API [28] in combination with an HTC Vive individual tracker mounted on a keyboard. The hand representations are provided by the Logitech Bridge API via the HTC Vive's integrated camera and Logitech's proprietary computer-vision

hand recognition techniques. This technique behaves the same as using a physical keyboard except that the user sees a virtual representation of the keyboard (and a video image of their hands) rather than the physical keyboard itself.

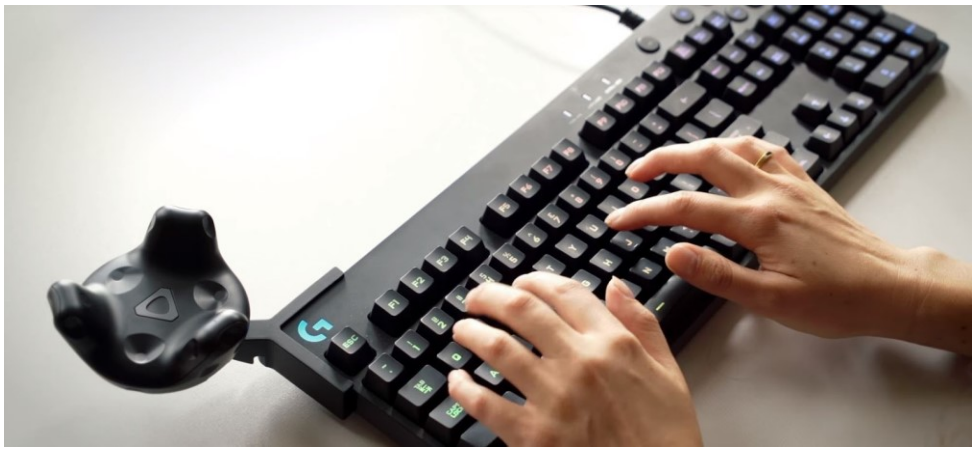


Figure 17 Logitech Bridge real world setup with individual vive tracker [28]

Figure 18 Logitech Bridge virtual keyboard with hand representations [28]

3.3 Text Entry Aids

The testbed supports the following add-on features that can be applied with any of the above text entry techniques (or any other that could be added). This demonstrates the flexibility of the testbed in providing a way to add different techniques with varying

levels of complexity. The testbed currently supports two such aids, BigKey and word disambiguation, described below.

***BigKey*:** BigKey [1], detailed in chapter 2, employs a custom algorithm which operates by analyzing the input stream of the current word. The higher the probability a letter will be typed next, the bigger the size of its key on the virtual keyboard. The algorithm starts when an initial character for a new word is entered (i.e., following press of the SPACE key). Prior to entry of the initial character for a new word, all keys are the same average size.

Following the first keystroke for the new word, the algorithm retrieves all possible words that start with this character provided in the phrase set used by the testbed. Since the testbed included only a fixed phrase set size, there are a limited number of possible words that can be entered, so the overall computing time for creating an array of such words is short. Prior to entering the next character, the system calculates the frequency of all subsequent next characters. This is done by counting the occurrences of each character in sequential order, starting with 'a', 'b', 'c', and so on, that directly follow the entered character for all words in the array. After this calculation is complete, all letters with 0 frequency (i.e., those that *never* follow the initial character) are scaled to the smallest size, while the letter or letters with highest frequency, are scaled up to the maximum size. Letters with frequencies between 0 and the highest frequency are scaled by a factor multiplier which starts with the scaling factor of the letter with the highest count. The highest ranked letter would be scaled up by a factor of 1.75, the second by a factor of 1.7, and so on in decrements of 0.05. This ensures that all characters that *might* follow the initial character receive some scaling multiplier, with the more probable ones

being scaled proportionally larger. See Figure 19.

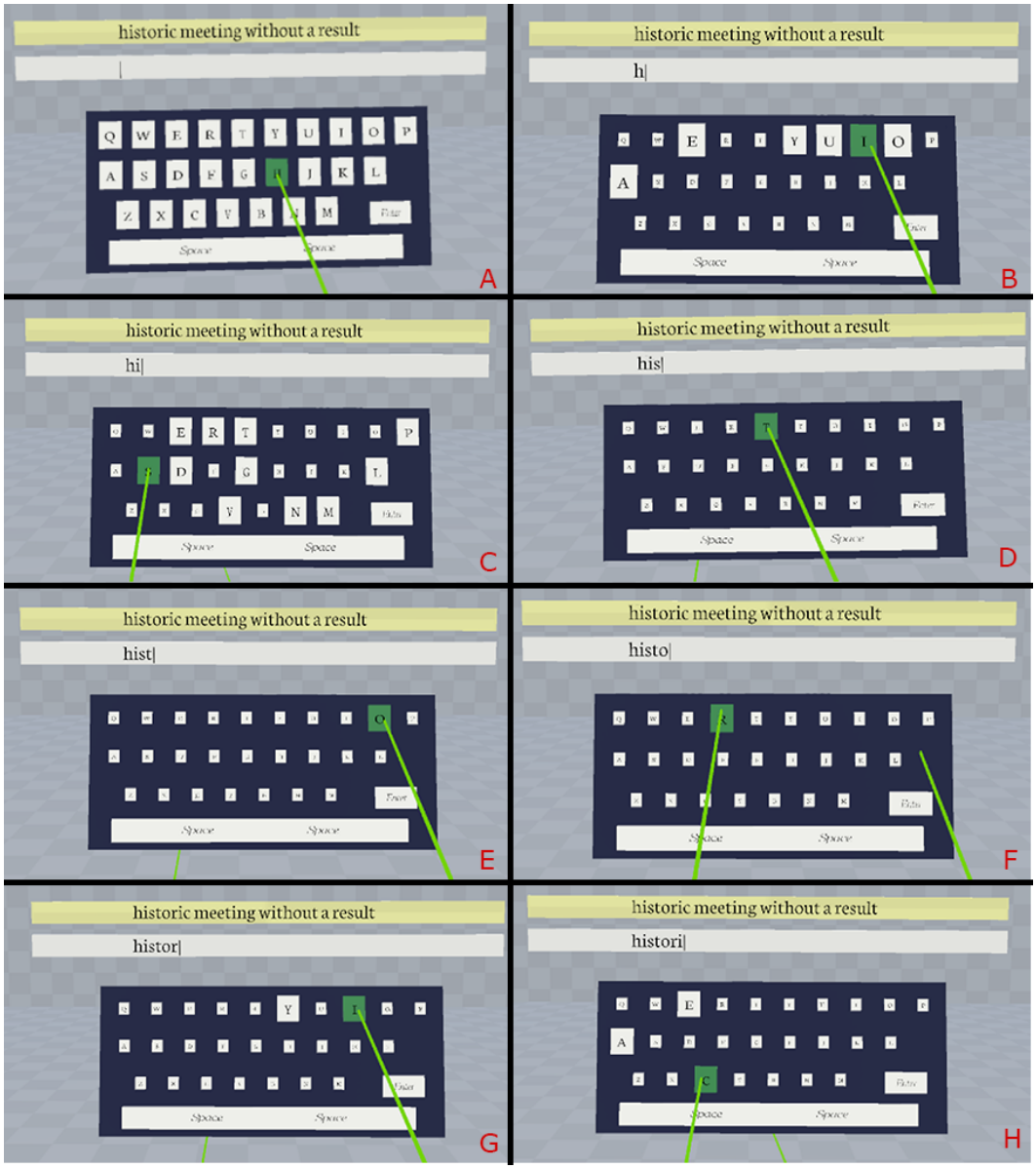


Figure 19 Image showing keys rescaling on virtual keyboard based on BigKey algorithm. In every slide, a new character is selected, and the rest of the characters are resized based on the words found with those characters as an option to be typed next.

BigKey operates as follows. After the initial keystroke the word list is filtered using the first letter. All words that do not contain this new character as their next character are removed from the list. The algorithm then repeats this process, refining the list and calculating new scale factors for all keys on each subsequent keystroke. Upon

finishing the word (i.e., the SPACE key is pressed), the words list is cleared and every key is rescaled to the default size in preparation for the next word.

According to Fitts' law [15], dynamically resizing keys in this fashion decreases the difficulty (and thus selection time) in selecting a more probable next key, while increasing the difficulty (and thus selection time) in selecting a less probable next key. Using BigKey, the virtual keyboard dynamically changes individual key sizes each time the user enters a character. The expectation is that this technique will on average, improve text entry speed since the original work on BigKey demonstrated significant results in favor of BigKey [1]. However, there is a high likelihood that it *decreases* text entry speed when the user makes a mistake, due to how the algorithm analyses the next possible characters. In situations where the user enters the wrong key (e.g., mistakenly hits the wrong key), BigKey provides incorrect resizing from what is expected, or might even not resize at all since the characters typed might not resolve to any word – and thus compounds the effects of errors. In order to make it easier for users, the keys scale to their default size if a wrong sequence of characters is entered and they cannot resolve any words.

Word Disambiguation: In every language, there are frequencies of how often each word appears. From these frequencies, one can implement a general Word Disambiguation system that can be used with most text entry scenarios. The testbed implements this in the form of suggestions, every time the user enters a key, the system computes the most likely word to be typed in order of ranking, then it displays the top three ranked words as selection options to complete a word in one selection rather than finishing the whole word. See Figure 20 for an example.

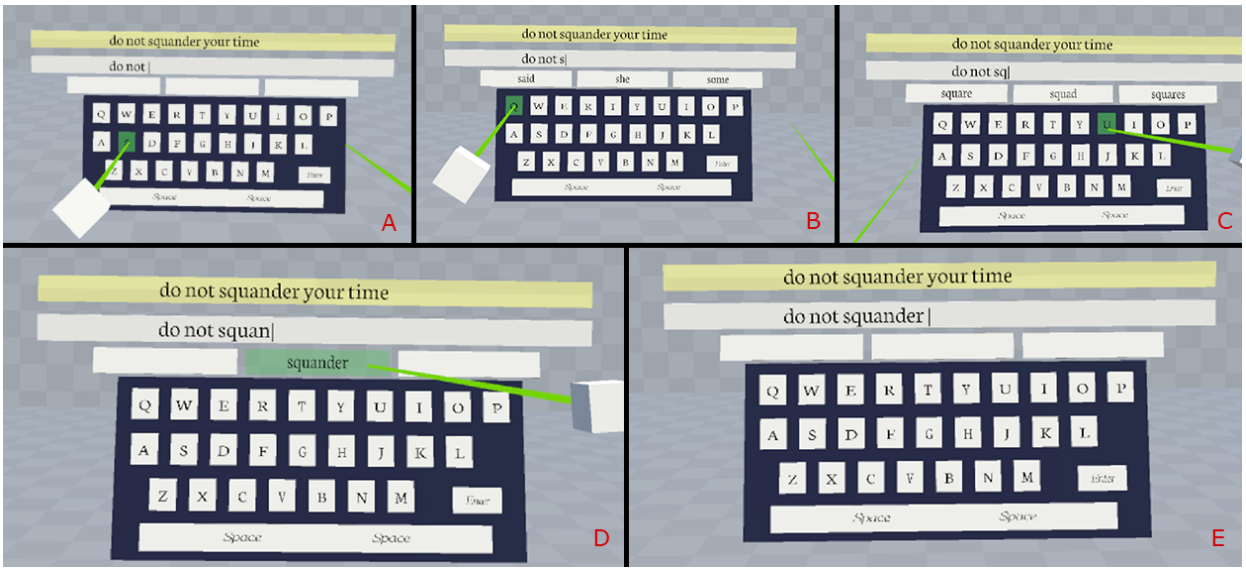


Figure 20 Image showing how word disambiguation is shown to the user. (A) (B) and (C) show how the list of suggestions gets updated with every key press, while (D) and (E) show the selection of a suggested word and the addition of a space character after selection is complete

This system works in a similar way to how key prediction is implemented for BigKey. The main difference is that with disambiguation, the ranking for every word in the phrase set is used. Along with Mackenzie's phrase set, a list of word rankings is provided. This list contains all words used in the phrase set along with their ranking. The rankings are in the form of a simple integer number, representing the frequency of use of this word. Following the same method as BigKey, where a list of words is gathered once a character is entered, the array of possible words can then be ordered in terms of their ranking, after which the top three ranking words are simply selected and displayed as selection options to the participant. Note that the ranked words in the list correspond to possible words containing the keys that have been typed so far, it does not predict the next word to come, but instead suggests three different words to complete the current word.

For text entry studies, this method works well due to the limited phrases and words used in the phrase set. This also allows for consistency between studies since the sample size of the phrases remains fixed, allowing for better comparative analysis of results between different studies. Ease of implementation is also a big factor when using such a predictive algorithm for both word disambiguation and BigKey. Using the Mackenzie phrase set allows for researchers to create these types of techniques without worrying about optimization and more complicated prediction algorithms which are fields of their own.

Chapter 4: User Study 1

This chapter presents the first user study using the text entry testbed described in Chapter 3. In addition to comparing four VR text entry techniques, the study also serves as a validation of the text entry testbed, which is also employed in the second user study described in Chapter 5. This first experiment consists of a performance comparison of the four text entry techniques for virtual reality: Controller Pointing, Continuous Cursor Selection, FLIK, and Physical Keyboard. This study was approved under CUREB-B file number 109263.

4.1 Hypotheses

The following hypotheses for this first user study are decided based on pre-testing of the text entry techniques used:

H1: Physical keyboard will perform the best, followed by FLIK, and controller pointing coming in third, but not far behind FLIK. It is expected that continuous cursor selection will offer the worst performance.

H2: Continuous cursor selection will yield the lowest fatigue, followed by physical keyboard, and lastly FLIK.

4.2 Participants

This study included 24 participants, 14 male, 10 female, and aged 18 – 30 (*SD* = 2.96). Recruitment was done through flyers posted around campus as well as word of mouth. Participants were asked before-hand if they were touch-typists or not (if they are able to type on a keyboard effectively without looking at the keyboard) of which 15

claimed to be touch-typists. All participants stated that they are comfortable with basic or advanced computer use.

4.3 Apparatus

4.3.1 Hardware

The experiment was conducted using a VR-ready laptop with an Intel core i7-6700HQ quad core processor, an Nvidia Geforce GTX 1070 GPU, and 16GB of RAM, running the latest build of Microsoft Windows 10. The Logitech G810 was used as the physical keyboard due to integration requirement with the Logitech Bridge SDK. We used the HTC Vive HMD with the touchpad enabled 6 degree of freedom controllers. The HTC Vive provides a 1080 x 1200-pixel resolution per eye, 90 HZ refresh rate, and 110 degrees of field of view.

To minimize potentially confounding variables, participants were always seated in our lab, with a wide radius of free space around them to maximize range of movement. Participants used the HTC Vive controllers as the main input device in all conditions, with the exception of the physical keyboard condition, where they instead used the tracked keyboard. Participants pressed the trigger buttons on the HTC Vive controllers to confirm selections where applicable and used the touchpad on the controllers for the continuous cursor selection portion of the experiment, see Figure 21.

Figure 21 Vive controller inputs

4.3.2 Software

The study employed the text entry testbed described in Chapter 3. It was developed in Unity 3D and it integrates easily with any text entry technique. The study included the following text entry techniques: Controller Pointing, Continuous Cursor Selection, FLIK, and Physical Keyboard (see Chapter 3 for detailed descriptions). Each text entry technique used in this study was developed in Unity independent of the text entry testbed and was then integrated with the testbed in order to use the full functionality of the system. Once all the text entry techniques have been integrated and tested with the text entry testbed, the experiment can begin by providing the number of blocks and number of phrases. The integration process is described in Chapter 3. All necessary data is logged by the system into unique log files to avoid overwriting previous data.

4.3.3 Procedure

Upon arrival, participants were asked to read the consent form provided in Appendix A. They were informed of their right to withdraw from the experiment at any point without any obligation to finish the experiment if at any time they felt

uncomfortable or nauseous. Participants were then asked to try the HMD, and were presented with their first text entry technique (based on counterbalancing order). They were given around two minutes to practice with the technique, to become familiar with the general system and the first text entry technique they would use. These practice trials were not recorded. After this practice period, the study began with the same text entry technique used in the practice round.

The task involved transcribing presented phrases using the current text entry technique. Participants performed 3 blocks with each of the four text entry techniques, where each block consisted of 8 phrases to be transcribed. Once a block completed, participants took a short break if desired and continued with the next block once they were ready, when all three blocks were finished, the next text entry technique was chosen and the process repeated from trying out and practicing the technique for 2 minutes. In between each technique, participants were also asked to fill out the NASA-TLX questionnaire provided in Appendix A. During the breaks, participants were encouraged to provide any feedback about the current technique. Participants were instructed to transcribe the phrases presented as quickly and accurately as possible but were not told to prioritize either-or in order to not be biased towards speed or accuracy.

In this study, error correction was allowed via the backspace key, but it was not necessary to transcribe the phrases perfectly in order to continue. At the end of the study, participants were asked to fill out one last questionnaire asking them to rank each technique as well as provide any additional feedback they might have. This questionnaire is also provided in Appendix A.

4.4 Design

The experiment employed a within-subjects design, with two independent variables, text entry technique with four levels (FLIK, controller pointing, physical keyboard, and continuous cursor selection) and block with three levels (block 1, 2, and 3). Blocks are used as independent variables since they can potentially reveal learning curves in the resulting data. The order of text entry technique was counterbalanced using a Latin square.

We recorded three dependent variables (entry speed, error rate, and NASA-TLX scores). Entry speed was measured in words per minute (WPM), and calculated as seen in Equation 1 from Chapter 2 and detailed in Chapter 2. Error rate was based on the MSD, see Equation 2 and Chapter 2 for full discussion. TLX scores are the overall results from the NASA-TLX questionnaire. Exit questionnaires were given once the experiment was done. Across all 24 participants, this yielded *3 blocks × 8 phrases × 4 techniques × 24 participants = 2304 phrases transcribed*

$$Error\ Rate\ \% = \frac{100\ \times MSD(P,T)}{\max(|P|,|T|)}$$

Equation 2 - Error rate formula

4.5 Results

4.5.1 Task Performance

In this subsection, we present quantitative results from the first study in relation to speed and accuracy. They are computed per participant, for each text entry technique, and averaged per block.

Starting with text entry speed, results ranged from 12.32 WPM for Continuous Cursor Selection to 49.56 WPM for Physical Keyboard. See Figure 22 for full results.

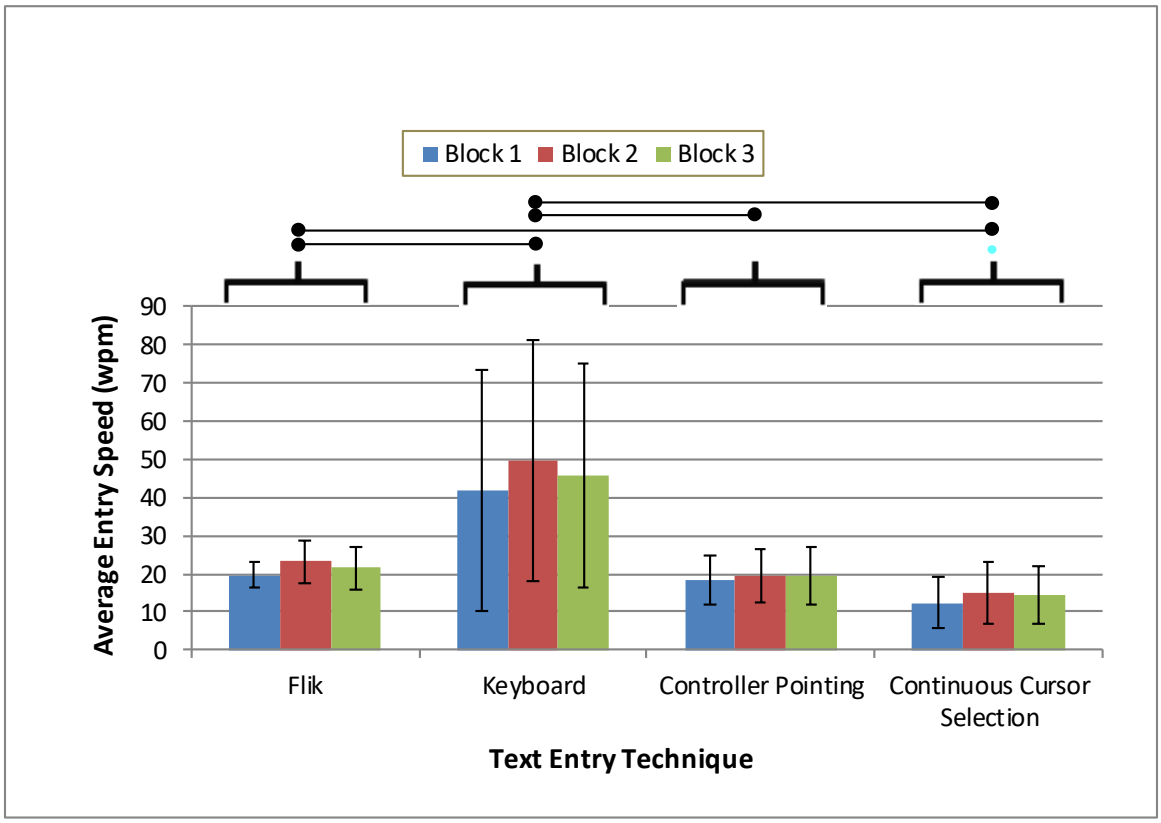


Figure 22 Text Entry Speed Average Comparison Chart. Error bars show standard deviation. Black horizontal bars show pairwise significant differences between some text entry techniques.

Repeated-measures analysis of variance revealed that the effect of text entry technique on entry speed was statistically significant ($F_{3,\ 69} = 54.886$, $p < .0001$), as was the effect of block on entry speed ($F_{2,\ 46} = 88.432$, $p < .0001$). A Post Hoc Bonferroni-Dunn test at the $p < .05$ level revealed pairwise significant differences between some text entry techniques. These pairwise differences are represented as horizontal bars on Figure 22.

For error rate, results ranged from 1.83% error for physical keyboard to 4.03% for continuous cursor selection. See Figure 23.

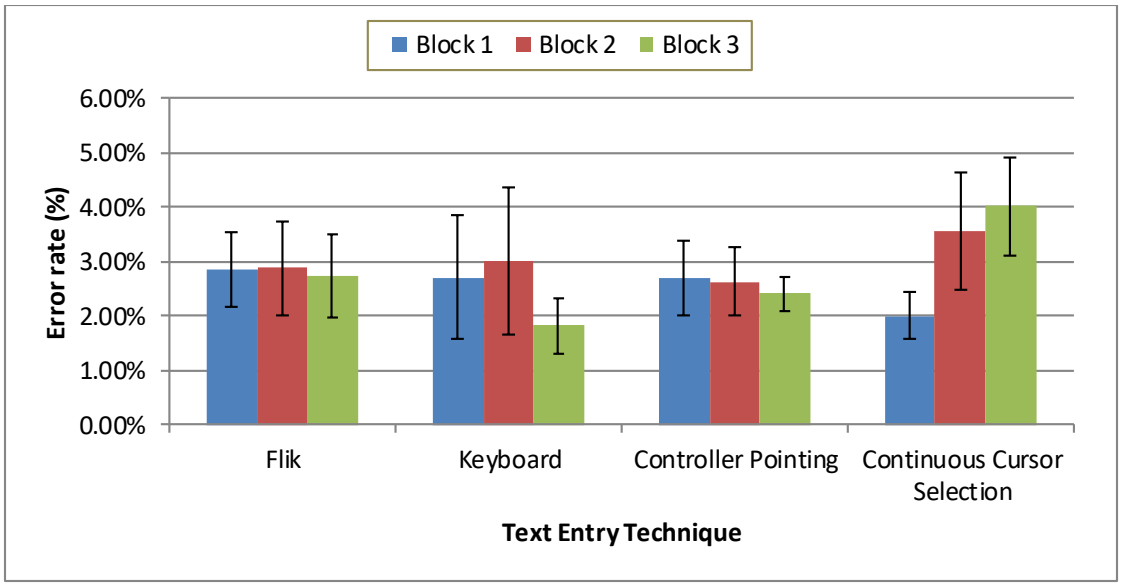


Figure 23 Average Error Rate (%) Comparison chart. Error bars show standard deviation.

Analysis of variance revealed that the effect of text entry technique on error rate was not statistically significant ($F_{3,\ 69} = 0.431$, ns), nor was the effect of block ($F_{2,\ 46} = 1.681$, $p > .05$).

4.5.2 User Preference

Workload

User workload is based on Hart and Staveland's NASA Task Load Index (TLX). NASA TLX is a short questionnaire based on 6 questions to assess perceived workload related to performing a task. The questionnaire can be found in Appendix A:

Each of these questions can be answered on a scale from 'Very Low' to 'Very High', with 'Performance' as an exception which uses a scale from 'Perfect' to 'Failure'. In all cases, a lower overall score signifies less workload and greater overall preference to this technique. See Figure 24 for NASA TLX results.

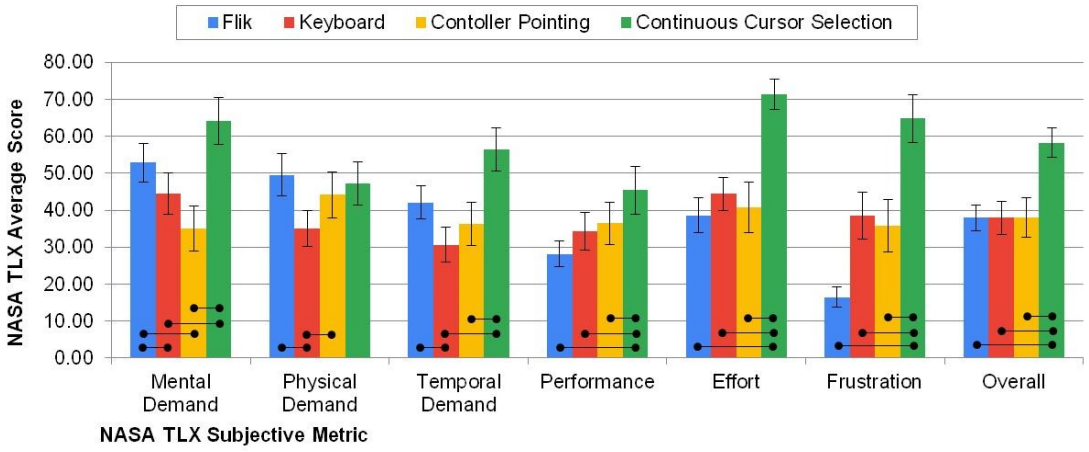


Figure 24 Average results of NASA TLX questionnaire with standard error bars. Lower scores represent more positive results. Black bars depict Friedman test results showing significance between each text entry technique.

Friedman post hoc pairwise comparison using Conover's F test results are represented on Figure 38 by horizontal bars between each text entry technique.

Exit Questionnaire

The exit questionnaire consists of two questions and was filled out at the end of the study when participants have used all techniques. The first question is to rank each text entry technique in terms of their favourite to use, with 1 being their overall favourite and 4 being their least favourite. Table 1 summarizes these rankings.

Table 1 Text entry technique rankings based showing number of participants who ranked each technique as 1st, 2nd, 3rd, and 4th. Each column adds up to 24 based on the 24 participants.

Text entry Technique	Rank			
	1	2	3	4
FLIK	10	12	2	0
Keyboard	8	4	11	1
Controller Pointing	5	8	8	3
Continuous Cursor Selection	1	0	3	20

4.6 Discussion

The physical keyboard condition did best overall, however, this was expected. After that, FLIK did about as well as controller pointing, with an average entry speed of 23 WPM on the third block, where the top entry speed achieved on any block was 28 WPM. When it comes to user preference and workload, FLIK scored worse on mental and physical demand. This is reasonable since this technique is analogous to something like playing the drums, where focus is placed on managing the accurate movement of hands as well as the physical effort of moving both hands continuously. Despite this, FLIK makes up for this in performance and low frustration, which are important factors when it comes to user preference. Due to the fact that these text entry techniques are, for the time being, meant to be used in short text entry tasks, the physical demand aspect becomes less of an issue since users would not be writing long form text, hence, reducing the exposure to fatigue. Ranking first overall as participants' favorite text entry technique, it serves as proof that standard VR text entry techniques such as Controller Pointing, and continuous cursor can be not only, outperformed but could be replaced by more favorable alternatives.

Physical keyboard demonstrated high results in terms of text entry speed, with a top entry speed score of 49.6 WPM on the third block. While there were some cases of low scores among some of the participants, with one participant scoring the lowest of 15.7 WPM, this is far offset with the proficiency of participants using regular keyboards, with most being able to type without looking down at the keyboard. Cases where scores were low could be attributed to poor tracking, appearing as the virtual hands not aligning perfectly with real world hands, these cases were rare and the offset of the virtual hands

with the real-world hands was not extreme. While the physical keyboard offered performance double that of the other techniques, we again note that the physical keyboard is, in many ways, a sub-optimal choice as a VR text entry technique due to several issues unrelated to performance. For example, the extra hardware (specific Logitech keyboard, attachment piece for the keyboard, and external HTC Vive sensors) and software required is not something that is readily available with out of the box VR headsets. This decreases the adoption rate of such a technique, at least until these components become an integrated part of a HMD package. The most negative factor is that this technique is not very portable. VR users tend to be standing and walking, rotating in place, and generally changing their position more often than not. Carrying a full-sized keyboard does not lend itself very well for this type of activity since it would need to be stationary.

Controller Pointing is currently the most commonly used technique in commercial VR products, and was shown to offer good performance as well. However, with text entry speed results coming in below FLIK (although not significantly worse), and being overall ranked second as participants' favorite choice of text entry technique, there is room for improvement when it comes to the standard method of entering text in VR. The continuous cursor technique, which is also commonly available in commercial products, was by far the least preferred and worst performing technique, further demonstrating that the industry VR software developers have gravitated towards several sub-optimal techniques.

As per **H1**, results support that physical keyboard performs the best, and that continuous cursor selection performs the worst, however, there was no significant difference between FLIK and controller pointing. **H2,** on the other hand, revealed that

continuous cursor selection was not the technique that yields the least fatigue due to high physical, mental, and temporal demand. Physical keyboard did show lower fatigue, followed by controller pointing, and lastly FLIK.

Taking these results into consideration, the decision of continuing to validate the text entry testbed with a second study was made. During this second study, only FLIK and Controller Pointing techniques were used in combination with BigKey and Word Disambiguation.

4.6.1 Participant Comments

Below are some of the most notable comments made by participants, which further support the points made above.

- I found the physical keyboard cool, but I don't see it as a feasible solution to VR due to the extra hardware.
- FLIK was very fun however, more tiring than the rest. Keyboard was cool, but only looked at the virtual keyboard a few times, and it was disorienting.
- FLIK was good and fast. Keyboard was nice, but the tracking could be better. Touchpad was nice and low effort, but it was easy to make mistakes.
- FLIK is good and different, however, keys are too close, so it causes unexpected typos. Keyboard tracking is not accurate.
- Touchpad is annoying when typing.
- FLIK feels good to type with, very flexible and fun, it does sometimes feel too sensitive making me have to make some corrections.

Chapter 5: User Study 2

This chapter presents the second user study which compares performance between Controller Pointing and FLIK with the addition of text entry aids, BigKey and word disambiguation. To further demonstrate the flexibility of the text entry testbed, the Oculus Rift CV1 was used in this study instead of the HTC Vive which have different setup requirements and APIs. This study was approved under CUREB-B file number 109263.

5.1 Hypotheses

The following hypotheses for this second user study are decided based on pre-testing of the text entry techniques used:

H1: FLIK will outperform Controller Pointing without aids.

H2: FLIK + BigKey will show better performance as compared to controller pointing + BigKey due to more direct interaction.

H3: Despite greater "gorilla arm" effect and fatigue that FLIK might cause, user preference will favour FLIK.

5.2 Participants

This second study consisted of 24 participants, 13 male, 11 female, and aged 18 – 55 (*SD* = 9.01). Recruitment was done through flyers posted around campus as well as word of mouth. In this study, all participants had little to no experience in virtual reality systems. All participants stated that they are comfortable with basic or advanced computer use. There are no repeated participants from the 1st study.

5.3 Apparatus

5.3.1 Hardware

The experiment was conducted on a VR-ready laptop with an Intel Core i7-6700HQ quad-core processor, an Nvidia Geforce GTX 1070 GPU, and 16GB of RAM, running the latest build of Microsoft Windows 10. We used the Oculus Rift CV1 HMD with two Oculus Touch 6DOF controllers. The Oculus Rift CV1 provides a 1080 x 1200-pixel resolution per eye, a 90 HZ refresh rate, and 110 degrees of field of view.

Participants were seated in a spacious study room, positioned far away enough from obstacles to avoid accidentally hitting anything. Participants used the Oculus Touch controllers as the main input device, pressing the triggers to confirm selections for Controller Pointing and used the 3D tracked capabilities of the controllers for both text entry techniques.

5.3.2 Software

Similar to the first study (Chapter 4), this study employed the text entry testbed described in Chapter 3. The text entry techniques were implemented and integrated with the text entry testbed for the previous study, hence, only the BigKey and word disambiguation conditions were added on top of the text entry techniques before testing the experiment software. All necessary data is logged by the system into a unique log file to avoid overwriting previous data.

5.3.3 Procedure

Upon arrival, participants were asked to read the consent form provided in Appendix B. They were told that if at any time they felt uncomfortable or nauseous that they were free to stop the experiment without finishing. Participants were then asked to try the HMD, followed by presenting them with their first text entry technique with the first condition (no aids, BigKey, or word disambiguation), where they were free to try it out and practice for around two minutes. Once they felt comfortable, the study began with that text entry technique.

Participants performed a transcription task for three blocks. Each block consisted of eight phrases to be transcribed; this was done for all six conditions. Once a block was finished, participants took a short break then continued with the next block. When all three blocks were finished, the next text entry technique and aid was chosen and the process repeated from trying out and practicing the technique for two minutes. In between each technique/condition, participants were also asked to fill out the NASA-TLX questionnaire provided in Appendix A. During the breaks, participants were encouraged to provide any feedback about the current technique. Finally, just before the start of each block, participants were instructed to transcribe the phrases presented as quickly and accurately as possible.

For this study, error correction was disabled, to reduce variability between participants. This decision was based on behaviors observed in user study 1, where some participants took much longer in order to correct the phrases, whereas other participants ignored errors and finished with less time.

At the end of the study, participants were asked to fill out one last questionnaire

asking them to rank each technique as well as provide any additional feedback they might have.

5.4 Design

The experiment employed a within-subjects design, with three independent variables:

- Text entry technique: FLIK, controller pointing
- Aid: no aid, BigKey, Word Disambiguation
- Block: 1, 2, and 3

The text entry techniques and aids were counterbalanced using a Latin square. With all 24 participants, this resulted in:

*3 blocks * 8 phrases * 2 text entry techniques * 3 aids * 24 participants = 3456 phrases transcribed.*

There were three dependent variables (entry speed, error rate, and NASA-TLX scores). User experience was the overall results from the NASA-TLX and exit questionnaire found in Appendix A. The NASA TLX remained the same, and for the exit questionnaire, two questions are asked. The first was to choose their most favored technique/aid combination in terms of which one the participant liked the best, and the second was if there are any comments, feedback, opinions about the study, or the techniques used.

5.5 Results

5.5.1 Task Performance

Starting with entry speed (WPM), the slowest condition was Controller Pointing with no aid, at 18.12 WPM. The fastest condition, at 27.8 WPM was FLIK+BigKey. See Figure 25 for full results of study 2 entry speed.

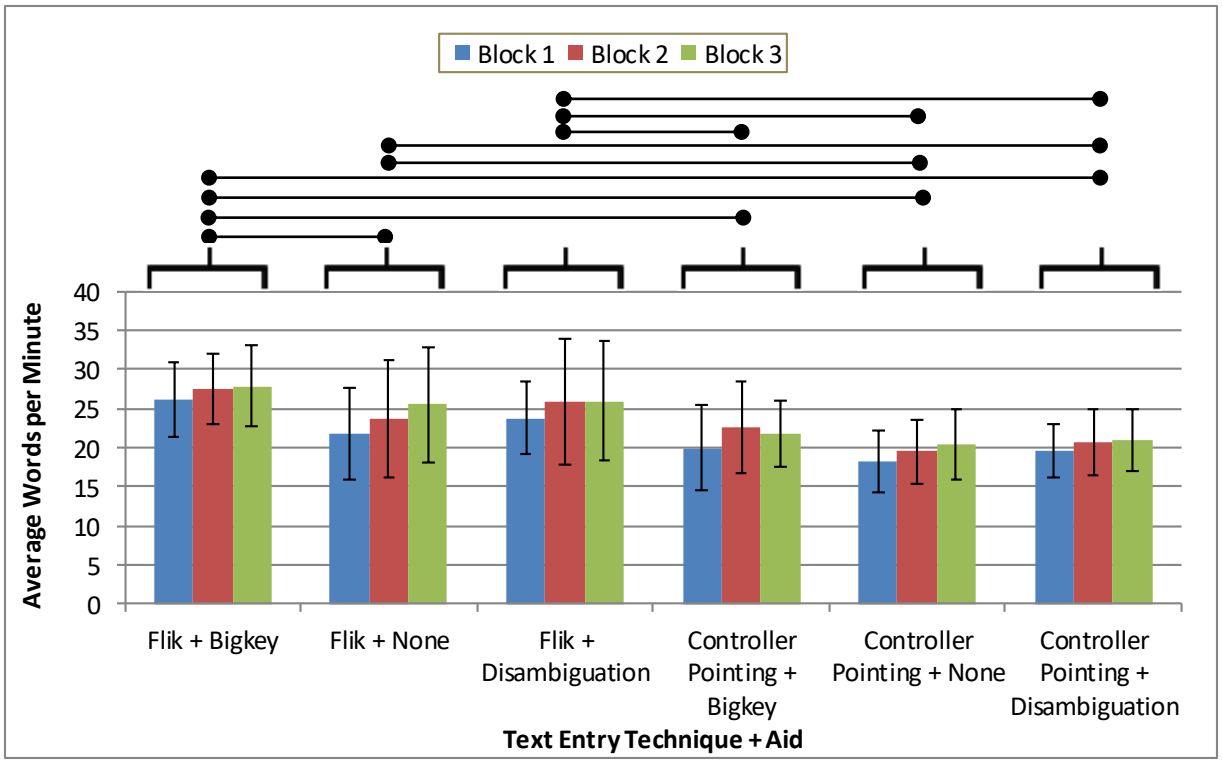


Figure 25 Study 2 Average WPM Comparison Table. Error bars show standard deviation. Black horizontal bars show pairwise significant differences between some text entry techniques.

Repeated-measures analysis of variance revealed that the effect of text entry technique on entry speed was statistically significant ($F_{1,\ 23}$ = 67.705, p < .0001). The effect of aid on entry speed was statistically significant ($F_{2,\ 46}$ = 41.098, p < .0001). The effect of block on entry speed was statistically significant ($F_{2,\ 46}$ = 107.446, p < .0001). The text entry technique-block interaction effect was statistically significant ($F_{2,\ 46}$ =

5.338, $p < .01$). The aid-block interaction effect was statistically significant ($F_{4, 92} = 6.179$, $p < .0005$).

Post Hoc Bonferroni-Dunn test at the $p < .05$ level results are represented as horizontal bars showing individual significant results for each text entry technique + aid combination.

Error rates ranged from 0.73% error for Controller Pointing + BigKey to 4.11% for FLIK + word disambiguation. See Figure 26.

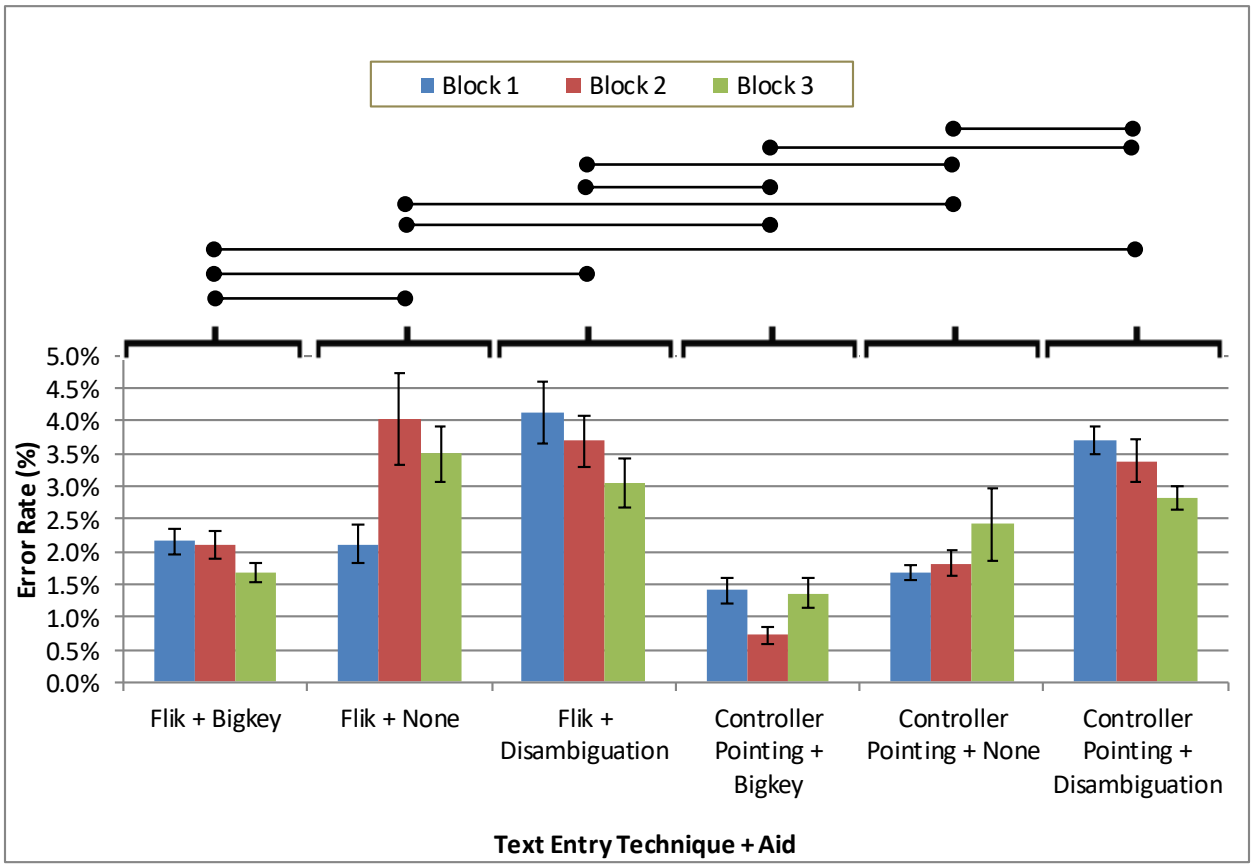


Figure 26 Study 2 Average error rate (%) comparison table. Black horizontal bars show pairwise significant differences between some text entry techniques.

Repeated-measures analysis of variance revealed that the effect of text entry technique on error rate was statistically significant ($F_{1, 23} = 6.456$, $p < .05$). The effect of

aid on error rate was statistically significant ($F_{2,\ 46} = 23.412$, $p < .0001$). The text entry technique-block interaction effect was statistically significant ($F_{2,\ 46} = 13.855$, $p < .0001$). The aid-block interaction effect was statistically significant ($F_{4,\ 92} = 12.067$, $p < .0001$).

A Post Hoc Bonferroni-Dunn test at the $p < .05$ level showed individual significant results on error rate for each of the text entry technique + aid combination and are represented by horizontal bars.

5.5.2 User Preference

Workload

Similar to the first user study, user workload is based on the NASA TLX questionnaire, where six questions are asked related to perceived mental demand, physical demand, temporal demand, performance, effort, and frustration. Each of these questions can be answered on a scale from 'Very Low' to 'Very High', with 'Performance' as an exception which uses a scale from 'Perfect' to 'Failure'. A lower overall score signifies less workload and overall preference towards a technique.

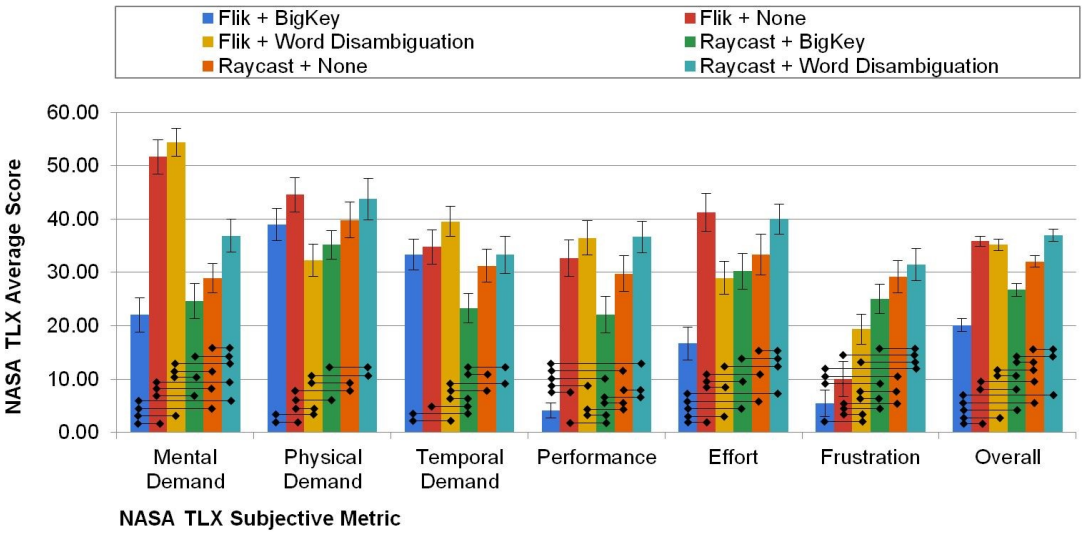


Figure 27 Averages of Study 2 NASA TLX results

As seen in Figure 27, FLIK + None and Controller Pointing + None scored similarly to corresponding conditions in study 1. Friedman post hoc pairwise comparison using Conover's F test results are represented by horizontal bars between each text entry technique; this shows that the BigKey condition had a positive impact overall for both text entry techniques, affecting some categories significantly, such as mental demand, performance, effort, and frustration. It is also interesting to note that Word Disambiguation is overall rated negatively, on average receiving higher scores than the rest, this could be attributed to a distraction factor in the study, for example, having to keep track of the suggested words while typing at the same time.

Exit Questionnaire

For this second study, the exit questionnaire's first question was modified to choose their favorite technique/condition pair alone, in this way, showing participants' favorite technique/condition pair would make it clear of the technique/condition rankings. See Figure 28 for the sum total of the 24 participant's favorite overall text entry technique-aid combination.

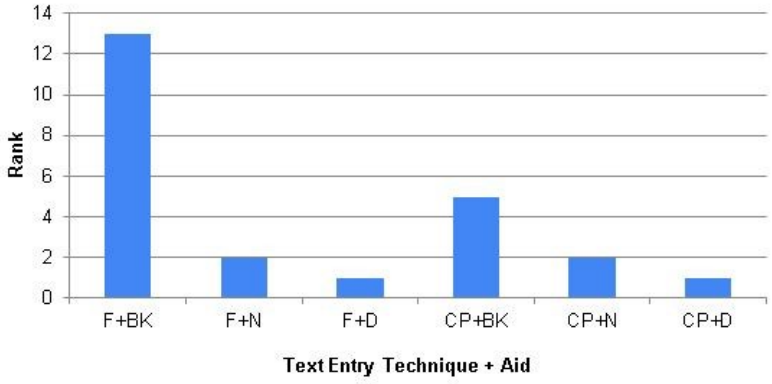


Figure 28 Study 2 Text entry Technique/Aid Pair Rankings. *F* = FLIK, *CP* = Controller Pointing, *BK* = BigKey, *N* = None, *D* = Disambiguation.

From this chart, we see how FLIK + BigKey pair was the clear favorite among

participants. Another interesting point to notice is how BigKey is ranked first for both techniques, followed by no condition, and ranked last is word disambiguation.

5.6 Discussion

Using BigKey with FLIK improved both performance and satisfaction, yielding the highest entry speed at 27.9 WPM. This suggests that there is potential in alternative VR text entry techniques, and how they can be improved with different tweaks and aids, even offering better performance than common commercial approaches, such as ray casting. Word disambiguation demonstrated to have below average results and actually caused some frustration and higher mental demand than other text entry technique + aid combinations. Even with the Controller Pointing technique, BigKey demonstrated the best overall performance for the technique, and word disambiguation coming in second while hindering user preference in a big way.

While keeping in mind the text entry error rates and entry speed (word disambiguation scoring relatively poor in these) an assumption can be made that the task of choosing and selecting fully suggested words from the list of suggestions increases mental workload, particularly when taking into consideration that this type of VR typing task is already something that might be new to most people. On the other hand, BigKey is non-intrusive and works with the participant in the task of purely typing character by character, making it easier and more obvious to type the next character in the sequence, achieving a greater flow in the typing task as well as user satisfaction. The higher performance of BigKey can be attributed to Fitts' law, which states that the time required to rapidly move to a target area is a function of the ratio between the distance to the target and the width of the target.

Unlike our first user study, FLIK + none shows significant results against controller pointing + none. This is notable because these two combinations are the same as FLIK and controller pointing in the first study. The change in between the two studies could be attributed to the HMD used, and this could potentially be a factor in the performance of these techniques. When using the HTC Vive, the controllers are larger and heavier, which could benefit controller pointing and hinder FLIK. With more weight and larger controllers, moving your hands fluidly in 3D space becomes tiresome as is the case for FLIK. In contrast, controller pointing requires sensitive rotational movements which are better achieved with the weight and size of the controllers. When using the Oculus Rift, the controllers are smaller and lighter. Performing continuous fluid motions in 3D space becomes easier when the controllers are lighter in weight and smaller in size. For controller pointing, the control of rotational movements becomes sensitive and thus harder to select targets at a distance.

NASA TLX scores show that FLIK + BigKey are the top favorite technique + aid combination among participants with overall significant results. However, FLIK without BigKey shows worse overall scores than controller pointing + BigKey, with poor word disambiguation scores as well. This shows that existing techniques can be improved with carefully selected modifiers to aid in performance and user satisfaction.

Below some of the most notable comments made by participants are presented, which further support the points made above.

- "*Word suggestion is nice and all, but it is very distracting for the FLIK technique, it is also distracting for Controller Pointing but to a lesser extent. It simply affects my concentration while performing the task at hand*."
- "*Overall, I like both techniques, but by far BigKey is something I would love to have in something like my iPhone keyboard*."
- "*At first, I thought FLIK had an issue with the spacing between the letters, but*

after trying BigKey I was very pleasantly surprised at how this became a non-issue very quickly."

- "*I would love to try FLIK in a real-world situation, like for example, playing a game and having to type a message to a friend or type some text to solve a puzzle or something like that*."

Given these results, both **H1** and **H2** are proven to be correct which could be true for a number of factors which benefit FLIK, such as the more direct interaction of FLIK which is further supported by Fitt's Law [15]. **H3** is supported from results as well as participant comments, showing that FLIK is preferred as depicted by overall TLX results.

Chapter 6: Conclusion

6.1 Summary

In this thesis, the focus was to propose, implement and test the Text Entry Testbed, a general-purpose text entry testing software with a focus on virtual reality text entry experiments. This tool was designed to be flexible and customizable, and to be able to use any text entry technique, not only text entry techniques for VR. The text entry testbed was made with the 3D game engine software Unity, which was chosen for its 3D capabilities and, for the support of 2D experiences. By providing researchers with simple plugin functions that they can add to their text entry techniques, the tool takes care of many other features, from logging data, to study structure and flow. Furthermore, the tool is tested by conducting two studies summarized below.

The first study regarded an implementation of two common VR text entry techniques used in commercial VR products; Controller Pointing and continuous cursor selection. Besides the two common techniques, two other text entry techniques were also implemented: physical keyboard-based typing using Logitech's Bridge technology to allow users to visualize the keyboard and their hands within VR, and the novel Fluid Intersection Keyboard (FLIK). Results of the first study indicated that the physical keyboard is by far the best technique in terms of text entry speed. FLIK performed similar to the controller pointing technique, which is currently the industry standard for entering text into VR systems. Despite the similar entry speed results between the two, FLIK was favored by participants, showing lower frustration levels and overall positive results from NASA-TLX scores.

The second study featured a subset of the text entry techniques from the first study; Controller Pointing, and FLIK. Across both techniques, we evaluated two aids, BigKey and Word Disambiguation, in addition to a third "default" implementation of each text entry technique without either aid. The conditions without aids provided comparison points to the first study. In total, this meant six different conditions were researched; FLIK + none, FLIK + BigKey, FLIK + Word Disambiguation, Controller Pointing + None, Controller Pointing + Bigkey, and Controller Pointing + Word Disambiguation. Results showed that using Bigkey as an aid improved entry speed of both text entry techniques, with significant results in favor of FLIK + Bigkey. This is notable due to the fact that the FLIK as compared to Controller Pointing techniques from the first study did not show significant results.

6.2 Discussion

Further comparing Controller Pointing in both studies as well as Continuous Cursor Selection in the first user study with results obtained by Speicher et al. [27], similar results can be seen for both techniques. The thesis showed that text entry speed for Controller Pointing resulted in higher entry speed than Speicher. However, the contrast between Controller Pointing as compared to Continuous Cursor Selection text entry speed is reflected in both this thesis and the research of Speicher. A similar paper by Boletsis and Kongsvik [3] further validates our results by implementing their version of Controller Pointing, Continuous Cursor Selection, and CutieKeys, which is what FLIK is based on. As can be seen in Table 2, many results are similar on all these techniques.

Grubert et al. [9] reported an entry speed of 38.7 WPM text entry speed on their VideoHand technique, which is what the keyboard technique with video pass-through in

this thesis is based on. Table 2 presents that the system demonstrated different results, the standard deviation for both papers is large and presents similar results overall.

Table 2 - Text Entry Speed (WPM) comparison to similar studies. *Used CutieKeys Technique, which is the most similar to Flik.

	Results from our Studies	Speicher [27]	Boletsis [3]	Grubert [9]	Knierim [12]
Controller Pointing	19.13	15.44	16.65	-	-
Continuous Cursor Selection	13.9	8.35	10.17	-	-
FLIK	21.49	-	*21.01	-	-
Physical Keyboard	45.64	-	-	38.7	40

In terms of subjective data, BigKey had a big impact on user preference, which made this aid significantly preferable. On the other hand, it was surprising to see that participant's did not find Word Disambiguation helpful; these subjective results were confirmed by the performance results, as Word Disambiguation did not provide significantly better performance results than the null condition.

A different VR headset was used in each user study to further demonstrate the flexibility of the text entry testbed. Integration of different VR headsets is straight-forward within the Unity environment, however, setup and configuration does differ and thus we tested for ease of use with different VR headsets. The testbed can also be made to operate as a general purpose (i.e., non-VR) text entry testbed. For example, consider removing the VR component of the physical keyboard technique in the first user study. The technique would then work as a standalone non-VR text entry technique. This emphasizes that this tool is not just for VR experiments, but also serves as a general text entry experiment tool.

6.3 Limitations

6.3.1 Text Entry Testbed

The text entry testbed is designed to allow for quick and simple setup of VR text entry experiments; this can come with some limitations due to the rapid evolution of VR technology regarding APIs and development configuration. It often occurs that a new VR headset is announced, moreover, new APIs and development tools for each headset. Integrating these into Unity is usually simple and comes with good documentation. However, there are many different ways that these can be implemented. Due to all these options, it is not simple to release the text entry testbed as a ready to use VR text entry testing tool with all the necessary libraries or plugins needed to support any VR headset. Researchers would have to make sure that any new headset that is not already part of the text entry testbed is set up correctly with Unity to be compatible with the tool.

It is also important to note that as of now, the testbed must be used with Unity. This means that all project files come in the form of a Unity project. This is important since it would require some experience with the Unity engine to implement the desired text entry technique into the software. It is included in this thesis since future work will focus efforts on implementing a standalone version of the text entry testbed with simple techniques to integrate any text entry technique. In its current form, it may not be completely accessible due to the experience required with the engine; however, this makes the tool highly customizable to support any kind of testing environment. Having the tool as open source, researchers and developers are able to contribute changes and updates to expand and refine the functionality of the text entry testbed.

VR experiences differ largely between one another, they could require users to be

moving, walking, standing, or sitting down, which makes it difficult to provide specifications for external validity and generality. For our studies, participants were sitting down, which supports experiences where users are also sitting down. This decision was made due to the fact that the physical keyboard condition requires users to be sitting down, thus, to maintain consistency, other text entry techniques were also tested sitting down.

The use of the QWERTY keyboard layout is also adopted in our studies for being the most widely used keyboard layout in real world scenarios; however, using the text entry testbed the keyboard layout can be changed to other layouts if needed.

6.3.2 Studies

A limiting factor of our studies comes down to the tracking technology used for the physical keyboard condition in the first user study. While efforts were made to keep light interference, keyboard position, and head position constant, constant calibration of the system was necessary to maintain tracking accuracy. We used the VideoHand version of the physical keyboard as proposed by Grubert et al. [9] in our studies. While this technique performed best overall in their studies, it is also reported that the four hand representations tested in their work are just a small set of possible conditions in a large design space, in which more could be proposed.

For both the first and second study, FLIK was developed, which is originally based on CutieKeys [31], and tested among different text entry techniques. CutieKeys is not as standardized in the industry like Controller Pointing or Continuous Cursor Selection. However, it has gained popularity among developers. For the thesis, the

inclusion of the CutieKeys text entry technique could have resulted in interesting results when compared with FLIK.

The Mackenzie phrase set used works well when it comes to development of the predictive algorithms used by BigKey and word disambiguation, however, this poses an issue in that this phrase set is clearly limited to a set of words, making the results of the predictive algorithms biased to better performance as opposed to using the entire corpus of a given language. The phrase set used did not change in any of our studies, so while results might differ from using a different phrase set or corpus, the comparative analysis between text entry techniques and aids in this thesis maintains its validity.

6.4 Future Work

For future iterations of the software, a main focus is to dedicate development efforts to allow researchers to implement any text entry techniques in such a way that they can just start the text entry test-bed, enter the name of the technique that they are using, and have text automatically entered into any text entry area from their text entry technique. This would require some education to researchers in the area of developing their text entry techniques to work on operating system methods that allow for typical keyboards to enter text onto the machine.

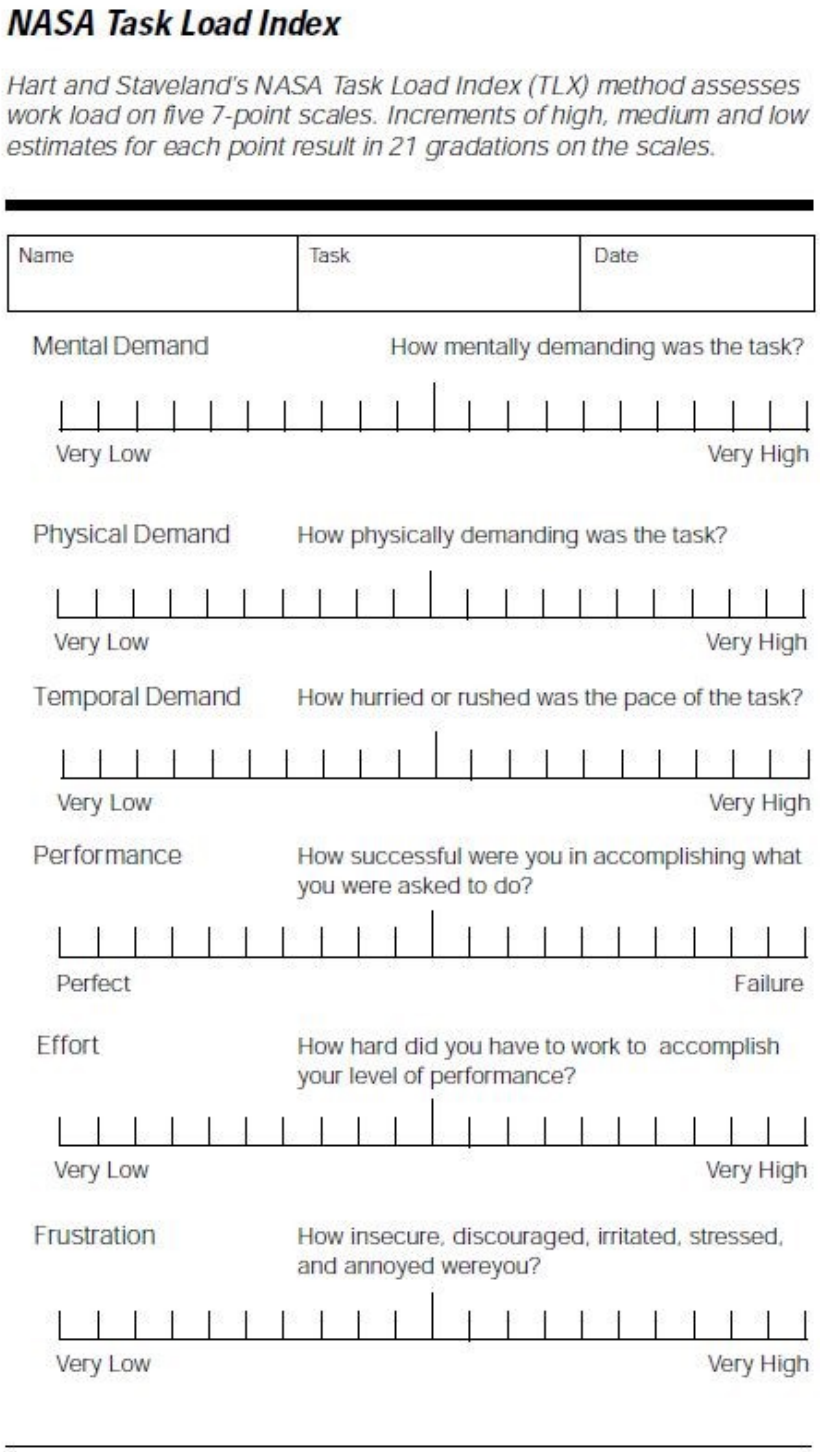
Other future work could explore different kinds of text entry experiments using this software. From plain desktop text entry techniques to virtual or augmented reality techniques, as well as other types of human-computer interactions other than these such as wearables (smart watches, smart glasses…), brain-computer technologies such as EEG's like EMOTIV devices [5]. Conducting studies comparing different types of HMDs and controllers is also an important study to perform since this could potentially

be a significant factor when it comes to text entry in VR.

Providing this tool as a free open source project would allow future researchers and developers to contribute their work and ideas to expand the functionality of the text entry test-bed and ultimately create a consistent, fully formed version of the tool to conduct text entry experiments.

Appendices

Appendix A - Questionnaires



NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Name	Task	Date

Mental Demand — How mentally demanding was the task?

Very Low — Very High

Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?

Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?

Very Low — Very High

Study 1 Exit Questionnaire

Exit Questionnaire

Please rank each text entry technique from 1 to 4 (1 – best, 4 – worst) in terms of which one was your overall favourite. Write your answer on the boxes to the right.

- Controller Pointing ---------- ☐
- Continuous Cursor Selection (Touchpad) ---------- ☐
- Real Keyboard ---------- ☐
- Flik ---------- ☐

Do you have any positive, negative, or general comments about any of the text entry techniques used or about the overall study?

Study 2 Exit Questionnaire

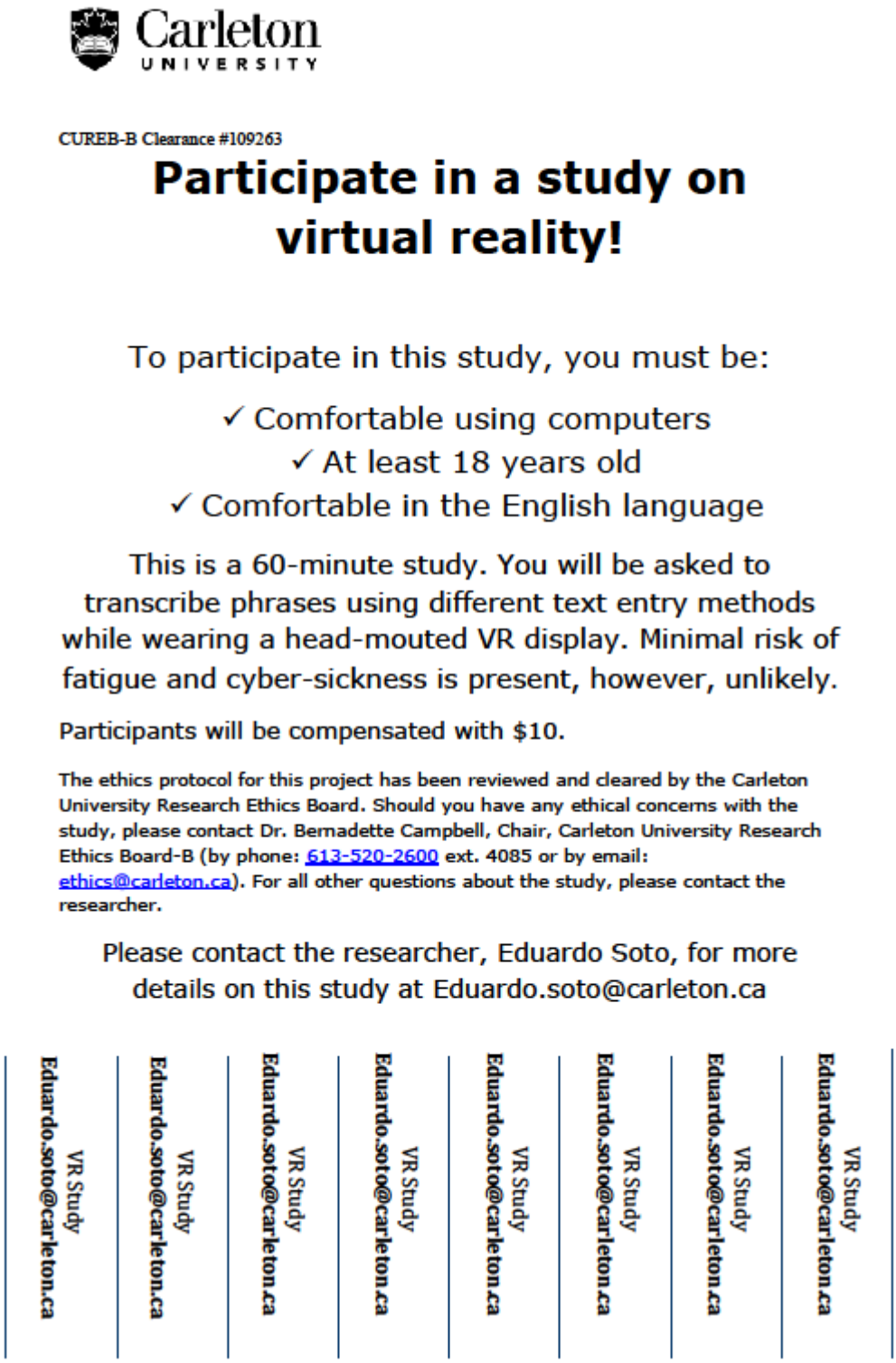
Exit Questionnaire

Please choose your favourite text entry technique + aid pair by drawing a circle around your selection.

Flik + BigKey	Flik+None	Flik+Word Suggestion
Controller Pointing + BigKey	Controller Pointing + None	Controller Pointing + Word Suggestion

Do you have any positive, negative, or general comments about any of the text entry techniques used or about the overall study?

Appendix B - Study Recruitment and Consent Forms



Online Invitation

To be posted on Carleton Research Participants Facebook page:

Volunteers needed for virtual reality study

We are looking for volunteers for a study on virtual reality sickness. Participants will receive $10. The study takes place at Carleton University.

This project is on text entry in Virtual Reality systems. The study aims to determine how well current text entry systems used in VR perform and investigate potential improvements to these systems or overall replacement with alternate text entry methods. Minimal risk of fatigue and cyber-sickness is present, however, unlikely.

To be eligible, you must be English-speaking, comfortable using computers to type and at least 18 years of age.

The study will take place on campus and should not take approximately 60 minutes to complete.

If you are interested, please email Eduardo Soto at eduardo.soto@carleton.ca for more details on participating.

The ethics protocol for this research has been reviewed and approved by the Carleton University Research Ethics Board B. Should you have any ethical concerns with the study, please contact Dr. Bernadette Campbell, Chair, Carleton University Research Ethics Board-B (by phone: 613-520-2600 ext. 4085 or by email: ethics@carleton.ca). For all other questions about the study, please contact the researcher.

CUREB-B Clearance #109263

Carleton UNIVERSITY
Canada's Capital University

Consent Form

Title: Evaluating Improvements on Current Virtual Reality Text Entry Methods

Date of ethics clearance: To be determined by the REB (as indicated on the clearance form)
Ethics Clearance for the Collection of Data Expires: To be determined by the REB (as indicated on the clearance form)

I _____, choose to participate in a study on Virtual Reality text entry systems. I acknowledge that this study aims to assess the strength and the weakness of current text entry systems while immersed in a Virtual Reality environment. **The researcher for this study is Masters' students Eduardo Soto.** He is working under the supervision of Dr. Robert Teather **in The School of Information Technology.**

This study will take approximately one hour in total. It involves 4 trials using different text entry methods (10 – 15 minute VR exposure with a short break between trials). With your consent, we will proceed with the experiment. Data gathered during the experiment includes the speed of text entry for each phrase transcribed, and sequence of characters entered.

Minimal risk of fatigue and cyber-sickness is present, however, unlikely. If you feel uncomfortable or nauseous you have the right to end your participation in the study at any time during the session, for any reason. Simply tell the researcher that you want to end the session. If you withdraw from the study, all information you have provided will be immediately destroyed. We just mention that one of the participants cannot finish the test.

As a token of appreciation, you will receive a $10. This is yours to keep, even if you withdraw from the study.

All responses will be kept anonymous. All research data and notes will be kept on a password protected computer of the researchers. Research data will only be accessible by the researchers and the research supervisor until the publishing date of this research where cumulative results will be made public in the publication of this work.

This document has been printed on both sides of a single sheet of paper.
Please retain a copy of this document for your records.

CUREB-B Clearance #109263

Carleton UNIVERSITY
Canada's Capital University

The ethics protocol for this project was reviewed by the Carleton University Research Ethics Board, which provided clearance to carry out the research. Should you have any ethical concerns with the study, please contact Dr. Bernadette Campbell, Chair, Carleton University Research Ethics Board-B (by phone: 613-520-2600 ext. 4085 or by email: ethics@carleton.ca). For all other questions about the study, please contact the researcher.

Researchers contact information:

Eduardo Soto
School of Computer Science
Carleton University
eduardosoto@cmail.carleton.ca

Supervisor contact information:
Dr. Robert Teather
School of Information Technology
Carleton University
613-520-2600x4176
rob.teather@carleton.ca

_____	_____
Signature of participant	Date
_____	_____
Signature of a researcher	Date

This document has been printed on both sides of a single sheet of paper.
Please retain a copy of this document for your records.

CUREB-B Clearance #109263

Appendix C - CUREB-B Protocol Form

ROMEO - Researcher Portal

Carleton UNIVERSITY
Canada's Capital University

CUREB-B Protocol Form

Project Info.

File No: 109263
Project Title: Evaluating Improvements on Current VR Text Entry Methods [Eduardo Soto]
Principal Investigator: Mr. Eduardo Soto (Faculty of Engineering and Design\Information Technology (School of))
Start Date: 2018/08/01
End Date: 2019/04/30
Keywords: Digital Humanities and Interactive Technologies,Information Technology,Virtual Environments

Project Team Info.

Principal Investigator

Prefix: Mr.
Last Name: Soto
First Name: Eduardo
Affiliation: Faculty of Engineering and Design\Information Technology (School of)
Rank: Student - Masters
Email: eduardosoto@cmail.carleton.ca
Phone1: 2899376344
Phone2:
Fax:
Primary Address: 860 Canterbury Ave., Apt 204Ottawa, ON, K1G3B2
Institution: Carleton University
Country: Canada
Comments:

Other Project Team Members

Prefix	Last Name	First Name	Affiliation	Role In Project	Email

Dr.	Teather	Robert	Faculty of Engineering and Design\Information Technology (School of)	Research Supervisor	rob.teather@carleton.ca

Attachments

Doc / Agreement	Version Date	File Name	Description
Annual Status Report	2019/09/27	Teather 109263 asr renewal 2019-09-27.docx	asr renewal
Clearance Certificate	2018/10/31	Teather 109263 Clearance 2018-10-31.pdf	clearance
Clearance Certificate	2019/10/03	Teather 109263 clearance renewal 2019-10-03.pdf	clearance renewal
Protocol Form	2018/10/01	Teather 109263 Protocol and Appendices Binder 2018-10-01.pdf	N/A
Response Letter	2018/10/29	Sample CUREB response letter (Eduardo Soto) 2018-10-29.pdf	response letter
Revised Protocol	2018/10/22	Teather 109263 Revisions Binder 2018-10-29.pdf	revisions

References

[1] K. Al Faraj, M. Mojahid, and N. Vigouroux, "BigKey: A Virtual Keyboard for Mobile Devices," 2009.

[2] a. S. Arif and W. Stuerzlinger, "Analysis of text entry performance metrics," *Sci. Technol. Humanit. (TIC-STH), 2009 IEEE Toronto Int. Conf.*, pp. 100–105, 2009.

[3] C. Boletsis and S. Kongsvik, "Controller-based Text-input Techniques for Virtual Reality: An Empirical Comparison," *Int. J. Virtual Real.*, vol. 19, no. 3, pp. 2–15, 2019.

[4] W. Contributors, "The Sword of Damocles (Virtual Reality)," *Wikipedia, The Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/The_Sword_of_Damocles_(virtual_reality). [Accessed: 09-Jan-2020].

[5] EMOTIV, "EMOTIV." [Online]. Available: https://www.emotiv.com/?gclid=CjwKCAiAhc7yBRAdEiwAplGxX_rAdLfhY2VUpfE9LQJeVovnSZZRWLlQjIEDhP9QF_Wif5CrkAyc_xoC-lAQAvD_BwEhttps://www.emotiv.com/?gclid=CjwKCAiAhc7yBRAdEiwAplGxX_rAdLfhY2VUpfE9LQJeVovnSZZRWLlQjIEDhP9QF_Wif5CrkAyc_xoC-lAQAvD_BwE. [Accessed: 10-Feb-2020].

[6] S. S. Fels and G. E. Hinton, "Glove-TalkII - A neural-network interface which maps gestures to parallel formant speech synthesizer controls," *IEEE Trans. Neural Networks*, vol. 9, no. 1, pp. 205–212, 1998.

[7] D. R. Gentner, J. T. Gruding, S. Larochelle, D. A. Norman, and D. E. Rumelhart, "A Glossary of Terms Including a Classification of Typing Errors," in *Cognitive Aspects of Skilled Typewriting*, New York: Springer Verlag, 1983, pp. 39–43.

[8] J. Grubert, L. Witzani, E. Ofek, M. Pahud, M. Kranz, and P. O. Kristensson, "Text Entry in Immersive Head-Mounted Display-based Virtual Reality using Standard Keyboards," pp. 1–8, 2018.

[9] J. Grubert, L. Witzani, E. Ofek, M. Pahud, M. Kranz, and P. O. Kristensson, "Effects of Hand Representations for Typing in Virtual Reality," pp. 1–8, 2018.

[10] M. Jensen and N. R. Melzack, "SpiderWorld and SnowWorld," no. July, 2004.

[11] A. Kano, J. C. Read, A. Dix, and I. S. MacKenzie, "ExpECT: An Expanded Error Categorisation Method for Text Input," *Proc. HCI'07 Conf. People Comput. XXI*, vol. 1, p. 15, 2007.

[12] P. Knierim, V. Schwind, A. M. Feit, F. Nieuwenhuizen, and N. Henze, "Physical Keyboards in Virtual Reality: Analysis of Typing Performance and Effects of Avatar Hands," *Proc. 2018 CHI Conf. Hum. Factors Comput. Syst.*, pp. 1–9, 2018.

[13] A. J. Ko and J. O. Wobbrock, "Text Entry," *User Interface Software and Technology*. [Online]. Available: https://faculty.washington.edu/ajko/books/uist/text-entry.html. [Accessed: 05-Jan-2020].

[14] J. J. LaViola and D. Bowman, *3D User Interfaces*, 2nd ed. Addison-Wesley Professional, 2017.

[15] F. Law and I. S. Mackenzie, "Fitts' Law," vol. 1, pp. 349–370, 2018.

[16] I. S. Mackenzie, "Evaluation of Text Entry Techniques," *Text Entry Syst.*, pp. 75–101, 2007.

[17] I. S. MacKenzie and R. W. Soukoreff, "A character-level error analysis technique for evaluating text entry methods," *Proc. Second Nord. Conf. Human-computer Interact. - Nord. '02*, no. 1, p. 243, 2002.

[18] I. S. MacKenzie and R. W. Soukoreff, "Phrase sets for evaluating text entry techniques," *CHI '03 Ext. Abstr. Hum. factors Comput. Syst. - CHI '03*, p. 754, 2003.

[19] C. Maggioni, "Novel gestural input device for virtual reality," *1993 IEEE Annu. Virtual Real. Int. Symp.*, pp. 118–124, 1993.

[20] M. Pérez-Ramírez and N. J. Ontiveros-Hernández, "Virtual Reality as a Comprehensive Training Tool," 2009.

[21] I. Poupyrev, N. Tomokazu, and S. Weghorst, "Virtual Notepad: Handwriting in immersive VR," *Proc. - Virtual Real. Annu. Int. Symp.*, pp. 126–132, 1998.

[22] F. E. Sandnes and A. Aubert, "Bimanual text entry using game controllers: Relying on users' spatial familiarity with QWERTY," *Interact. Comput.*, vol. 19, no. 2, pp. 140–150, 2007.

[23] W. J. Shelstad, D. C. Smith, and B. S. Chaparro, "Gaming on the rift: How virtual reality affects game user satisfaction," in *Proceedings of the Human Factors and Ergonomics Society*, 2017, vol. 2017-October, pp. 2072–2076.

[24] R. W. Soukoreff and I. S. MacKenzie, "Metrics for text entry research- an evaluation of MSD and KSPC, and a new unified error metric," *Proc. Conf. Hum. factors Comput. Syst. - CHI '03*, vol. 5, pp. 113–120, 2003.

[25] R. W. Soukoreff and I. S. MacKenzie, "Recent developments in text-entry error rate measurement," *Ext. Abstr. 2004 Conf. Hum. factors Comput. Syst. - CHI '04*, p. 1425, 2004.

[26] R. W. Soukoreff and I. S. MacKenzie, "Measuring errors in text entry tasks," *CHI '01 Ext. Abstr. Hum. factors Comput. Syst. - CHI '01*, p. 319, 2001.

[27] M. Speicher, A. M. Feit, P. Ziegler, and A. Krüger, "Selection-based Text Entry in Virtual Reality," *Proc. 2018 CHI Conf. Hum. Factors Comput. Syst. - CHI '18*, no. April, pp. 1–13, 2018.

[28] V. Tucker, "Introducing the Logitech BRIDGE SDK," 2017. [Online]. Available: https://blog.vive.com/us/2017/11/02/introducing-the-logitech-bridge-sdk. [Accessed: 10-Oct-2019].

[29] J. Walker, S. Kuhl, and K. Vertanen, "Decoder-Assisted Typing using an HMD and a Physical Keyboard," *Chi '16*, no. May, 2016.

[30] J. Walker, B. Li, K. Vertanen, and S. Kuhl, "Efficient Typing on a Visually Occluded Physical Keyboard," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*, 2017.

[31] M. Weisel, "Cutie Keys." [Online]. Available: https://www.normalvr.com/blog/an-open-source-keyboard-to-make-your-own/. [Accessed: 19-Jan-2020].

[32] J. O. Wobbrock, "Measures of Text Entry Performance," *Text Entry Syst.*, pp. 47–74, 2007.

[33] J. O. Wobbrock and B. A. Myers, "Analyzing the input stream for character- level errors in unconstrained text entry evaluations," *ACM Trans. Comput. Interact.*, vol. 13, no. 4, pp. 458–489, 2006.