

# **Optical Hand Tracking in Virtual Reality to Teach American Sign Language**

Christian O'Brien BSc

Submitted in partial fulfilment for the degree of Master of Science in  
Serious Games and Virtual Reality

August 12, 2020

Supervised by Dr. Daniel Livingstone MSc BEng FHEA

School of Simulation and Visualisation

The Glasgow School of Art

## **Abstract**

Optical hand tracking technologies in Virtual Reality (VR) offer an exciting new method of human-computer interaction. Among the applications being explored, its use for education holds great promise. Gesture recognition systems are in development, but none have yet proven to support existing gesture collections such as American Sign Language (ASL). Fingerspelling is central to ASL and is often a challenge to ASL learners, so digital tools could assist them in the acquisition of this skill. This paper explores the use of a virtual reality serious game which utilizes optical hand tracking to teach the ASL alphabet. The question is posed whether the use of such a game will improve a player's ability to fingerspell.

The new "ASL Fingerspeller" game was developed for the Oculus Quest using state of the art software and hardware, the latest of which came to market in February 2020. The game utilizes a simple sign recognition system which defines a sign as finger positions and wrist orientation and is customizable to each player. The game challenges the player to fingerspell names and places using the ASL alphabet.

Evaluation evidence confirmed that the game was able to successfully teach the ASL alphabet to participants in the research and improved their fingerspelling ability over a short period of time. Online users and in-person research participants enjoyed playing the game and thought it to be a useful learning tool. Results from extended testing saw an average improvement in sign production rate of 60 percent within five uses of the game. Testing showed areas for the game's improvement including improving the capability to detect signs with crossed or hidden fingers. These results are illustrative yet with the caveat that since testing of the game was limited by Covid-19 related restrictions, sample sizes were statistically insignificant.

The research demonstrates the potential use of hand tracking for educational purposes. The technology is not yet capable of supporting full ASL vocabulary but can support most signs of the alphabet. The game may not promote development of fingerspelling skills in the same manner native

signers learn the skill, but it can be used as a starting point for new learners. Further research could explore the inclusion of more advanced gesture recognition systems, especially those that recognize dynamic gestures.

This research is some of the first conducted using the Oculus Quest's hand tracking technology, and its application towards education. The methods in this study could be used as an example for further research in the fields of hand tracking games, games about ASL education, and simple gesture recognition systems. As the technology develops, including becoming even more user-friendly and affordable, this project illustrates that the opportunities for new applications are extensive and exciting.

# Table of Contents

Abstract	i
Table of Contents	iii
List of Tables	vi
List of Figures	vii
Declaration of Originality	viii
Definitions and Abbreviations	ix
1 Introduction	1
1.1 Application	1
1.2 Hand Tracking Technology	1
1.3 Groups of Interest	2
1.4 Structure of this Dissertation	3
2 Literature Review	5
2.1 Digital Game Based Language Learning (DGBLL)	5
2.2 ASL Instruction and Fingerspelling Comprehension	9
2.3 Gesture recognition technology	11
2.4 Summary	12
3 Materials	14
3.1 Virtual Reality Headsets and Hand Tracking Solutions	14
3.1.1 Oculus Quest	14
3.1.2 Hand Tracking	14
3.1.3 Alternative Solutions	15
3.2 Development Tools	16
	iii

3.2.1	PC to VR Workflow	16
3.2.2	Game Engine	16
3.2.3	Hand Tracking Compatibility	17
3.2.4	Modelling and Animation	18
4	Methods	19
4.1	Planning	19
4.2	Sign Detection	20
4.2.1	Defining a “Sign”	20
4.2.2	Sign Saving and Detection	20
4.2.3	Dynamic Signs	21
4.2.4	Sign Detection Implementation	21
4.3	Sign Functionality	22
4.4	Gameplay	22
4.5	User Interface Design	23
4.6	Finishing Touches	25
5	Implementation Results	26
5.1	Menu	27
5.2	Sign Calibration	29
5.3	Practice Mode	30
5.4	Fingerspelling Challenge	31
5.5	Issues	33
6	Evaluation	35
6.1	Approaches	35

6.1.1	In-person Medium Scale Testing	35
6.1.2	Online Testers	35
6.1.3	Extended User Testing	36
6.2	Limitations	36
6.3	Feedback from Online Users	37
6.3.1	SideQuest	37
6.3.2	Reddit	37
6.3.3	Twitter	37
6.3.4	Online Survey	38
6.4	In-person testing	41
6.5	Conclusion	43
7	Conclusion	45
7.1	Discussion	45
7.1.1	Achievements	45
7.1.2	Limitations	46
7.1.3	ASL Fingerspeller	46
7.1.4	Other Uses	47
7.2	Conclusion	47
	References	49
	Referenced Software and Games	51
	List of Third-Party Materials Used	52
	Appendix A. User Evaluations Results	53
	Appendix B. Code	58

## **List of Tables**

Table 6.1: Resulting Statistics of a Play Session from a Native ASL User.....	38
Table 6.2: SUS Survey Responses .....	39
Table R.1: Third Party Materials Used.....	52
Table A.1: Open Ended User Responses from Online Survey.....	53
Table A.2: In-person User Testing Results .....	55

## **List of Graphs**

Graph 6.1: Production Rate Improvement Over Time .....	42
Graph 6.2: Average Ratio of Incorrect Letters Detected per Correct Letter Produced .....	43

## List of Figures

Figure 1.1: Oculus Quest VR Headset .....	2
Figure 2.1: The ASL Fingerspelling Alphabet .....	8
Figure 3.1: HTC Vive Headset with LMC Attached.....	15
Figure 3.2: Hand Tracking Under Occlusion .....	17
Figure 3.3: Oculus Integration Bone ID Legend .....	18
Figure 4.1: Games Design Document .....	19
Figure 4.2: Interaction Range .....	23
Figure 4.3: Implementation of a "Physical" Button in Early Stages of Development.....	24
Figure 4.4: Tap-to-Click System Developed by Oculus.....	24
Figure 5.1: Main Menu.....	26
Figure 5.2: Set Active Hand Menu.....	27
Figure 5.3: Player Setting the Right Hand as Active.....	28
Figure 5.4: Player Using The Main Menu.....	28
Figure 5.5: Sign Calibrator .....	29
Figure 5.6: Player Using The Sign Calibrator .....	30
Figure 5.7: Practice Mode .....	30
Figure 5.8: Player Using the Practice Mode.....	31
Figure 5.9: Fingerspelling Challenge .....	32
Figure 5.10: Player Playing the Fingerspelling Challenge .....	33
Figure B.1: Spellchecker Code.....	58
Figure B.2: Code Implementation of a "Sign" .....	59
Figure B.3: Overwrite Function used in Calibration .....	59
Figure B.4: Sign Detection/Recognition System.....	60



# Declaration of Originality

STUDENT ID No.: 19129939

## Own Work Declaration

You must complete this declaration (each box ticked to show that the condition has been met), signed and dated, and included with each piece of work submitted for assessment (practical and written). Please note work will not be accepted unless this form is attached.

As GSA have now obtained Turnitin plagiarism software, a cross section of student submissions will be evaluated.

---

Name: Christian O'Brien

Course/Programme: MSc Serious Games and Virtual Reality

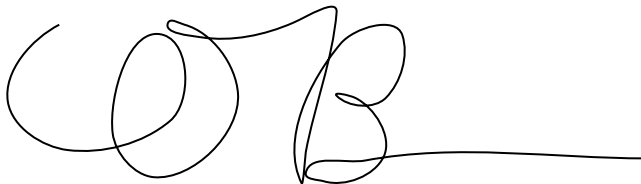
Title of Work: Optical Hand Tracking in Virtual Reality to Teach American Sign Language

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and added inverted commas to all quoted text (from books, journals, web, etc)
- Given the sources of all pictures, sound, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present, or lifted extracts from web pages without appropriate referencing
- Not sought or used the help of any external professional agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course Handbook

I understand that any false claim for this work will be penalised in accordance with the GSA regulations

Signature:

A handwritten signature in black ink, appearing to read 'C O'Brien', with a long horizontal line extending to the right.

Date: August 12, 2020

## Definitions and Abbreviations

<b>Term</b>	<b>Definition</b>
American Sign Language (ASL)	A non-verbal language used largely by the Deaf and Hard-of-Hearing community in North America. Vocabulary of ASL is made of hand gestures and facial expressions.
Application Programming Interface (API)	A software interface which gives the developer access to a set of functions
Augmented Reality (AR)	A system in which computer-generated imagery is overlaid on the user's view, which gives the impression of a digitally augmented space
Bone	A point on the hand tracked by the Oculus Quest which holds its own positional and rotational data
Degrees of Freedom (DoF)	The axes along which a VR headset will track motion. 6DoF includes the X, Y, and Z axes for translation (movement) (2 degrees per axis), as well as the X, Y and Z axes for rotation (turning)
Digital Game-based Language Learning (DGBLL)	Learning a language using digital games
Extended Reality	A term which encompasses technologies related to simulated experiences such as virtual reality and augmented reality
Fingerspelling	A process in ASL in which words are spelled out using the signs for the letters of the English alphabet. Fingerspelling is used for names and words for which there is no dedicated sign.
Game-based Learning	Learning that occurs through explicit instruction when playing a game
Game-enhanced Learning	Learning that occurs through implicit instruction when playing a game
Gesture Recognition	The technology by which human hand gestures are used as an interface to a computer system
Global Space	The position of an object relative to a fixed origin point
Hand Tracking	The technology of tracking the motion and rotation of hands for virtual re-creation.

Hands Interaction Toolkit (HIT)	A subsection of the Unity Integration API which allows for hand ray casting and physics interactions
Handshape	The positioning and orientation of the fingers when producing a sign in ASL. Tall handshapes refer to signs with fingers straight or unfolded. Short handshapes refer to signs with the fingers bent or folded
Head-mounted Display (HMD)	A headset which places screens in front of the user's eyes, used for virtual and augmented reality
Hidden Markov Model (HMM)	A neural network algorithm which is commonly used in speech recognition systems to find "hidden" states (Fok <i>et al.</i> , 2015)
<code>InverseTransformPoint</code>	A method of finding an objects local space relative to another object
k-Nearest Neighbour (kNN)	A machine learning algorithm used for comparison and sorting
Leap Motion Controller (LMC)	An optical hand tracking module that captures the movements of the users hands using near-infrared sensors. (Ultraleap, 2019)
Local Space	The position of an object relative to itself or another object
Multilayer Perceptron (MLP)	A feed forward artificial neural network (Mapari and Kharat, 2016)
Near-Infrared	A section of the electromagnetic spectrum which is invisible to the naked human eye
Oculus Quest (Quest)	A VR HMD released by Oculus in 2018. The Quest uses four cameras for spatial tracking and offers wireless optical hand tracking
Oculus Unity Integration	An API which allows for the development of Quest games using Unity
Ray Cast	A method of checking overlap or intersection. One object fires a ray. If the ray hits the other objects, the system recognizes the objects as overlapping.
Reddit	A social media site which hosts discussion boards separated by user interest groups
Second Language (L2)	An individual's learned but non-native language
SideQuest	An unofficial user-driven platform for Oculus Quest games

Spawn	To generate
String	A data type consisting of a series of characters
Support Vector Machine (SVM)	A machine learning algorithm used for classification
Unity	A game engine and development platform which offers a real-time physics engine and customizable render pipelines. Unity is developed by Unity Technologies
Unreal	A game engine and development platform which offers a real-time physics engine and customizable render pipelines. Unreal is developed by Epic Games
Virtual Reality (VR)	The technology of creating a simulated experience, modern systems use head mounted displays
XR Interaction Toolkit	An API which allows for the development of flexible VR games which can be deployed to a multitude of platforms

# 1 Introduction

As virtual reality (VR) technology becomes more readily accessible, the range of possible applications of the technology is growing. May 2019 saw the release of the Oculus Quest, one of the first commercially available standalone VR headsets to offer six degrees of freedom of motion (see Figure 1.1). In December of the same year, a software update added hand tracking capabilities to the Quest, making it the first VR headset with built-in hand tracking. Optical hand tracking is the process of using cameras and sensors to detect the position and rotations of a user's hands (and fingers), and then applying these transforms to three dimensional (3D) models of hands inside a virtual environment to mirror the user's real hands. The addition of hand tracking made the Quest a powerful tool for development of experimental applications.

## 1.1 Application

One such application of the Quest's hand tracking is sign language recognition. Sign language uses hand poses and gestures as a form of vocabulary. The recognition of such signs could be used for a variety of purposes including education, chat rooms, games, etc. This project aims to explore the use of optical hand tracking to teach key dimensions of American Sign Language (ASL). The game will focus on teaching the ASL alphabet, as well as developing fingerspelling skills. The ASL alphabet consists of 24 static signs and two dynamic signs, which poses an interesting challenge of discerning between the two forms. This project will explore the current capabilities of the Quest's optical hand tracking with the goal of using this technology to teach ASL through a serious game. Through this the strengths and weaknesses of the Quest can be assessed. The project will give some insight to how well the Quest's hand tracking fits ASL production and potentially other applications in the future.

## 1.2 Hand Tracking Technology

The Quest's hand tracking is an important step in the evolution of hand tracking technology. Prior to this device coming on the market, the most notable commercial solution was the Leap Motion Controller (LMC). That device connected to computers via USB, and Application Programming Interfaces (API) made the device compatible with Unity and Unreal for game development. The device could be attached

to VR headsets for hand tracking implementation in VR applications. While this solution worked, the detection range was narrow and required cables, thereby limiting the freedom of the user. The Quest has the advantage of having hand tracking built-in and being untethered, arguably a purer virtual reality experience. The LMC has been used in studies to detect American Sign Language signs, however few, if any, studies have tested the application in VR or applied the use of sign detection towards education.



Figure 1.1: Oculus Quest VR Headset (Oculus, 2018; [Oculus.com](https://www.oculus.com))

### 1.3 Groups of Interest

This project may be of interest to several groups. Gesture recognition systems integrated into VR hold potential to be used for a multitude of purposes. This project will attempt to use gesture recognition for sign language instruction in a serious game. This could prove useful for sign language educators.

American Sign Language is taught in schools, where the use of educational games is becoming more common. Educational games have been shown to provide specific learning advantages, especially in the linguistics field. More research is needed to determine if the advantages apply to manual languages such as ASL, but this study may show potential for the use of serious games in ASL classrooms.

ASL is taught largely to two groups: deaf children and their relatives (especially parents). As younger generations become more accustomed to immersive tools, they will be well equipped to make use of the unique advantages extended reality (XR) offers. Parents may not engage with digital games in the same

way, but still stand to see the benefits of serious games. This project will specifically look at fingerspelling, an area of ASL which non-native learners struggle with. A serious game could provide the context and motivation a parent needs to make improvements in this area.

Another group which could take interest in this project is experimental game developers and user interface (UI) designers. This project will explore the new design choices posed by the lack of physical peripherals or interaction methods. Gesture recognition opens a host of new interaction methods. By using hand tracking, the user is left with their hands open to interact with real-life environments, however it takes away all the functionality which can be mapped to buttons and joysticks. A gesture recognition system could be used in virtual reality games as locomotion method, menu navigation, spell casting, data input, etc. VR versions of role-playing and exploration games like Minecraft, Skyrim, or Kingdom Hearts could take advantage of such systems. The same system could be applied to user interfaces in augmented reality (AR) apps. Apps requiring text input could use a system of ASL fingerspelling to fill such requirements. AR headsets could use gesture recognition for functions such as volume control, play/pause, closing windows, and other common functions. These functions might be well used in apps focused on interior design, digital finger painting, or digital puppetry. As peripheral-free technologies become more prevalent, designers will be faced with the challenge of creating natural interfaces, for which gesture recognition could be a perfect solution.

## **1.4 Structure of this Dissertation**

This dissertation is structured in six parts:

- Literature Review
- Materials
- Methods
- Implementation Results
- Evaluation
- Conclusion

The literature review discusses previous research conducted in areas pertinent to this study. Materials and Methods provides an overview of the development of the ASL Fingerspeller game. Implementation

results describe the finished ASL fingerspelling app and its functionality and gameplay. Evaluation discusses feedback, testing, and efficacy of the game. Finally, the conclusion summarises findings, highlight achievements, and point out areas for improvement and future work. Ethical approval to carry out this research was given by Dr. Daniel Livingstone in accordance with the ethical guidance set by The Glasgow School of Art.



## **2 Literature Review**

Digital games have quickly become one of the most popular recreational activities across the globe. Since the rise of mobile technologies, games have become more accessible than ever. Digital games are played on PCs, consoles, handheld devices, mobile phones, and VR headsets. As games have become more accessible and more prevalent, educators have explored the possibility of exploiting games for educational purposes. Sykes (2018) points out that games designed to encourage self-improvement have become popular, such as Pokémon Go for walking and community building. The success of such games shows the potential for digital games to be used for educational purposes.

### **2.1 Digital Game Based Language Learning (DGBLL)**

Reinhardt and Sykes (2012) make a case for the use of games in education. They argue that learning can be game-enhanced, or game-based. Game-enhanced learning refers to implicit instruction when players pick up the knowledge or skills as they play without being made aware of their learning (Reinhardt and Sykes, 2012). Players may learn the content to enhance the gameplay, rather than play the game specifically to learn the content. Games such as Portal 2 can promote learning of puzzle solving and teamwork but are not explicitly designed to teach these skills. Game-based learning refers to the explicit instruction given through games designed to teach a specific subject (Reinhardt and Sykes, 2012). These games are usually a better fit to be used in classroom settings than commercially developed games. It is still unclear if one approach produces more substantial learning than the other and more research is needed on that point. Sykes (2018) suggests that as commercial games become more accessible to larger audiences, they may provide better learning experiences for more students globally.

Regardless of whether learning is game-based or game-enhanced, the advantages of learning through a game are the same. Sykes (2018) makes the case that digital games are:

- Customizable
- Repeatable
- Offer multiplayer over distance

Digital games offer repeatability through short play segments or save states. Short play segments mean having the player complete one or two tasks within a time frame, usually under five minutes. This lets the player practice the same skills multiple times in one gaming session. Save states are a saving system which allows the player to return to a previous point in the game and play it over again. These are used in narrative based games, allowing a player to save their progress and return to key points in the game, repeating scenarios to extract more information each time.

In addition to these advantages, Reinhardt and Sykes (2012) put forward five pedagogical advantages of education digital games:

- Learner-directed goal orientation
- Opportunities for interaction with the game, through the game, and around the game
- Just-in-time, individualized feedback
- Relevant narrative and context
- Motivation

One of the most important of these is just-in-time, individualized feedback. A key feature of digital games is the player being informed when they make mistakes and are then encouraged to rectify them quickly. This is applicable to language learning, where mistranslations can cause discrepancies (Reinhardt and Sykes, 2012)

Motivation is also an important aspect of DGBLL. Games which properly embody learning into enjoyable mechanics can foster motivation to play the game more. This is difficult to achieve and is a mark of excellent game design (Reinhardt and Sykes, 2012).

A collective of researchers from institutes in Taiwan (Hung *et al.*, 2018) reviewed 50 studies on DGBLL and discovered trends which give us indications of the state of the field. The key findings were:

- Most DGBLL games are custom built by researches for use in studies
- The most common platform for such games was PC
- English was the most common focus of DGBLL
- Test samples were mostly made up of university students

Studies using custom built games have tended to yield findings that suggest DGBLL games are pedagogically effective, yet the same conclusion cannot be drawn about commercially released educational games. PC is likely the most used platform as PCs are accessible and familiar to most participants. Further study is needed to draw conclusions about pedagogical efficacy of games played on alternative consoles.

The use of DGBLL for Sign Language instruction is not novel. Researchers from the University of Tunis conducted a study using a PC game designed to teach Tunisian-Arabic Sign Language to children. The participants (children) mostly reported that the game seemed useful and was satisfying (Bouزيد *et al.*, 2016). They unanimously reported that the game was easy to use and learn from. Researchers noted that the children enjoyed the presence of a signing avatar. The enjoyment of a flat screen memory matching game could be a positive indication for participants to enjoy a virtual reality game in which a player can use their own hands. This study did not test pedagogical efficacy, so conclusions about pedagogical effects of DGBLL Sign Language games cannot be drawn. While the sample size for this study was statistically insignificant and the game focused on Tunisian-Arabic Sign Language, it shows potential for the usefulness and satisfaction of a VR ASL game.

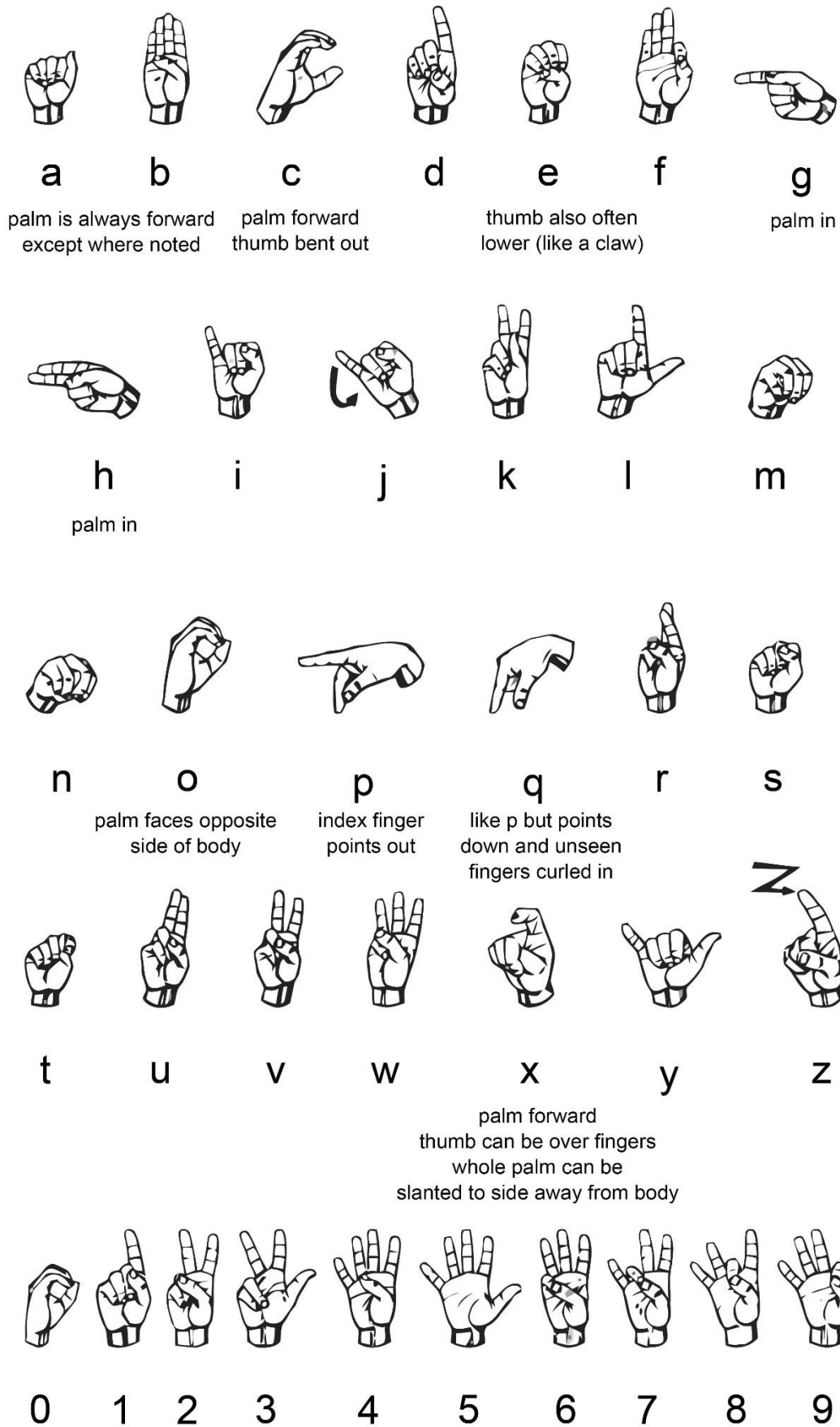


Figure 2.1: The ASL Fingerspelling Alphabet (NIH, 2019; [NIDCD.NIH.gov](https://www.nidcd.nih.gov))

## 2.2 ASL Instruction and Fingerspelling Comprehension

American Sign Language (ASL) is the de facto manual language for the Deaf and Hard-of-Hearing (HoH) community in North America. ASL has roots in French Sign Language and is not closely related to British Sign Language (Snoddon, 2017; NIH, 2019). A common misconception about ASL is that it is a manual translation of English. This is untrue and can be detrimental to an ASL learner's development. ASL has its own grammatical structure and vocabulary which is not equivalent to a 1-for-1 translation from English (Snoddon, 2017). It is important to recognize the history and culture of the American Deaf/Hard-of-Hearing community and their language.

One aspect of ASL which does have connections to English is fingerspelling. Fingerspelling is the practice of producing consecutive signs representative of English letters to spell out a word. Fingerspelling is used to spell names, proper nouns, or English words for which there is no equivalent dedicated sign (Snoddon, 2017; NIH, 2019). Each English letter has a manual form (see Figure 2.1). Twenty-four are static, and two (letters J and Z) are dynamic, in which the letter is traced with a certain fingertip movement. Native signers can produce signs in quick succession, spelling out words at high speeds, just as many native English speakers can write or type quickly. Fingerspelling is an area in which second language (L2) ASL learners struggle (Quinto-Pozos, 2011; Geer, 2016).

Research into fingerspelling comprehension conducted by Geer and Keane (2014; 2018) has given some insight into how L2 learners develop fingerspelling comprehension skills. When learning the ASL manual alphabet, learners are often shown pictures of the static form of the letter. As a result of this, they focus on recognizing the canonical held forms of letters, rather than being able to see them made during transitions from letter to letter (Keane and Geer, 2014). This can be thought of similarly to having to “sound out” a word while reading, rather than being able to recognize a word as a whole. It is thought that native signers can recognize whole words from the “shape” of handshapes and the transitions between them. “Shape” of a sign refers to the stretch/curl of the fingers. Tall letters have at least one outstretched finger (D, U, W). Short letters used curled fingers or closed hands (A, N, S).

Geer and Keane (2014) conducted a study using 16 ASL students at the University of Texas at Austin, 12 of whom were native English speakers. Each participant was presented with two sequences of videos of a person fingerspelling English words. One video blacked out the transitions between canonical forms leaving only the static holds of the letter, while the other video blacked out the holds leaving only the transitions. Participants were asked to write the word they had just seen fingerspelled. The results showed that L2 learners perform much better when shown static signs. They performed better even than when shown videos of fingerspelling with no segments removed, indicating L2 ASL learners' dependency on canonical hand forms (Keane and Geer, 2014).

Geer and Keane have conducted several studies focused on the importance of implicit vs explicit instruction. In two experiments of 63 and 80 participants, results showed that L2 learners struggle with hand shapes produced in non-perfect form (Keane and Geer, 2014). The most significant of this was the recognition of the letters K and P, which differ only by wrist rotation. Multiple participants interpreted a spelling as "Kortugal" when the actual word was "Portugal". Geer (2016) also notes that native signers flex their wrist forward when producing the letter Y. This flexion indicates the letter Y, even if the fingers do not hit their canonical form (Geer, 2016). Explicit instruction can help L2 ASL learners pick up on subtle differences and improve their fingerspelling comprehension skills. Implicit instruction was not as significantly impactful in short term learning (Geer, 2016).

Across North America there are different dialects of ASL and variances in forms. For example, when fingerspelling words with an R following a U, some signers will combine the two and a wrist flexion. Geer and Keane (2018) found that unless specifically instructed about such variances, participants were unable to discern these forms and were unable to recognize the words being fingerspelled. This further shows that L2 learners focus on canonical handshapes and that short-term improvement is best fostered through explicit instruction.

These studies from Keane and Geer are important as they highlight some of the issues L2 ASL learners struggle with while practicing fingerspelling. The results are particularly interesting as they show that L2 learners develop fingerspelling comprehension skills much differently from native signers. Native

signers can develop skill through implicit instruction over long periods of time while exposed to other signers. Studies exploring the use of implicit instruction on L2 ASL learners over long periods could find results contrasting to the results found in Geer and Keene's studies.

Digital fingerspelling tools have been in development for over two decades. In 1998, Su and Furuta developed an online fingerspelling comprehension tool using VR Modelling Language. The system allowed users to input a word and watch a 3D model of a hand spell the word. A system like this could be perfect for incorporating into a game designed for fingerspelling comprehension skills. The researchers noted that their system was slow, however technological advances over the last 20 years would improve this system. (Su and Furuta, 1998)

### **2.3 Gesture recognition technology**

A digital game which teaches the player sign language could make use of a system that recognizes signs and hand gestures. Research in the field of gesture detection has advanced over the last decade. Several solutions exist, most notably motion capture systems, gloves fitted with accelerometers and flex sensors, and the Leap Motion Controller (LMC). Of these solutions, the LMC has been most studied by academic researchers, likely due to their low cost and ease-of-use. The LMC works using two Near-Infrared cameras, and three LEDs to detect hands at a range of 20-60cm (Ultraleap, 2019). Software Development Kits are available for use of the LMC with both Unity and Unreal, as well as compatibility with VR headsets such as the HTC Vive or Oculus Rift.

Since the release of the LMC in 2012, researchers have explored the application of the device for gesture recognition. A notable focus of such research has examined the use of machine learning and neural network sorting algorithms. These algorithms are used to compare captured signs against datasets of example signs or other captured signs and use the results to refine the calibration of the gesture recognition.

An early study compared a k-Nearest Neighbour (k-NN) to a Support Vector Machine (SVM) algorithm in recognizing gestures using an LMC. The k-NN model achieved an average detection accuracy rate of 73% while the SVM model achieved 80% (Chuan, Regina and Guardino, 2014). These

figures can be compared to a similar study which used a k-NN algorithm and resulted in an average detection accuracy rate of 82.5% (Clark and Moodley, 2016). The statistics are respectable for such early stages in the technology but showed room for improvement.

Improvement was found with the incorporation of Hidden Markov Models (HMM). One study achieved an 86% average detection accuracy rate when testing 24 ASL signs using an HMM. Furthermore, an average accuracy rate of 93.14% was achieved when using an HMM to classify signs detected by two LMCs simultaneously (Fok *et al.*, 2015). This result also shows potential for multi-sensor systems. While these studies demonstrate the evolution of the field, none were statistically significant.

One of the most significant studies to date used 146 participants testing 32 static signs of ASL letters and numbers with a Multilayer Perceptron (MLP) neural network. It reported an average accuracy rate of 90% (Mapari and Kharat, 2016). While this study is more statistically significant than others, it is difficult to tell if this success comes from a larger test sample, the MLP neural network, or both. Regardless, it further demonstrates the prevalence of neural networks in this field.

A recent literature review of hand gesture recognition techniques (Cheok, Omar and Jaward, 2017) highlights some trends seen in the cases previously described. While the field is active and under development, no test of statistical significance had yet been carried out, and therefore no system was close to commercial readiness. HMMs are used prevalently in dynamic gesture recognition while SVMs have been used for static gesture recognition. The study does note that while sign language vocabulary is vast, the catalogue of gestures tested is narrow (Cheok, Omar and Jaward, 2017). As the technology progresses, neural networks improve, and more tests are carried out the field will approach a viable commercial solution to gesture recognition software. Much more work is needed, however, to support full sign language integration.

## **2.4 Summary**

The literature shows some important points to consider when exploring future uses of the technology. Firstly, digital games offer a set of advantages for teaching, especially teaching languages, however this research mostly comes from digital games using the PC platform, so the efficacy of digital games on other



platforms is yet to be determined. Secondly, American Sign Language is a complex and intricate language, just as spoken languages are, and proposes a certain set of challenges for integration in language recognition technologies. Thirdly, gesture recognition systems are improving in conjunction with hand tracking technologies, but sign language recognition is still in its infancy. The technology still faces accessibility issues in terms of hardware, and so commercial, in-home solutions are not yet available. Keeping these points in mind, the questions can be posed:

- Does virtual reality (VR) and optical hand tracking in its current state provide a sufficient platform for ASL education?
- What are the limitations preventing ASL education and communication in VR platforms?
- Will the use of a serious game improve a player's ability to use fingerspelling?

The following study will attempt to answer these questions through the development of a VR serious game using hand tracking technology to teach the ASL alphabet and fingerspelling skills.

### **3 Materials**

To evaluate the efficacy of a VR serious game about ASL, the first challenge is to create such an application. At the outset of the project, there was no other game developed, to my knowledge, using any hand tracking technology with the intent of ASL instruction, hence the progress began from scratch. I set out to develop a game to use Oculus Quest's hand tracking technology to teach the ASL alphabet and improve fingerspelling skills. This game should make use of a gesture recognition system to indicate when users are producing signs correctly. Through this, a list of recognizable signs will be produced, and a list of signs which the system is unable to recognize. Analysis of this list will give insight to the strengths and weaknesses of the hand tracking/gesture recognition system. After having users play the game, conducting a survey will allow evaluation of the pedagogical efficacy of the game: whether it improved fingerspelling skills or not. Through this study, it will be shown what improvements could be made to hand tracking technology to better suit use for ASL instruction.

#### **3.1 Virtual Reality Headsets and Hand Tracking Solutions**

##### **3.1.1 Oculus Quest**

A key piece of equipment for this project is the Oculus Quest. The Quest is a virtual reality headset which uses inside-out optical tracking to provide untethered six degrees of freedom (6DoF). 6DoF refers to translation through three dimensions, and rotation on three axes. The Quest has four wide-angle, monochromatic cameras; one on each corner of the front of the headset. The field of view of each camera is angled in such a way that there are overlapping regions, which is crucial for stable spatial tracking. The two upper cameras, which are rarely obscured, can track ceilings while the two lower cameras can track the floor. All four cameras can be used for tracking walls and other objects. These same cameras are used to track the user's hands.

##### **3.1.2 Hand Tracking**

Oculus is a subsidiary of Facebook. It was able to source photographic data from millions of devices with Facebook owned applications installed. These images were used to train neural networks to recognize hands and fingers. The network was so well trained and computationally optimized that the

recognition system could be run on mobile processors, such as the Snapdragon 835 in the Quest. Hand tracking was released as a beta feature to the Quest in December 2019, along with the API needed for app development. The API is regularly updated, and functionality is adjusted frequently.

### 3.1.3 Alternative Solutions

The Quest was selected due to its advantages over other systems such as the Leap Motion Controller. The LMC is an accessory to VR headsets, meaning that both the headset and LMC would need to be purchased (see Figure 3.1). An LMC retails for roughly \$90 (£70) at time of writing, while the two most popular compatible headsets, the Oculus Rift and the HTC Vive, retail for roughly \$600 (£460) and \$700 (£540) respectively. The Quest base model retails for \$400 (£310). Both LMC compatible headsets are tethered devices, meaning they are reliant on high power PCs for graphical processing. The Quest is an all-in-one device, meaning all processes are executed by the processor in the headset. While this means the Quest is computationally much less powerful and less graphically capable than the other two headsets, the Quest has the distinct advantage of being wireless, thereby giving the user more freedom of movement. Additionally, built-in hand tracking developed and supported by the manufacturer of the device makes development easier as the systems work together seamlessly. It is not dependent on an additional API to join hand tracking with the operating system, as well as the API necessary for game engine integration.



Figure 3.1: HTC Vive Headset with LMC Attached (Ultraleap 2016; [Blog.Leapmotion.com](https://blog.leapmotion.com))

## **3.2 Development Tools**

Development of the game was done on a Windows PC with a Ryzen 2700X processor, RTX 2070 graphics card, and 16 GB of RAM.

### **3.2.1 PC to VR Workflow**

A crucial part of the development process was the use of Oculus Link. Oculus Link is a software which allows the Quest to be used as if it were a tethered headset, much like the Oculus Rift S. The software is usually used to play PC VR games on Quest. Oculus Link allows for games being run in a game development platform to be play-tested on the Quest without having to build and deploy the application.

On February 6, 2020, an update to the Oculus Unity Integration software allowed for Quest hand tracking to be used in Oculus Link. This means that data generated at playtime could be accessed on the PC, saved, and manipulated for use in development of the game. Without this system, a gesture recognition system could not be developed as gestures cannot be saved and given meaning within the context of the game. The Oculus Quest 16.0 firmware update broke this functionality on April 13, 2020, effectively killing development of the game. Fortunately, an update to Oculus Unity Integration package fixed this functionality on April 28th. Development of this project has only been possible for several months.

### **3.2.2 Game Engine**

The game was developed in Unity. Unity is a game engine and development platform that is free to use for non-commercial products and small businesses. Unity boasts several advantages such as a built-in physics engine, support for external packages, and several render pipelines. Unity supports scripts written in C#, which makes it a flexible platform. One of the greatest advantages of Unity is the Oculus Integration package which provides the tools needed to use Quest hand tracking in a game. A similar package is available for the Unreal engine, but Unity was chosen due to prior experience with the platform.

### 3.2.3 Hand Tracking Compatibility

The Oculus Integration package provides a VR compatible camera system, hand tracking ability in Unity, hand models and hand management. This means the game will automatically detect hands and update the hand models each frame. If hand tracking is lost, the hand model is destroyed, and a new hand model is generated when tracking is resumed (see Figure 3.2). This removes problems that would otherwise take large amounts of work to fix.

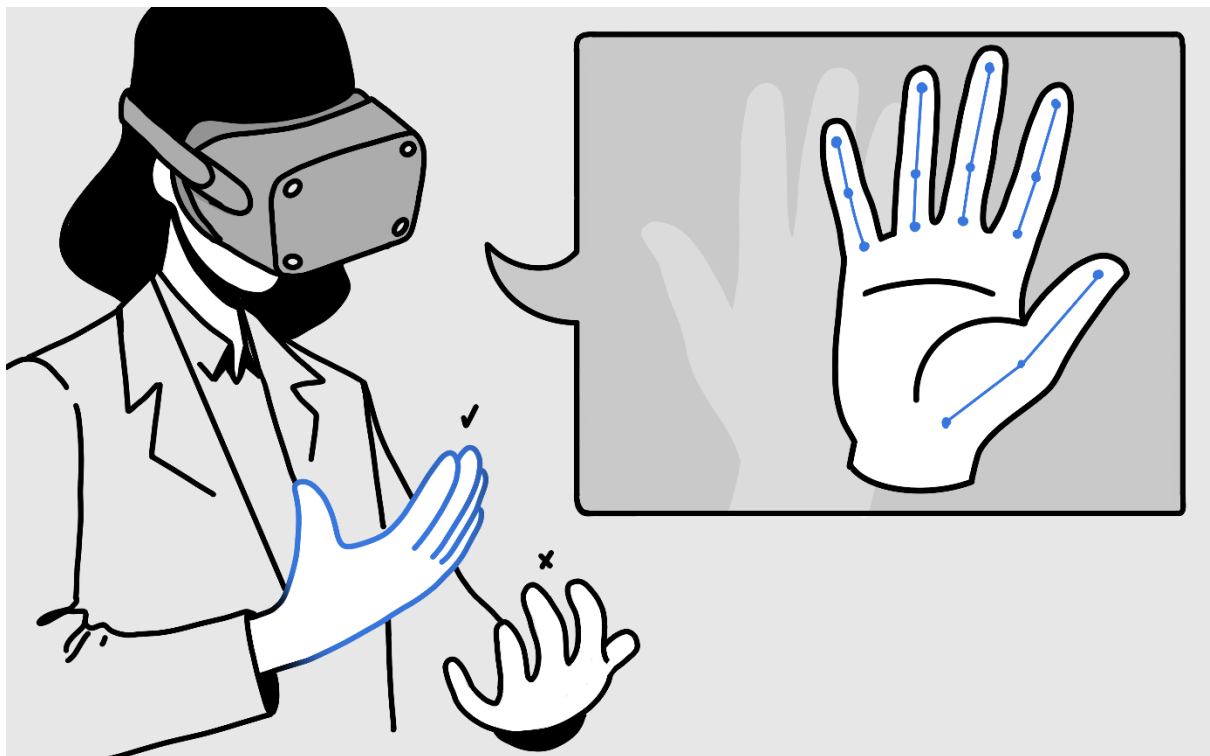


Figure 3.2: Hand Tracking Under Occlusion (Oculus, 2019; [Developer.Oculus.com](https://developer.oculus.com))

Oculus Integration provides data about each hand being tracked. Each hand has a tracking confidence level, positions, and rotations for 22 “bones” in the hand, whether each finger is pinching (touching the thumb), and how tight the pinch is (open, close to touching, or touching the thumb). The “bones” refer to the wrist, each joint in the thumb and fingers, and the thumb and fingertips (see Figure 3.3). These data points can be used to classify gestures or signs. It may be important to note that using the Oculus Integration package may be falling out of fashion, with many developers favouring the Unity XR Interactions Toolkit. The XR Toolkit offers more flexibility and support for multiple headsets and may be

more future proof. However, the XR Toolkit does not offer hand tracking support, so Oculus Integration was the only available choice for this project.

OVRPlugin.BoneID.Hand\_

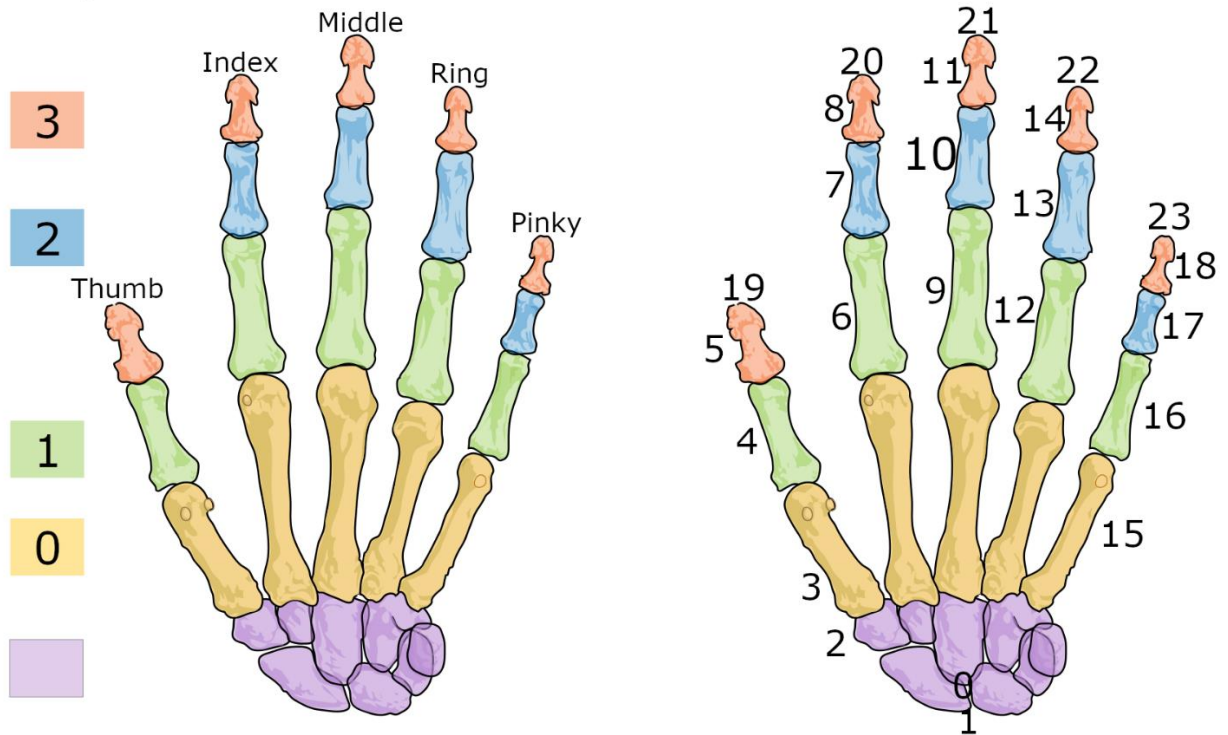


Figure 3.3: Oculus Integration Bone ID Legend (GowerGames, 2019; [Reddit.com](#))

### 3.2.4 Modelling and Animation

Modelling and animation for this game was done using Blender. Blender is a free and open-source tool used for 3D modelling, animation, and many other processes. Some models were taken from Mixamo, a platform owned by Adobe which provides easy rigging and animation.

## 4 Methods

### 4.1 Planning

Once the tools have been defined, the project turns to development methods. The first step was to develop a game design document. This is a planning document which outlines the gameplay, goals and features, mechanics, and art style of the game (see Figure 4.1). Research has shown that L2 ASL learners struggle with fingerspelling comprehension, so in the early stages of planning, the possibility of a fingerspelling comprehension challenge was considered. This would consist of having the player watch a video of a person or 3D avatar fingerspelling a word, and then selecting that word from a list of choices. However, due to time constraints, this game mode was dropped from the development plan. Instead, it was decided to focus on fingerspelling production and sign recognition, as this better demonstrates the strengths and weaknesses of the Quest's hand tracking.

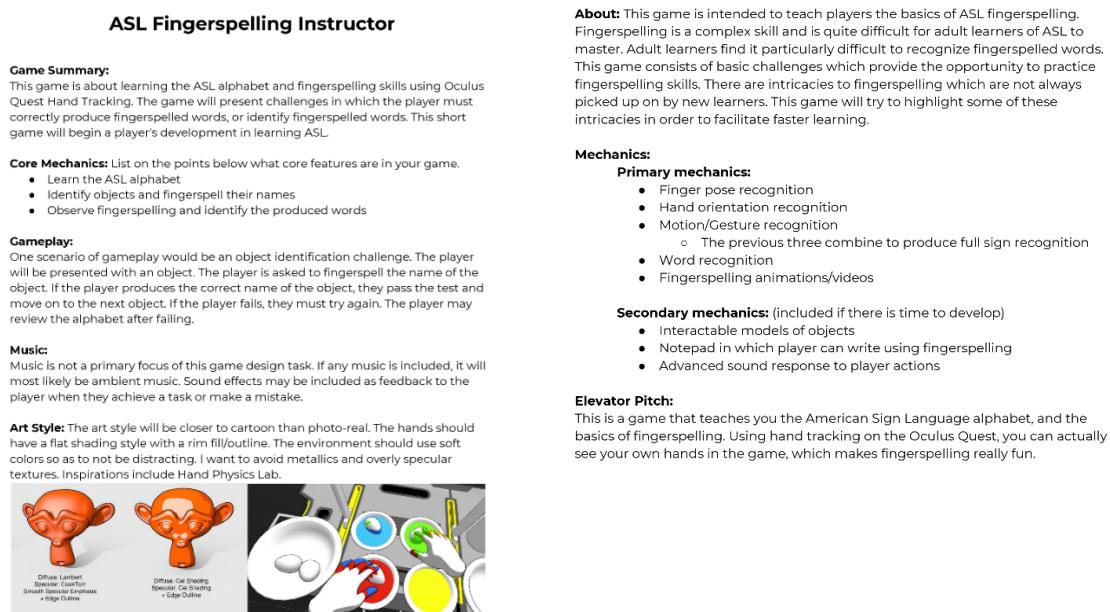


Figure 4.1: Games Design Document

Ideas for a fingerspelling challenge included simply presenting a written word to the player and asking them to produce the ASL signs for each letter, or presenting the player with a picture and asking them to produce the name of the object/animal/etc. This may be more fun but could introduce trouble as pictures may be interpreted differently (e.g. rabbit vs bunny) and make the game inaccessible to non-

English speakers. For this reason, the game deliberately focuses on one player mechanic, sign production. This keeps the game simple to understand.

The art style was kept simple, both for aesthetic and development reasons. Early ideas included toon shaders or matte lambert shaders. This game design document proved to be a helpful guide but was adjusted to respond to user needs as the game evolved.

## **4.2 Sign Detection**

The first challenge of game development was creating a sign detection system. This requires a technical definition of what a sign is. For 24 signs of the ASL manual alphabet, a sign is the static position and flex of all fingers and thumb, and the rotation of the wrist. As Geer showed in her research, wrist rotation is an important part of sign classification, as some signs differ only by wrist rotation, and two signs (J, Z) are dynamic.

### **4.2.1 Defining a “Sign”**

The Oculus Integration provides global positional and rotational data for each joint in the hand. Using this information, a data structure can be created which will hold the position of a certain number of joints in the hand, and the rotation of the wrist. This structure can be called a ‘sign’. Positional data of each joint can be stored in a list of Vector3s (a data type consisting of three floating point numbers). An important aspect is that joint data, by default, is returned in global positioning. It is highly impractical to produce a hand gesture in the exact same global position, so instead joint position should be saved and checked relevant to the wrist local position, which will always be 0. Rotational data can be stored in the Quaternion data type. Wrist local rotation will always also be 0, so it must be saved as a global rotation (see Figure B.2).

### **4.2.2 Sign Saving and Detection**

To test a sign recognition system, there must first be signs to recognize. Signs could be created manually by inputting data, but this is highly impractical as each sign consists of 70 data points. Instead a function was written to save the positional data of every joint and the wrist rotation of a specified hand on



the frame the function is called. The positional data of each joint is calculated local to the wrist using the `InverseTransformPoint` method (see Figure B.3). This system only saves static signs.

The next step is devising a method of detecting these signs. A player is unlikely to ever reproduce a sign with all joints in the exact same position and wrist rotation as when they saved it, so the system must allow the user some variation – referred to as “tolerance”. Each joint’s position is compared iteratively to the equivalent joint position of the sign being checked. If any joint is further than the set tolerance, the check is stopped. If no joint position fails the distance test, the wrist rotation is checked using a dot product calculation. This equation compares the two quaternions and returns a value between -1 and 1. -1 corresponds to completely opposite quaternions, and 1 corresponds to completely similar quaternions. A value of 0 corresponds to perpendicular quaternions. The resulting dot product is compared to a rotational tolerance value. For easier comprehension, the angle tolerance is subtracted from 1 before comparing against the dot product. If the dot product is greater, the sign is considered produced correctly, and marked as detected.

### **4.2.3 Dynamic Signs**

Just as video is a sequence of still photos, a sequence of still poses could be saved to constitute a dynamic sign. Detecting this would be more difficult. Reproducing a motion with accuracy is more difficult than producing a single hand pose. The system could check all poses through the motion and consider the sign produced correctly if a certain number of poses met the distance threshold. Rather than spend time developing and implementing this complex system, I decided to cheat the static sign system I already had. Thinking of a dynamic sign as a series of still poses, then there must be a final pose. For the sake of the simplicity, the final pose of both dynamic signs was saved so that by producing the whole motion, the sign would be detected at the end.

### **4.2.4 Sign Detection Implementation**

Implementing the detection system presents several choices. First, should it check the user’s hands for signs upon request or continuously? Either is viable but having the system check upon user request may prove tedious to the player. If they struggle to produce a sign correctly, they will be requesting a check

over and over, reducing the game's fun and slowing down their production rate. Additionally, requesting a check does not mimic real life sign production. Native signers produce signs quickly and sequentially without breaks to check if the listener is comprehending each individual letter. So, the "continuously checking" option is preferred albeit computationally more strenuous (see Figure B.4).

The next choice is whether the system should check for any sign, or only the sign expected to be produced. Checking for all signs offers the advantage of being able to count how many times signs other than the expected sign were produced, providing a form of feedback to the player. Checking all signs on all frames means the game is checking a possible maximum of 22 bones, for 26 signs, 60 times a second. In testing, the hardware kept up with these demands, so this option was selected as it offers the most functionality.

### **4.3 Sign Functionality**

For signs to have meaning, as they do in ASL, the game must give each sign its respective function to execute when detected. Unity Events were a perfect solution for this. Unity Events are customizable functions which can be invoked when needed. One event can hold multiple functions, meaning they are flexible and useful for testing purposes. The sign data structure holds a Unity Event, which can be customized per sign. This means that signs can be used for purposes other than spelling. Signs can be given functions such as changing colours of an object, raising or lowering volume, turning elements of the environment off and on, etc. The choice of detecting all signs on all frames also means that signs that are not appropriate for the current situation could be detected, and their functions could be executed when it is not convenient for the player. To manage this, a spell-checking system was implemented. This consisted of a simple comparison of the sign produced to the letter expected (see Figure B.1).

### **4.4 Gameplay**

One level of the fingerspelling challenge consists of five words, each randomly selected from 40-50-word lists, depending on the level. A string is set equal to the word being proposed, which at the start of the level is the first word. A null string is created to hold the letters produced by the player. If a player were to produce an incorrect letter, that letter would need to be removed from the string. Sign language is

not used as a typing mechanism, so there is no natural sign for “backspace” or “delete”. Rather than create a sign for this function, which could also be accidentally detected, I chose to implement a spell checker.

The `CheckLetter` function compares the letter assigned to the sign against the expected letter of the word. If the letter is correct, it is added to the string and the checker moves on to the next letter. If the letter is incorrect, it is discarded. Once all letters have been produced correctly, the checker moves on the next word and the spelled string is reset to null. Once all 5 words have been spelled correctly, the level has been completed and the player can be shown a summary of the gameplay.

The gameplay summary shows three statistics:

- Correct letters produced; incremented every time the `CheckLetter` function returns true
- Incorrect letters produced; incremented every time the `CheckLetter` function is returned false
- Time taken to complete the level

Two timers ran during the gameplay. The first shows time elapsed since the player was asked to spell a new word. Every time the player completes a word, the timer is reset. The second is the level timer. It begins when the level starts, ends when the player completes the last word and is shown in the post-level summary. Dividing the level timer by the number of correct letters gives a “typing speed”/production rate. These are good metrics to assess player improvement over time.

## 4.5 User Interface Design

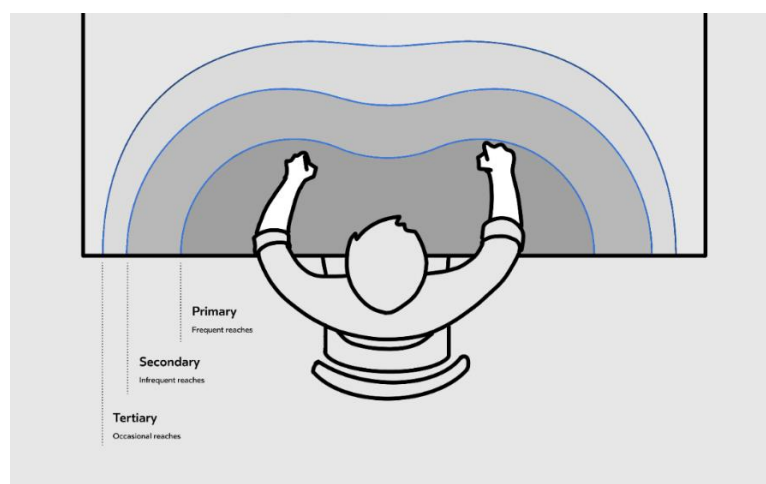
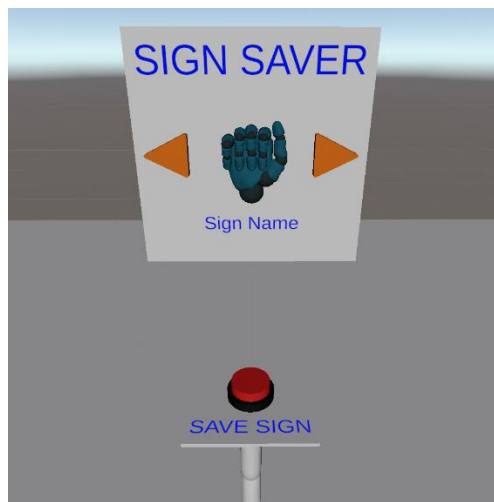


Figure 4.2: Interaction Range (Oculus, 2019; [Developer.Oculus.com](https://developer.oculus.com))

A particularly tricky challenge in game design for hand tracking apps is user interface. Without peripherals such as a mouse or controller, there is no familiar way to navigate an interface. Gestures could be used to navigate through the game, but requires the players to learn new gestures, on top of the ASL alphabet signs. A trial version of the game tested “physical buttons” in close range of the player which they can push with their hands (see Figure 4.3). They worked but did not offer haptic feedback. The sensation of seeing your hand touch, but not feel a button can be unsettling.



**Figure 4.3: Implementation of a "Physical" Button in Early Stages of Development**

Oculus Integration offers a solution called the Hands Interaction Toolkit (HIT) which includes a ray casting tool used for far-field interactions (outside of arm’s reach). The ray cast extends from the palm of the hand and latches on to designated “buttons”. The user can then “click” on the button by tapping their thumb and index finger together (see Figure 4.4). A small script is added to the far-field buttons to implement Unity Events functionality. The buttons can change colours when selected, to give the user confirmation of their selection.



**Figure 4.4: Tap-to-Click System Developed by Oculus (Oculus, 2019; [Developer.Oculus.com](https://developer.oculus.com))**

## **4.6 Finishing Touches**

With a gesture recognition system, spell checker, and interaction method, we have all the tools needed to make the game. The game needed a way to teach the player the signs of the ASL manual alphabet. A full rigged body model was taken from Mixamo, from which only the hands were used. A library of hand poses for each sign, and animations for the letters J and Z were created inside Blender. The models were used in both the sign calibration mode, and the practice mode.

## 5 Implementation Results

This chapter provides a succinct description of the finished ASL fingerspelling app and its functionality and gameplay. The game itself consists of four sections:

- I. Main menu
- II. Sign calibration
- III. Practice mode
- IV. Fingerspelling challenge mode

For easier data handling, the entire game is set in one scene, with minor environmental changes for each mode. The environment is deliberately not physically realistic. The space is encapsulated by a white skybox with multicoloured spots, at an infinite distance. There is no visible floor or otherwise familiar environmental features which would suggest scale of distance. A physically unsituated environment removes the player's curiosity to explore. The player is more likely to remain in the starting position, which is optimal for gameplay. This abstract space allows the player to focus on the gameplay.



Figure 5.1: Main Menu

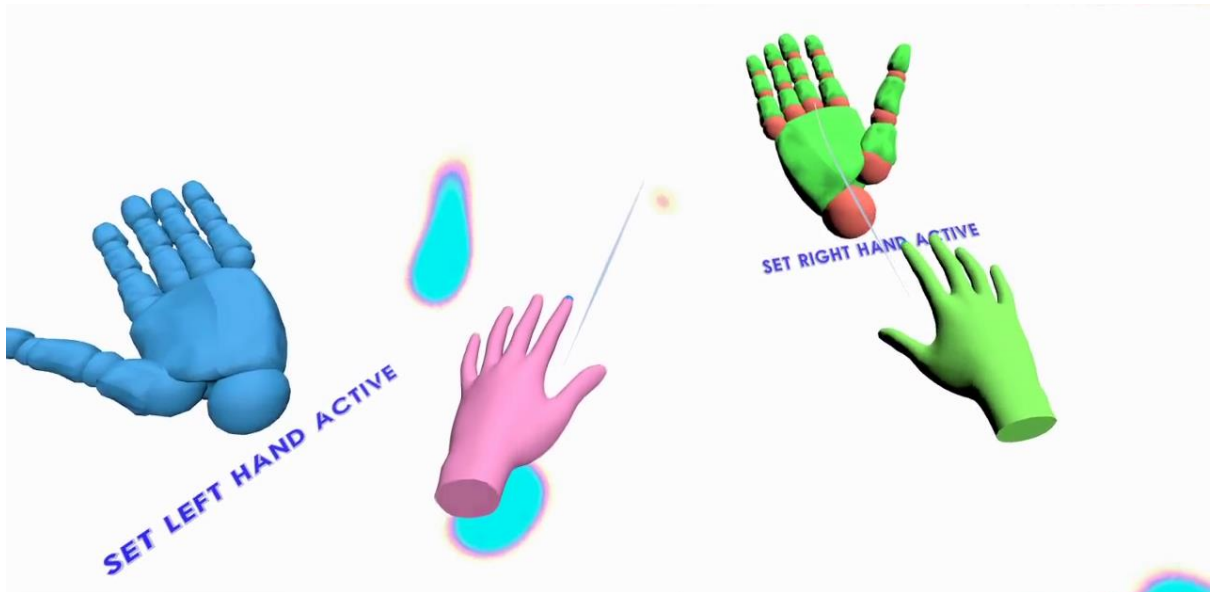
## 5.1 Menu

The game launches into the main menu which consists of four large buttons; labelled “Sign Calibrator”, “Practice”, “Level Select”, and “Quit”. Above the main menu is the game title in large letters, with hand models above spelling “ASL” (see Figure 5.1). Below the player are two large hands, one left and one right, placed, respectively. Below each is a label that reads “Set right hand active” or “Set left hand active”. The right hand is red, and the left hand is blue (see Figure 5.2 and Figure 5.3).

Players notice a beam stemming from each of their palms. These act as a cursor for menu navigation. When a beam intersects either hand, the hand turns green. By tapping their thumb and index finger together, the player selects which hand the system should use for sign detection. In development builds, the active hand is coloured green and the inactive hand is coloured pink. In the final build, a bug prevents the hands from always being coloured this way but will be correctly coloured when the palms are facing the player’s face.



Figure 5.2: Set Active Hand Menu



**Figure 5.3: Player Setting the Right Hand as Active**

When the beam intersects any of the four buttons in front of the player, the button turns green to indicate that it is being selected (see Figure 5.4). Tapping the thumb and index finger together triggers the event tied to the button, just like clicking with a mouse. This tapping as a click function was designed by Oculus developers, as touching the thumb and index finger together provides a form of haptic feedback to the user.



**Figure 5.4: Player Using The Main Menu**



## 5.2 Sign Calibration

Tapping the “Sign Calibrator” button removes the main menu and loads the calibration scene. The scene consists of a title which reads “Sign Calibrator”, a hand model and label which show the active letter to calibrate, two buttons labelled “previous letter” and “next letter”, a button labelled “Save Sign”, and a button labelled “Main Menu” (see Figure 5.5). The player can hold their active hand in the shape shown by the model in front of them (see Figure 5.6). Using their inactive hand, they can tap on the save sign button. This overwrites the data for that sign with the current position and rotation of the active hand. Upon saving, the hand model is updated to the next letter of the alphabet.



Figure 5.5: Sign Calibrator

If the player wishes to save a certain letter, they may navigate to that letter by using the “previous letter” and “next letter” buttons. The J and Z hand models each play the respective animation once upon appearing. If the player wishes to see the animation again, they may move to the next letter and then back to J or Z. Pressing “next letter” on Z cycles back to A, and vice versa when pressing “previous letter” on A. After calibrating all letters, the player may choose to return to the main menu by selecting the “Main Menu” button.

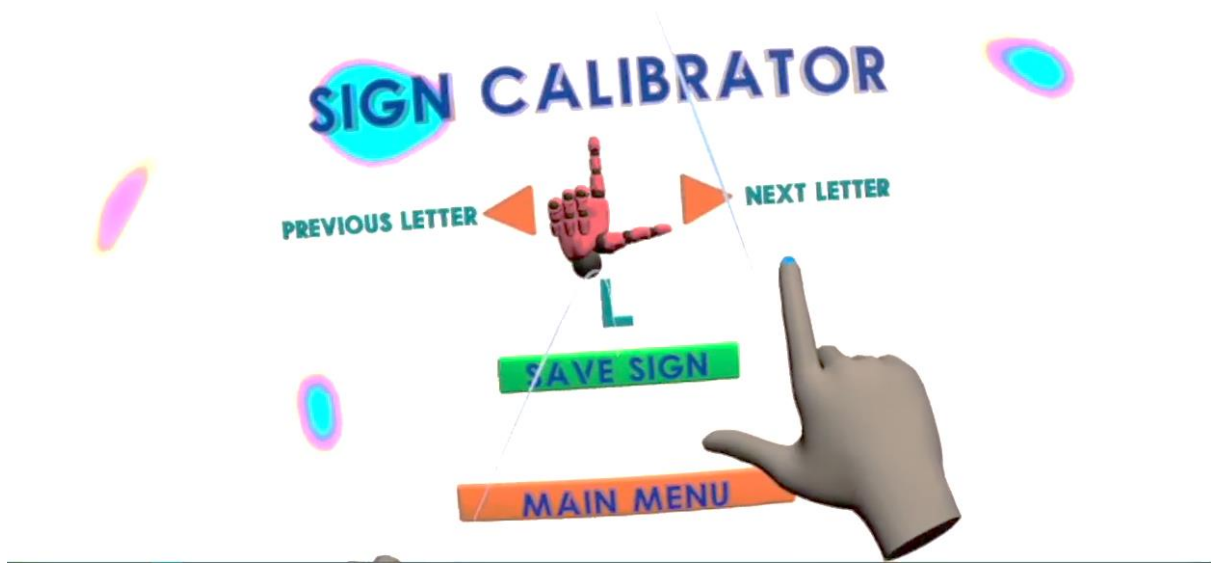


Figure 5.6: Player Using The Sign Calibrator

### 5.3 Practice Mode

The player may then select “Practice” to disable the main menu and launch the practice mode. The practice mode scene consists of 26 hand models, one for each letter of the alphabet, and their respective label. Under the letter A is the main menu button. The sign detection system is activated in practice mode. When the player correctly produces a sign, the letter will spawn in front of the respective hand model and tumble downwards. After five seconds it disappears. Each hand model serves as reference, so the player can copy each sign and practice producing the hand form correctly. The spawning letter serves as feedback to show that the form is being produced correctly (see Figure 5.7 and Figure 5.8).



Figure 5.7: Practice Mode

The sign detection algorithm checks wrist rotation, and wrist rotation can only be saved in global space. This means that if the player were to produce any sign but not be facing forward, the sign would not be recognized, which in turn means that the player would not be able to face the model of the letter they are trying to practice. To solve this problem, wrist rotation is ignored in practice mode. This allows the player to produce signs while facing any direction, but it also means that the system will not be able to discern perfectly between similar signs such as K and P, or U and H. After practicing until satisfied, the player may return to the main menu.

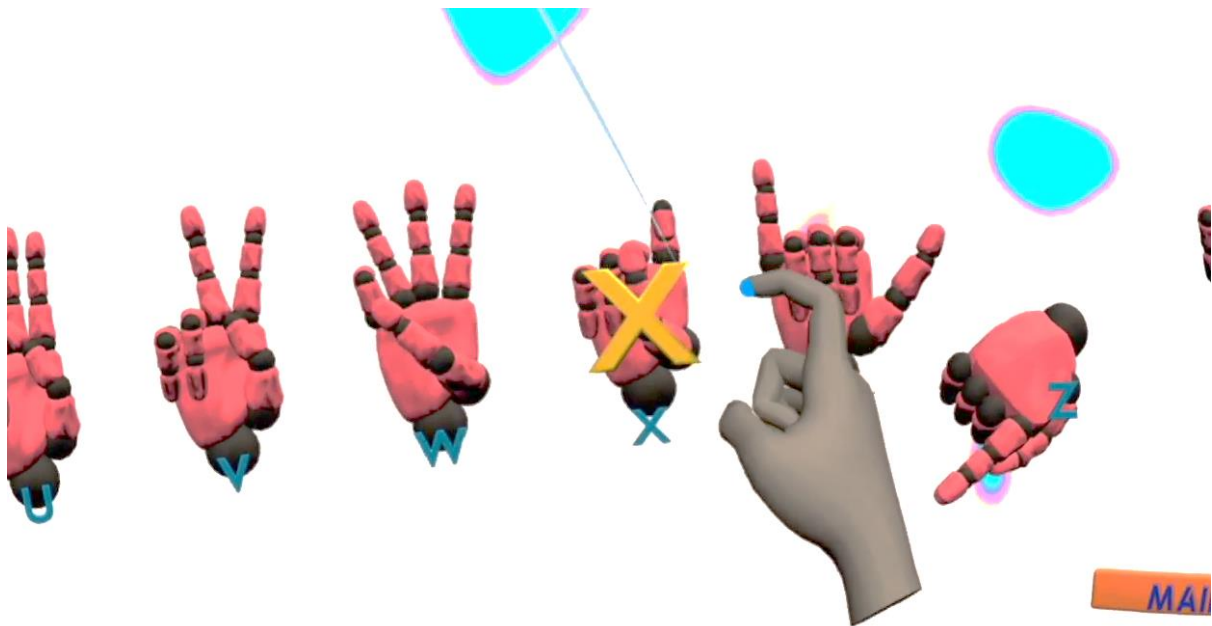


Figure 5.8: Player Using the Practice Mode

## 5.4 Fingerspelling Challenge

The next selection in the list is “Level Select”. This button brings up a new menu with six options: “Level 1” through “Level 5” and a main menu button. Selecting any of the level buttons disables this menu and activates the fingerspelling challenge scene. The fingerspelling challenge scene consists of a level title, a display of the word to spell, a display of the users spelling progress, a timer, a “skip letter” button, a “skip word” button, and a main menu button (see Figure 5.9). The sign detection system is activated upon the loading of the level. Level 1 consists of common first names, level 2 is US State names, level 3 is countries of the world, level 4 is global cities, and level 5 is names of flowers.



**Figure 5.9: Fingerspelling Challenge**

Each level asks the player to spell five words - such as “TOKYO” and “GERANIUM”. When a player produces the correct letter, a positive tone is sounded, and the letter is added to the display in front of them (see Figure 5.10). This is immediate positive feedback to the player. Once the player completes a word another positive tone is played, and the next word is loaded. Once the player completes the level, a positive jingle is played, and the player is presented with the level summary.

If the player is struggling with a certain letter, they may press the “skip letter” button. This adds the correct letter to the word and lets the player move on to the next letter. This letter is not added to the count of correct letters produced. If a player is struggling with a word, they may press the “skip word” button. This disregards the current word and loads the next word. None of the remaining letters of that word are added to the count of correct letters produced. These functions relieve player frustration if the game is not detecting signs properly.

The level summary scene shows the player how many letters they produced correctly, how many incorrect letters were detected, and how long (in seconds) it took to complete the level. On the right side the level high score from the current play session is displayed. From this scene the player may either return to the level select menu to play again or return to the main menu.

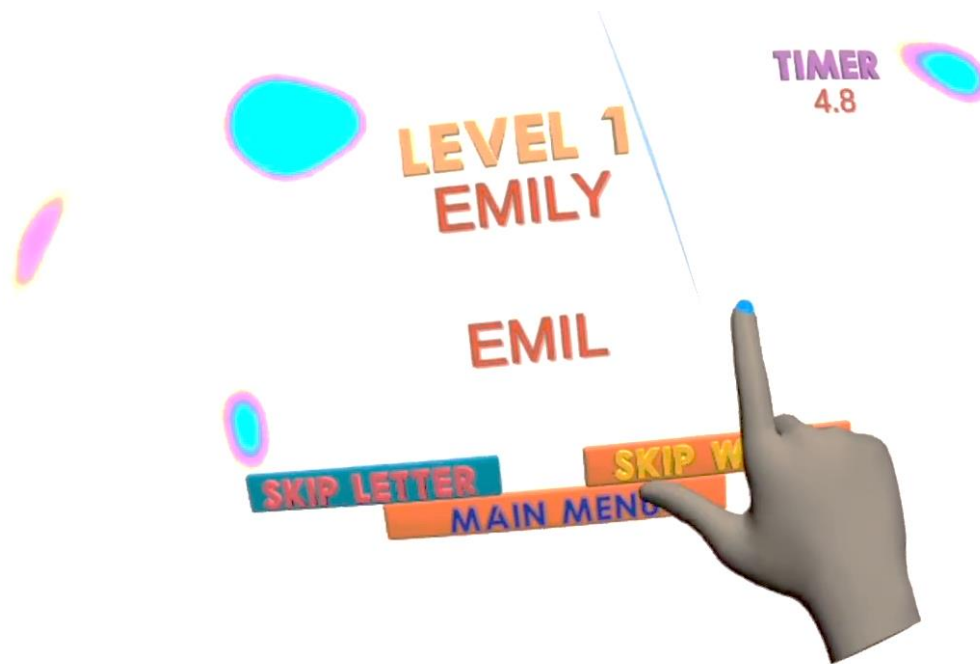


Figure 5.10: Player Playing the Fingerspelling Challenge

## 5.5 Issues

There are inevitably a few bugs still present in the final build. Ideally each of the player's hands should be pink or green, but when tracking confidence is not high, the hands are coloured grey.

The ray cast beam latches on to buttons, which helps relieve misclicks to hand instability. All buttons in the main menu disappear after clicking, but the beam remains latched to the position of the now invisible button. If the player clicks again without moving their hand first, the function will be called again. This can be especially detrimental to the game if the player tries to navigate menus with two hands. For instance, say the player hovers over "Sign Calibrator" with their right hand, and "Practice" with their left hand. Clicking with the right hand will launch the sign calibrator scene, making the "Practice" button disappear. However, the left-hand beam will stay latched to this position. Clicking with the left hand will launch the practice mode, thereby having both the calibration scene and the practice scene active at the same time. These things can be avoided by having the player move their hands after clicking any button, but players might forget to do this sometimes.

Another “bug” pertains to the scoring system and the timer. The level timer runs from the loading of the level until the level summary scene is loaded. The timer does not carry any penalty for selecting the “next word” button. This means that the player can select “next word” five times and complete the level with a fast time. The game does not differentiate this time from levels that were completed by spelling all letters. One solution could be implemented, in that by pressing “next word” a 20 second penalty could be added to the level timer. This would be an easy fix to implement, only 2 lines of code, but I had overlooked this issue before publishing the game. These bugs are very minor, but conceivably may induce some player frustration at certain points.

## **6 Evaluation**

This project sought to produce a game with a sign recognition system and hence to determine the capabilities and limitations of a hand tracking application for ASL education. There are several approaches to evaluate the game's usefulness and pedagogical efficacy.

### **6.1 Approaches**

#### **6.1.1 In-person Medium Scale Testing**

The first approach is in-person, medium scale user testing. This would consist of recruiting twenty to thirty participants to play the game one or two times. The testing would take place in an academic setting, using dedicated Quest hardware, a PC to record gameplay, and cameras to record the participants hands while playing. Participants would likely be fellow university students and staff who would likely have experience with VR technologies, and so require less training to play the game. Participants would be introduced to the hardware and the game before exploring the features of the game for half an hour. After playing the game, the participant would complete a System Usability Scale (SUS) survey and a questionnaire to gauge their impression of the game. Such survey data can be used to estimate the usefulness of the game. By recording the gameplay and hands of the user, the tracking accuracy can be evaluated, as well as signs that do and do not work well with the recognition system.

#### **6.1.2 Online Testers**

The second approach is release to the public through online platforms. The official Quest store has strict content guidelines and does not permit development builds. Fortunately, SideQuest, a third-party platform which allows users to upload home-made games and share them with other Quest users, is a good alternative. This is the de facto preferred release method for Quest developers and users. Alongside this posting, the game can be advertised through popular social media sites. These platforms allow users to respond to posts with comments, which can be used for feedback collection. Such feedback will show what general, non-academics think about using the technology for educational purposes.

### **6.1.3 Extended User Testing**

The third approach is extended user testing. Ideally this would consist of five to ten participants reporting to an academic space once a day for one or two weeks. The initial visit would take an hour for introductions and set ups, but subsequent visits would take only 20 minutes. In each visit, the participant would play each level of the game one time and the results would be recorded (see key metrics in Table 6.1). By recording these results for each test over a two-week period, player progression can be tracked.

## **6.2 Limitations**

Currently, the world is facing an unprecedented global health crisis. Due to the Covid-19 virus, circumstances are abnormal and physical interaction is limited. I had to leave the Glasgow School of Art and return to the USA, so access to hardware was shut off. There is a responsibility to adhere to health guidelines. For these reasons, the first approach of medium-scale in-person testing is not possible. Having multiple people use one headset poses serious health risks in current times. Instead, the second approach and a modified version of the third approach was used.

Testing in these circumstances is difficult, and so evaluation will not be as extensive as ideally desired. Use of the software is limited to those who own personal Quest headsets. Distribution of the game and collection of feedback will be “remote” rather than “in-person”. While this approach is useful, it may introduce certain biases and errors. Feedback is entirely self-reporting, so is likely biased to those with an interest in hand tracking technology and ASL. However, this could also have the advantage that the users have knowledge of systems and can offer constructive criticism.

Regardless of limitations due to current health guidelines, the evaluation conducted is a solid reflection of practical and analytical knowledge. The processes taken mirror those that would have been used in normal circumstances. Data has been gathered and processed in accordance with standard academic practice. Results found using these practices are indicative of how results would look with further study. Furthermore, extended testing would be easily produced as conditions return to normal.



## 6.3 Feedback from Online Users

### 6.3.1 SideQuest

The game was posted to SideQuest on July 10th, 2020 along with a video tutorial and link to a SUS survey. As of July 30, the game has been downloaded over 300 times. Launch numbers are not reported, so it is uncertain how many downloaders have played the game. Over two weeks, the game received five ratings. Four users rated the game five-stars (out of five), and one rated the game four-stars. All five-star ratings noted the usefulness of the game and its potential for application in educational settings. One user wrote *“Fantastic application. Hopeful to see a full ASL version in the future. I used to practice a little ASL fingerspelling but haven't done it in maybe 10 years. After only around 30 mins in and I can already do all the signs again and am successfully spelling words at a decent pace.”* The four-star rating notes the inability of the hand tracking to accurately represent certain signs, specifically ‘R’ and ‘N’. Overall, the average rating of the app is 4.8 out of 5.

### 6.3.2 Reddit

A promotional post was submitted to the Oculus Quest forum on Reddit. The post garnered a score of 487 points, with a 99% upvote ratio. The post received 22 positive comments. 7 of these specifically mentioned the game as a useful tool for ASL acquisition. One commenter posted *“Holy crap this is exciting! When I heard about hand tracking, I instantly thought about how I wanted to learn American Sign language!”*, while another wrote *“Not only will this help spread sign language it's an amazing learning tool.”* The remaining fifteen positive comments noted that the game was cool, a good idea, or looked enjoyable. Other comments on the post were questions about the gameplay or how to access the game. One user offered a suggestion that when the player has trouble producing a certain sign, the game should show them how to produce that sign. None of the comments were critical or negative.

### 6.3.3 Twitter

On July 18th, a user responded to the game's Twitter link with feedback and a video of their gameplay. This user describes themselves as deaf. The user reported that they enjoyed the sign calibration system and that *“it should be standard for any ASL based app.”* The user also noted the difficulty of

producing the signs for ‘R’, ‘M’, and ‘N’. Finally, the user reported that they struggled with the menu implementation and would prefer a physical touch-based menu system.

The footage uploaded by this user gives valuable insight to the system’s compatibility with native ASL use. The user was able to complete two levels, producing all letters requested. The user produced 23 correct signs in 23.1 seconds on level one, and 40 correct signs in 45.1 seconds on level two. This produces the following statistics, seen in Table 6.1.

**Table 6.1: Resulting Statistics of a Play Session from a Native ASL User**

<b>Stat</b>	<b>Level 1</b>	<b>Level 2</b>	<b>Average</b>
Correct letters produced	23	40	31.5
Correct letters expected	23	40	31.5
Incorrect letters detected	27	63	45
Time taken per level	23.1	45.1	34.1
Production Score	1.0000	1.0000	1.0000
Accuracy	0.4600	0.3883	0.4242
Ratio	1.1739	1.5750	1.3745
Production Rate	1.004	1.128	1.066

Results show the user produced signs at a rate of roughly one correct sign per second. This is double the estimated natural sign production rate of 0.5 seconds per sign. This suggests that the user had to slow down production for the game to recognize each sign properly. Nonetheless, it is a positive finding that this user was able to successfully complete two levels at a relatively rapid pace, on what is assumed to be the first play session of the game. This indicates that the technology is approaching a state in which it could be used for natural sign language detection.

### **6.3.4 Online Survey**

Online postings of the game were accompanied with a link to a SUS survey and an open response questionnaire. The SUS survey consists of 10 questions to determine the usability of the game by players'

opinions. Each question has five possible responses ranging from “strongly disagree” (-2) to “strongly agree” (2), with “neither agree nor disagree” at centre (0).

Response to the survey was voluntary. To access the survey, users must go to my personal website. A reminder to take the survey was placed in the game, and a link was provided in posts on every platform. Despite achieving 300 downloads and a 4.8-star rating, only two users responded and completed the survey in full – a slightly disappointing response rate. While this sample size is statistically insignificant, the consistency between answers, shown in Table 6.2, suggests how results from further testing may appear.

**Table 6.2: SUS Survey Responses**

<b>Question</b>	<b>User 1</b>	<b>User 2</b>
I think that I would like to use this system frequently	2	1
I found the system unnecessarily complex	-1	-1
I thought the system was easy to use	1	2
I think that I would need the support of a technical person to be able to use this system	-1	-2
I found the various functions in this system were well integrated	1	1
I thought there was too much inconsistency in this system	-2	-1
I would imagine that most people would learn to use this system very quickly	2	2
I found the system very cumbersome to use	-1	-1
I felt very confident using the system	1	1
I needed to learn a lot of things before I could get going with this system	-1	-2

As these results show, both users found: the game easy to use; the functions in the game were well integrated; they would like to play the game frequently; and they felt confident using the system. The users both disagreed: the game was unnecessarily complex; they would need assistance to play the game; there was too much inconsistency in the game; the game was cumbersome to play they needed to learn a lot of things before playing the game. The last statement is noteworthy, as it suggests that the players were able to learn the alphabet from playing the game and did not feel that they needed previous ASL knowledge to play the game.

The users also responded to the questionnaire which consisted of 9 questions garnering feedback about the game and relevant information about the responders. Responders indicated that they had at least one year's experience using virtual reality technologies, and neither had experience with ASL. In conjunction with results from the SUS, this could indicate the game can be effective for those with no prior ASL knowledge but is most effective for those who are familiar with VR. Full questionnaire responses are in Table A.1.

These responses offer some vital insights to the players' experience. One user commented on letters which the system struggles to detect, namely T, K, P, and N. K and P use the same handshapes in different orientations, which suggests something about the handshape gives either the hand tracking or the recognition system some trouble. T and N both involve tucking the thumb under a finger, which occludes the thumb from view.

One responder reported a bug that the system would not recognize the last letter of a word. This bug has not been reproducible, so further investigation is needed.

Both users gave suggestions for improving the game. One suggested that the hand models be shown from multiple angles, which would give the player a better understanding of the sign and how to shape their hand correctly. Another suggested that the player be prompted with clues during fingerspelling gameplay, which parrots a comment from Reddit suggesting that the player is given a clue if struggling to produce a sign.

Overall responses were positive. The users enjoyed the game and were able to use it without the need for technical support. Most importantly the results suggest that the game shows potential for teaching ASL to new learners.

## **6.4 In-person testing**

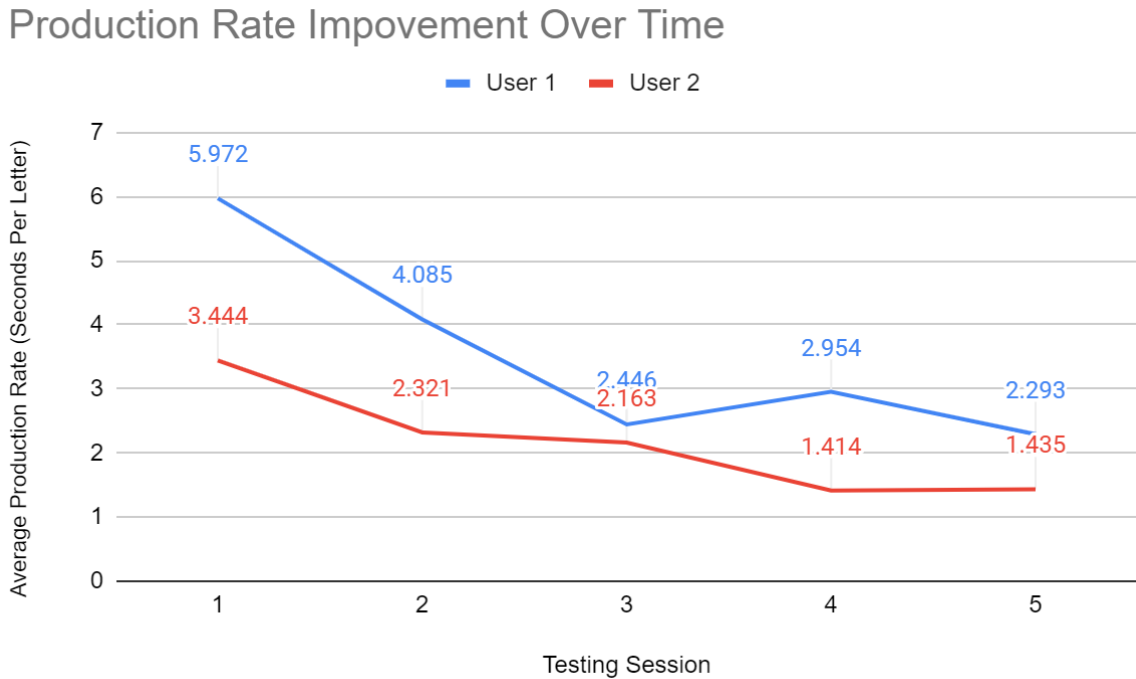
A small scale in-person test was conducted to measure the game's pedagogical efficacy. Two participants with no prior knowledge of ASL played the game five times each to measure how they improve over prolonged play. Each session consisted of practicing the alphabet and playing all 5 levels of the game for which statistics were tracked. As levels are randomly generated, lengths of tests will vary. The key statistics, (i) the ratio of incorrect letters to correct letters (accuracy ratio), and (ii) the production rate, are less dependent on level length.

As Table A.2 shows, Participant One set a baseline of an average production rate of 5.972 seconds per letter, and an average accuracy ratio of 2.907 incorrect letters detected for each correct letter produced. These stats saw a dramatic improvement over the next four tests, finishing with a production rate of 2.293 seconds per letter, and an accuracy ratio of 1.494. This is an improvement of 62% in production rate, and 49% in accuracy ratio.

Participant Two set a baseline of an average production rate of 3.444 seconds per letter, and an average accuracy ratio of 1.451 incorrect letters detected for each correct letter produced. These stats saw a steady improvement over the next four tests, achieving their lowest production rate of 1.414 seconds per letter, and an accuracy ratio of 1.1673. This is an improvement of 58% in production rate, and 20% in accuracy ratio.

By observing the players in real life, we uncovered some strengths and weaknesses of the system. The sign recognition system excelled with “short” hand forms, such as A, S, and E, when the fingers are folded, and the hand is closed almost to a fist. The system handled some tall signs – like B, L – with no finger occlusion well. Signs that include fingers overlapping such as M, N, and T, and signs such as K and that require the hand to block the view of the fingers proved a little more difficult to the system. The dynamic signs were not difficult to recognize when produced naturally.

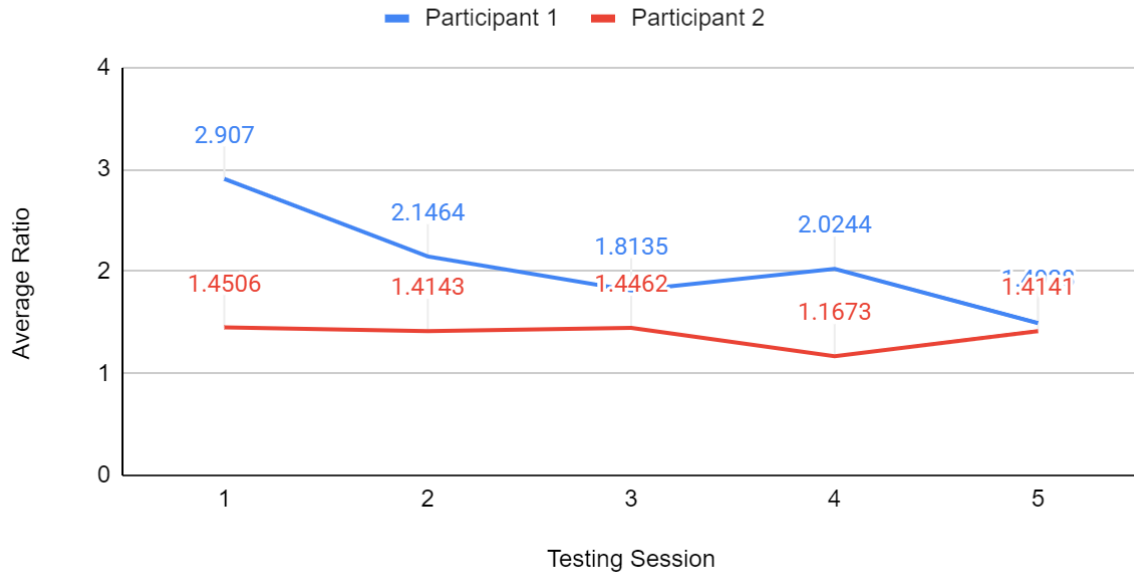
Graph 6.1: Production Rate Improvement Over Time



Such a small sample size, in terms of participants and data points/testing sessions, means any inferences from these tests are not statistically significant. Regardless, trends that arise can be noted, as they may suggest how results from further testing may appear. The graphs demonstrate the rapid improvement in production rate by both participants before plateauing towards the end. This may occur as the first few sessions are spent familiarising the player with the ASL alphabet, and the mechanics of the game. Learning these key aspects brings dramatic improvements in time. Once the player has memorized the alphabet and is familiar with the mechanics, production rate is limited by the player's ability to produce signs consistently and anticipate following letters.

**Graph 6.2: Average Ratio of Incorrect Letters Detected per Correct Letter Produced**

## Average Ratio of Incorrect Letters Detected per Correct Letter Produced



The graph of accuracy ratio shows participant one's improvement over time, similarly to their production rate, while participant two stayed consistent throughout testing. The two participants finished with similar figures. In fact, this figure was also that of the native ASL user from Twitter. This suggests that the sign recognition system has a natural error margin which will always be present. If a game or app were using this sign recognition system as a typing mechanism, there would be at least one typo for each correct letter produced. This would be frustrating, so this system could not currently be used for keyboard-free typing.

## 6.5 Conclusion

The combination of evaluative evidence suggests the game's potential as a learning tool for ASL fingerspelling. The improvement of the in-person participant's accuracy ratio and production rate over only 5 sessions of play indicates the game has some pedagogical value. The response from online users was overwhelmingly positive, which shows the interest from the public for the application of the technology towards education. The technology recognizes most letters of the alphabet, and while dynamic letters are not detected "truly", shortcuts allow them to be recognized. Letters in which fingers occlude

each other give the system some trouble but can be detected with adjustments to production form. Having to make adjustments is not ideal but is similar to speech recognition systems which are able to recognize American accents yet need adjustments for, say, Indian accents.

Results from an online native ASL user show the system is still not ready for natural recognition, but reaching this goal draws closer as the technology improves. This game does promote focus on canonical hand forms which will foster rapid improvement in the short term, just as Geer and Keane found. The tool may not be perfect for long term development of ASL fingerspelling skills but provides new learners a base of knowledge and a fun practice tool.



## **7 Conclusion**

### **7.1 Discussion**

#### **7.1.1 Achievements**

The project has created an “ASL Fingerspeller” game that is on the cutting edge of hand tracking in VR games. Hand tracking built into VR is still in its early stages, so ASL Fingerspeller is in the first wave of hand tracking games released to the public. Furthermore, it is one of the first games released to use a gesture recognition system. The development of the gesture recognition system was dependent on the PC to VR data transfer system provided by Oculus Link. Hand tracking support was added to Oculus Link in February 2020, so development of this game has only been possible for several months. It will be exciting to see what else can be created using these technologies as they become more accessible to developers.

An important feature of the ASL game is that it is customizable and adaptive to the player. This is especially important for any gesture recognition system, as users will produce gestures differently from each other. The sign calibration feature of ASL Fingerspeller brings an extra element of accessibility to the game. Calibrating each sign to the user’s hands makes the game a little bit easier, and a little bit more enjoyable.

Another important feature of ASL Fingerspeller is that it is socially useful. The app provides an interesting first look at American Sign Language, showing off one aspect of the language which is easy for English speakers to understand. Based on the response from online users, the game is fun and interesting for VR enthusiasts who have had little experience with ASL. While the game will not provide them all the knowledge and ability needed for sign language communication, it will get them started on their journey, and provide a fun practice tool for one part of the language. Additionally, results from in-person testing suggest that the game will help them improve their fingerspelling ability. Most importantly, the fun of the game will hopefully spark an interest in ASL, encouraging the player to take further lessons.

### **7.1.2 Limitations**

As leading edge as the ASL Fingerspeller game is, there are limitations that come with the technology. Hand tracking technology has improved since the release of the LMC yet even with four cameras, the Quest still struggles to track fingers under occlusion. Because of this, signs such as letter M, N, R, and T are not properly represented by the virtual hands. Dynamic signs are not recognized based on their motion, but rather by fitting one pose in a sequence. This is not a true representation of these signs, so another form of detection system is needed to recognize and discern between dynamic gestures.

Footage of gameplay from a native ASL user showed that the system was not ready to recognize fingerspelling at the fast rate of a “natural” signer. The system seems to have a natural error margin which makes it unusable as a typing mechanism. Unfortunately, response rate was low and due to Covid-19 health guidelines, testing was limited. Inferences from the evaluation are not statistically significant. Further testing is needed to truly test the pedagogical efficacy of ASL Fingerspeller.

### **7.1.3 ASL Fingerspeller**

ASL Fingerspeller is a fun and functional tool, but there is room for improvement. Further work on this project would include more development of the data management in the game. Saving data to the local drive proved to be quite tricky on the Quest. If this issue can be solved, persistent data systems can be implemented. This would include saving player profiles so multiple people can take turns using the same Quest, saving signs so that calibration is needed only once, and saving high scores so players can track their progress over time.

Other improvements could include the development of a dynamic sign recognition system. Such a system could be repurposed for sign language vocabulary, as many ASL signs require both a hand shape and a specific motion. In accordance with previous research, the inclusion of a machine learning algorithm could prove useful in such a system. Finally, fingerspelling comprehension proves to be a greater challenge to L2 ASL learners than fingerspelling production. A game mode in which players watch an avatar fingerspell and identify the word spelled could be a helpful tool in improving fingerspelling comprehension. Such a game mode could be networked multiplayer, prompting one player

to spell a word and the other to identify it. Multiplayer versions of ASL games could also prove to be a useful tool in L2 ASL education.

#### **7.1.4 Other Uses**

The development methods used in ASL Fingerspeller could be of use for other areas in game or app development. Gesture recognition is applicable in a wide variety of games, ranging from simple games such as rock paper scissors to magic-based RPGs to dance/rhythm games. One such educational implementation in rhythm gaming could be musical instrument instruction. A hand tracking game using “air-guitars” or “air-trumpets” could provide new learners with a new and exciting way to practice the motions needed to play such instruments.

Another area of use might be in augmented or mixed reality applications with no hand-held peripherals. Gestures could prove to be a natural method of interaction, used for purposes such as volume control, menu navigation, text or numerical input, environment manipulation, etc. Environment manipulation could be used for interior design focused apps, which would allow users to move, resize, or rotate superimposed pieces of furniture. Gesture recognition opens a whole new world of possibilities and human-computer interactions.

## **7.2 Conclusion**

This project has provided answers to the questions posed earlier. Optical hand tracking in virtual reality in its current state does provide a platform for ASL education. The development of ASL Fingerspeller and evaluation of the game highlighted the strengths of the Quest’s hand tracking system. The game was able to successfully teach the ASL alphabet to participants in the research and improved their fingerspelling ability over a short period of time. The system excelled with simple signs that resemble closed fists or pointing gestures but struggles with more complex signs that require intricate finger positions. Still, users were able to produce all signs in a way that the system could recognize. Investigation into how ASL users would interpret the production of signs in this manner could give further insight into the strengths and weaknesses of ASL Fingerspeller.

Hand tracking is sufficient for many static signs and using shortcuts allows dynamic signs to be recognized as well. The limitations of the system include loss of tracking due to occlusion. Fingers or hands crossing prevents accurate tracking and disrupts the virtual representation of the user's hands. This limitation prevents certain signs from being represented in VR, and as such the platform is not ready for intensive ASL education. Solutions for resolving occlusion could include external cameras to provide more optical coverage. Additionally, the development of a dynamic gesture recognition system could broaden the range of possibilities for the use of ASL in VR. Despite these shortcomings, the platform is ready for ASL education in small lessons, such as the alphabet, numbers, and some one-handed signs. The platform could be creatively exploited for a beginner's approach to ASL.

However the technology is used, it is clear to see that it is fun. Hand tracking in virtual reality is a brand new and exciting way to interact with a computer system. The possibilities of its application are extensive. Applications in the future can take advantage of gesture recognition, physically based interactions, or hands as a controller to produce creative and entertaining new types of games. Development of the technology has been rapid, extending accessibility in the past few years. Greater accessibility will lead to developers finding more creative uses for the technology, attracting more users to the platform. Hopefully, more development will lead to more games which will lead to more players learning, doing socially useful activity, and ideally having some fun along the way.

## References

- Bouزيد, Y. *et al.* (2016) 'Using educational games for sign language learning - A signwriting learning game: Case study', *Educational Technology and Society*, 19(1), pp. 129–141.
- Cheok, M. J., Omar, Z. and Jaward, M. H. (2017) 'A review of hand gesture and sign language recognition techniques', *International Journal of Machine Learning and Cybernetics*. Springer Berlin Heidelberg, 10(1), pp. 131–153. doi: 10.1007/s13042-017-0705-5.
- Chuan, C. H., Regina, E. and Guardino, C. (2014) 'American sign language recognition using leap motion sensor', *Proceedings - 2014 13th International Conference on Machine Learning and Applications, ICMLA 2014*. IEEE, pp. 541–544. doi: 10.1109/ICMLA.2014.110.
- Clark, A. and Moodley, D. (2016) 'A system for a hand gesture-manipulated virtual reality environment', *ACM International Conference Proceeding Series*, 26-28-Sept. doi: 10.1145/2987491.2987511.
- Fok, K. Y. *et al.* (2015) 'A Real-Time ASL Recognition System Using Leap Motion Sensors', *Proceedings - 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2015*. IEEE, pp. 411–414. doi: 10.1109/CyberC.2015.81.
- Geer, L. C. (2016) 'Teaching ASL fingerspelling to second-language learners: Explicit versus implicit phonetic training (The University of Texas at Austin, 2016)', *Sign Language and Linguistics (Online)*, 19(2), pp. 280–284. doi: 10.1075/sll.19.2.07gee.
- Geer, L. C. and Keane, J. (2018) 'Improving ASL fingerspelling comprehension in L2 learners with explicit phonetic instruction', *Language Teaching Research*, 22(4), pp. 439–457. doi: 10.1177/1362168816686988.
- Hung, H. T. *et al.* (2018) 'A scoping review of research on digital game-based language learning', *Computers and Education*. Elsevier, 126(May), pp. 89–104. doi: 10.1016/j.compedu.2018.07.001.
- Keane, J. and Geer, L. C. (2014) 'Understanding fingerspelling perception Results ', 22(6), p. 3278.

Mapari, R. B. and Kharat, G. (2016) ‘American static signs recognition using leap motion sensor’, *ACM International Conference Proceeding Series*, 04-05-Marc, pp. 1–5. doi: 10.1145/2905055.2905125.

NIH (2019) ‘American Sign Language’. Bethesda: NIDCD. doi: 10.4135/9781412957403.n31.

Quinto-Pozos, D. (2011) ‘Teaching American sign language to hearing adult learners’, *Annual Review of Applied Linguistics*, 31, pp. 137–158. doi: 10.1017/S0267190511000195.

Reinhardt, J. and Sykes, J. M. (2012) ‘Conceptualizing Digital Game-Mediated L2 Learning and Pedagogy: Game-Enhanced and Game-Based Research and Practice’, *Digital Games in Language Learning and Teaching*, pp. 32–49. doi: 10.1057/9781137005267\_3.

Snoddon, K. (2017) ‘Uncovering translingual practices in teaching parents classical ASL varieties’, *International Journal of Multilingualism*. Taylor & Francis, 14(3), pp. 303–316. doi: 10.1080/14790718.2017.1315812.

Su, S. A. and Furuta, R. K. (1998) ‘VRML-based representations of ASL fingerspelling on the World-Wide Web’, *Annual ACM Conference on Assistive Technologies, Proceedings*, pp. 43–45. doi: 10.1145/274497.274506.

Sykes, J. M. (2018) ‘Digital games and language teaching and learning’, *Foreign Language Annals*, 51(1), pp. 219–224. doi: 10.1111/flan.12325.

Ultraleap (2019) *Leap Motion Controller*. Available at: [ultraleap.com/product/leap-motion-controller/](http://ultraleap.com/product/leap-motion-controller/).

## Referenced Software and Games

- Blender
  - 2020. *Blender v2.83*. Blender Foundation. [software] Available at: <<https://www.blender.org/>>
- Kingdom Hearts
  - Square Enix, 2002. *Kingdom Hearts*. Disney Interactive Studios. [video game] Available at: <<https://www.kingdomhearts.com/>>
- Minecraft
  - Mojang Studios, 2011. *Minecraft*. Xbox Game Studios. [video game] Available at: <<https://www.minecraft.net/>>
- Mixamo
  - 2020. *Mixamo*. Adobe Systems Incorporated. [software] Available at: <<https://www.mixamo.com/#/>>
- Oculus Integration
  - 2020. *Oculus Integration v16.0*. Facebook Technologies, LLC. [software] Available at: <<https://support.oculus.com/release-notes/>> and <<https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022#version-current>>
- Pokémon Go
  - Niantic Inc., 2016. *Pokémon Go*. The Pokémon Company. [video game] Available at: <<https://www.pokemongo.com/>>
- Portal 2
  - Valve Corporation, 2011. *Portal 2*. Valve Corporation. [video game] Available at: <<https://www.thinkwithportals.com/>>
- Skyrim
  - Bethesda Softworks LLC, 2011. *The Elder Scrolls V: Skyrim*. Zenimax Media. [video game] Available at: <<https://elderscrolls.bethesda.net/en/skyrim>>
- Unity
  - 2019. *Unity 2019.3.11f1*. Unity Technologies. [software] Available at: <<https://unity.com/>>
- Unreal
  - 2004. *Unreal*. Epic Games. [software] Available at: <<https://www.unrealengine.com/>>

## List of Third-Party Materials Used

Table R.1: Third Party Materials Used

Asset	Use	Sourced From
little_robot_sound_factory_Jingle_Win_Synth_00.mp3	Word complete sound effect in game	<a href="https://ZapSplat.com">ZapSplat.com</a>
sound_ex_machina_Button_Blip.mp3	Button press/click sound effect in game	<a href="https://ZapSplat.com">ZapSplat.com</a>
zapsplat_multimedia_alert_notification_glassy_high_pitched_short_positive_001_42348.mp3	Correct letter sound effect in game	<a href="https://ZapSplat.com">ZapSplat.com</a>
zapsplat_multimedia_game_sound_synth_digital_tone_beep_005_38537.mp3	Letter spawn sound effect in game	<a href="https://ZapSplat.com">ZapSplat.com</a>
zapsplat_multimedia_game_tone_harp_warm_positive_correct_win_002_50713.mp3	Level complete sound effect in game	<a href="https://ZapSplat.com">ZapSplat.com</a>
Ybot.fbx	Hand model to demonstrate signs in game	<a href="https://Mixamo.com">Mixamo.com</a>
Xbot.fbx	Hand model to demonstrate signs in game	<a href="https://Mixamo.com">Mixamo.com</a>
Oculus Integration	Player hand models, interaction methods, hand tracking capabilities, VR integration for game development	<a href="https://Unity Asset Store">Unity Asset Store</a>
Image of Quest headset	Image in report	<a href="https://Amazon.com">Amazon.com</a>
Image of ASL fingerspelling alphabet	Image in report	<a href="https://NIDCD.NIH.gov">NIDCD.NIH.gov</a>
Image of Quest hand tracking	Image in report	<a href="https://Oculus.com">Oculus.com</a>
Image of Vive with LMC attached	Image in report	<a href="https://Leapmotion.com">Leapmotion.com</a>
Image of hand tracking under occlusion	Image in report	<a href="https://Developer.Oculus.com">Developer.Oculus.com</a>
Diagram of Oculus Integration bone ID legends	Images in report	<a href="https://Reddit.com">Reddit.com</a>
Diagram of close-Range Interaction	Image in report	<a href="https://Developer.Oculus.com">Developer.Oculus.com</a>
Diagram of Tap-to-click system developed by Oculus	Image in report	<a href="https://Developer.Oculus.com">Developer.Oculus.com</a>



## Appendix A. User Evaluations Results

Table A.1: Open Ended User Responses from Online Survey

Question	User 1	User 2
What did you like about the game?	<i>I've always wanted to learn ASL, and this was a super simple way to get started!</i>	<i>It is fun and easy to use</i>
What did you dislike about the game?	<i>There were a few letters that I simply could not get recognized, namely t, k, p, and n.</i>	
What worked well in the game?	<i>The recognition system works pretty well, except for the few letters I listed above...</i>	<i>I think it is effective to learn ASL</i>
What did not work well in the game?	<i>Occasionally, the game simply would not take the last letter of a word, no matter how hard I tried, though this was probably user error.</i>	
How would you improve the game?	<i>Sometimes, it was difficult to see exactly how each letter is formed; perhaps slowly turning around the example hands would help with that? On another note, it would help if for the practice, you could select a letter individually to highlight and practice.</i>	<i>Maybe with some clues in the gameplay</i>
Did you learn anything by playing this game? Do you think this game would be effective in improving your fingerspelling ability?	<i>I learned more ASL in half an hour than I did in the last ten years, so well done! Very effective!</i>	Yes
How much experience do you have with Virtual Reality VR? How much experience do you have with hand tracking in VR?	<i>In terms of my experience with VR, I consider myself a veteran at this point: I got my start with the Oculus Rift years ago and have recently upgraded to a Quest. I recently clocked in 2,000 hours on Steam and have some limited experience with Oculus Quest app development in Unity. I've also had Hand Tracking on when available since the day it was added to the experimental features tab and have produced some of my own hand-tracked content via Unity.</i>	<i>1 year of experience with VR and a few months with hand tracking</i>

<p>Do you have any previous ASL knowledge? Please share your level of proficiency</p>	<p><i>Absolutely none! I just started with it when I loaded up your app!</i></p>	<p>No</p>
<p>Please use this box to say anything else you would like to tell me about ASL Fingerspeller.</p>	<p><i>Very interesting concept! I think as hand tracking improves, the few hiccups I had will mostly get smoothed out, so very excited!</i></p>	

**Table A.2: In-person User Testing Results**

Legend

- CLP – Number of correct letters produced
- CLE – Number of correct letters expected
- ILP – Number of incorrect letters expected
- Time – Time taken to complete the level in seconds
- Production Score – Number of correct letters produced divided by number of correct letters expected
- Accuracy – Number of incorrect letters produced divided by the sum of incorrect letters produced and correct letters produced
- Ratio – Number of incorrect letters produced divided by the number of correct letters produced
- Production rate – Time taken to complete the level divided by correct letters produced

Test 1		CLP	CLE	ILP	Time	Production Score	Accuracy	Ratio	Production Rate
Level 1	User 1	23	24	73	235.6	0.9583	0.2396	3.1739	10.243
	User 2	24	26	51	132.6	0.9231	0.3200	2.1250	5.525
Level 2	User 1	41	42	126	199.1	0.9762	0.2455	3.0732	4.856
	User 2	30	32	52	110.2	0.9375	0.3659	1.7333	3.673
Level 3	User 1	34	34	100	218.2	1.0000	0.2537	2.9412	6.418
	User 2	32	32	28	63.5	1.0000	0.5333	0.8750	1.984
Level 4	User 1	41	43	98	226.2	0.9535	0.2950	2.3902	5.517
	User 2	46	46	48	101.2	1.0000	0.4894	1.0435	2.200
Level 5	User 1	46	46	136	130	1.0000	0.2527	2.9565	2.826
	User 2	42	42	62	161.2	1.0000	0.4038	1.4762	3.838
Average	User 1	37	37.8	106.6	201.82	0.9776	0.2573	2.9070	5.972
	User 2	34.8	35.6	48.2	113.74	0.9721	0.4225	1.4506	3.444

Test 2		CLP	CLE	ILP	Time	Production Score	Accuracy	Ratio	Production Rate
Level 1	User 1	28	28	40	69.3	1.0000	0.4118	1.4286	2.475
	User 2	25	26	29	73.3	0.9615	0.4630	1.1600	2.932
Level 2	User 1	28	32	71	96.9	0.8750	0.2828	2.5357	3.461
	User 2	36	37	43	76.8	0.9730	0.4557	1.1944	2.133
Level 3	User 1	32	33	55	107.7	0.9697	0.3678	1.7188	3.366

	User 2	36	37	49	92.3	0.9730	0.4235	1.3611	2.564
	User 1	39	43	94	286.2	0.9070	0.2932	2.4103	7.338
Level 4	User 2	42	42	62	77.1	1.0000	0.4038	1.4762	1.836
	User 1	36	39	95	136.2	0.9231	0.2748	2.6389	3.783
Level 5	User 2	50	50	94	107.1	1.0000	0.3472	1.8800	2.142
	User 1	32.6	35	71	139.26	0.9350	0.3261	2.1464	4.085
Average	User 2	37.8	38.4	55.4	85.32	0.9815	0.4187	1.4143	2.321

Test 3		CLP	CLE	ILP	Time	Producti on Score	Accuracy	Ratio	Producti on Rate
	User 1	19	22	33	64.8	0.8636	0.3654	1.7368	3.411
Level 1	User 2	26	26	33	68.8	1.0000	0.4407	1.2692	2.646
	User 1	37	37	80	65.3	1.0000	0.3162	2.1622	1.765
Level 2	User 2	36	36	71	82.1	1.0000	0.3364	1.9722	2.281
	User 1	31	33	66	77	0.9394	0.3196	2.1290	2.484
Level 3	User 2	30	30	34	52.2	1.0000	0.4688	1.1333	1.740
	User 1	39	40	69	122.9	0.9750	0.3611	1.7692	3.151
Level 4	User 2	44	44	67	92.7	1.0000	0.3964	1.5227	2.107
	User 1	37	37	47	52.5	1.0000	0.4405	1.2703	1.419
Level 5	User 2	45	45	60	91.9	1.0000	0.4286	1.3333	2.042
	User 1	32.6	33.8	59	76.5	0.9556	0.3606	1.8135	2.446
Average	User 2	36.2	36.2	53	77.54	1.0000	0.4142	1.4462	2.163

Test 4		CLP	CLE	ILP	Time	Producti on Score	Accuracy	Ratio	Producti on Rate
	User 1	22	23	33	51.4	0.9565	0.4000	1.5000	2.336
Level 1	User 2	30	30	36	45.8	1.0000	0.4545	1.2000	1.527
	User 1	42	43	89	111	0.9767	0.3206	2.1190	2.643
Level 2	User 2	47	47	54	52.4	1.0000	0.4653	1.1489	1.115
	User 1	32	33	74	125.3	0.9697	0.3019	2.3125	3.916
Level 3	User 2	33	33	44	56.2	1.0000	0.4286	1.3333	1.703

Level 4	User 1	42	42	88	102.4	1.0000	0.3231	2.0952	2.438
	User 2	36	36	35	49.3	1.0000	0.5070	0.9722	1.369
Level 5	User 1	42	44	88	144.4	0.9545	0.3231	2.0952	3.438
	User 2	44	44	52	59.7	1.0000	0.4583	1.1818	1.357
Average	User 1	36	37	74.4	106.9	0.9715	0.3337	2.0244	2.954
	User 2	38	38	44.2	52.68	1.0000	0.4628	1.1673	1.414

Test 5		CLP	CLE	ILP	Time	Producti on Score	Accuracy	Ratio	Producti on Rate
Level 1	User 1	27	27	21	49.9	1.0000	0.5625	0.7778	1.848
	User 2	22	22	18	29.5	1.0000	0.5500	0.8182	1.341
Level 2	User 1	38	39	73	111.3	0.9744	0.3423	1.9211	2.929
	User 2	42	42	54	54.6	1.0000	0.4375	1.2857	1.300
Level 3	User 1	35	35	63	86.9	1.0000	0.3571	1.8000	2.483
	User 2	31	31	49	44.4	1.0000	0.3875	1.5806	1.432
Level 4	User 1	37	37	50	84.9	1.0000	0.4253	1.3514	2.295
	User 2	42	42	56	54.4	1.0000	0.4286	1.3333	1.295
Level 5	User 1	44	44	71	84.1	1.0000	0.3826	1.6136	1.911
	User 2	38	38	78	68.6	1.0000	0.3276	2.0526	1.805
Average	User 1	36.2	36.4	55.6	83.42	0.9949	0.4140	1.4928	2.293
	User 2	35	35	51	50.3	1.0000	0.4262	1.4141	1.435

## Appendix B. Code

```
public void CheckLetter(string letter)//check if letter just produced is the right letter
{
    if(spellCheck)
    {
        if(letter[0].Equals(toSpell[letterIndex]))//if first char of string associated with produced sign matches expected letter
        {
            AddLetter(letter);//add the letter to the spelled word
        }
        else
        {
            ilp++;
        }
    }
}

void AddLetter(string letter)//check if word is fully spelled
{
    audioManager.Play("Correct Letter");
    spelled += letter;//add the letter to the spelled word
    letterIndex++;//go to the next letter

    clp++;

    if(spelled.Equals(toSpell))//if spelled word matches word to spell
    {
        NextWord();//go to next word
    }
}

public void NextWord();//go to next word
{
    audioManager.Play("Next Word");

    wordTimer = 0;

    spelled = null;//reset word to spell
    letterIndex = 0;//reset letter, go to start of word

    wordIndex++;//go to next word

    if(wordIndex == dictionary.Count)//if that was the last word
    {
        OnLevelComplete.Invoke();//invoke end of level event
        audioManager.Play("Level Complete");//play the end level sound
        levelActive = false;//turn off level active

        if(ilp < highScores[levelIndex].x)//if fewer incorrect letters detected
        {
            highScores[levelIndex].x = ilp;
        }
        if(levelTimer < highScores[levelIndex].y)//if less time taken
        {
            highScores[levelIndex].y = levelTimer;
        }
    }

    ToSpellUpdate();//update word to spell
}
```

Figure B.1: Spellchecker Code

```

[System.Serializable]
public class Sign
{
    public string name;//name of the sign
    public List<Vector3> fingerPositionalData;//list hold the positional data for each
bone in the hand
    public Quaternion rootRotation;//holds the rotational data of the root of the hand
(the wrist)

    public UnityEvent OnDetect;//event to execute when the sign is recognized

    public Sign DeepCopySign(Sign oldSign)
    {
        Sign newSign = new Sign();

        newSign.name = oldSign.name;
        newSign.fingerPositionalData = oldSign.fingerPositionalData;
        newSign.rootRotation = oldSign.rootRotation;
        newSign.OnDetect = oldSign.OnDetect;

        return newSign;
    }
}

```

**Figure B.2: Code Implementation of a "Sign"**

```

public void Overwrite()//function for overwriting the finger data of a sign
{
    Sign sign = activeSign;

    List<Vector3> positionalData = new List<Vector3>();//new list for positional d
ata of bones
    foreach(var bone in activeHand.Bones)//run through all bones in the hand
    {
        positionalData.Add(activeHand.transform.InverseTransformPoint(bone.Transform.position));//store positional data
    }

    sign.fingerPositionalData = positionalData;//overwrite the positional data with the new data
    sign.rootRotation = activeHand.Bones[1].Transform.rotation;//overwrite the rotation data with the new (current) rotation

    Debug.Log("Active Profile (index): " + profiles[profileIndex].name);
    Debug.Log("Active Profile (profile): " + activeProfile.name);

    Debug.Log("Active sign (index): " + profiles[profileIndex].rightHandSigns[signIndex].name);
    Debug.Log("Active sign (sign): " + activeSign.name);

    //activeSign = sign;//overwrite the active sign

    if(isLeft)
    {
        profiles[profileIndex].leftHandSigns[signIndex] = sign;
    }
    else
    {
        profiles[profileIndex].rightHandSigns[signIndex] = sign;
    }
}

```

**Figure B.3: Overwrite Function used in Calibration**

```

Sign Recognize()//function that checks current hand position
{
    Sign currentSign = new Sign();//get current sign
    float currentMin = Mathf.Infinity;//set minimum distance from listed sign

    for(int j = 1; j < activeList.Count; j++)//go through list of saved signs to compare
    {
        float sumDistance = 0;//how far away the current sign is from listed signs
        bool isDiscarded = false;//if the sign doesn't match listed signs

        //compare the current sign to listed signs
        for(int i = 0; i < activeHand.Bones.Count; i++)//iterate through bones
        {
            Vector3 currentPosition = activeHand.transform.InverseTransformPoint(activeHand.Bones[i].Transform.position);//get current positional data for all bones

            float distance = Vector3.Distance(currentPosition, activeList[j].fingerPositionalData[i]);//calculate distance from saved bone position

            if(distance > tolerance)//if current sign is not close enough, stop comparing, move on to next sign
            {
                isDiscarded = true;
                break;
            }
            sumDistance += distance;//add up the distance of all the bones
        }

        float rotDifference = Mathf.Abs(Quaternion.Dot(activeHand.Bones[1].Transform.rotation, activeList[j].rootRotation));
        if(spawner.spawn){rotDifference = 1f;}

        if(!isDiscarded && sumDistance < currentMin && rotDifference > (1f - angleTolerance))//if the sign is not discarded, close to listed sign, and the wrist rotation is close enough
        {
            currentMin = sumDistance;//set the minimal distance to how close we go to the listed sign against which we are comparing

            currentSign = activeList[j];//set the sign to the one against which we are comparing
        }
    }

    return currentSign;//return the sign
}

void Update()
{
    if(detect)
    {
        Sign currentSign = Recognize();//check if the sign being produced is a saved sign
        bool recognized = !currentSign.Equals(new Sign());//this tests that the Recognize() function returned a listed sign, instead of null

        if(recognized && !currentSign.Equals(previousSign))//check if current sign is different from last sign recognized (no repeats)
        {
            previousSign = currentSign;//store the current sign as previous so as to compare and not repeat

            currentSign.OnDetect.Invoke();//invoke OnDetect
        }
    }
}

```

**Figure B.4: Sign Detection/Recognition System**