Bachelor's thesis

Information and Communications Technologies

2020

Jere Ranta

A COMPARATIVE STUDY OF HAND TRACKING IN A VR ENVIRONMENT

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2020 | 40 pages

Jere Ranta

A COMPARATIVE STUDY OF HAND TRACKING IN A VR ENVIRONMENT

Virtual reality is a steadily growing industry with a multitude of applications from gaming to simulation and educational software. The common way to navigate these applications is with game controllers. Now through technological advances, it is also possible to use the player themselves as a controller instead of a mechanical input device.

The objective of this Bachelor's thesis was to compare and evaluate the advantages and disadvantages of using hand tracking in virtual reality compared to using controller inputs. In this thesis an existing virtual reality application is converted from a traditional virtual reality application into a mobile virtual reality application that uses hand tracking instead of controller inputs.

This thesis takes the reader through the development process for the hand tracking conversion and presents advantages and disadvantages for hand tracking over traditional input control while also examining and comparing different virtual reality technologies, with a focus on mobile virtual reality hardware and hand tracking technologies.

It was found that hand tracking is a feasible alternative to controllers in most cases but suffers greatly in applications where the player is expected to move around the virtual world. This thesis presents an alternative method for player movement while using hand tracking as well as the iterative process that produced the method.

Because of the relatively young age of hand tracking technologies, there are very few industry standards for hand tracking controls. The work on this thesis provides a possible solution to some of the problems that hand tracking brings and suggests a possible standard for hand tracking movement system in virtual reality.

KEYWORDS:

virtual reality, hand tracking, qian, oculus quest, wireless, mixed reality

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2020 | 40 sivua

Jere Ranta

VERTAILEVA TUTKIMUS KÄSIOHJAUKSEN KÄYTÖSTÄ VIRTUAALITODELLISUUS-YMPÄRISTÖSSÄ

Virtuaalitodellisuus (Virtual Reality) on nopeasti kasvava ala peliteollisuudessa, jossa nykyään keskitytään viihdepelaamisen lisäksi myös simulaatio- ja koulutusohjelmistoihin. Perinteisesti näitä ohjelmistoja kontrolloidaan peliohjaimilla, mutta uudet teknologiset saavutukset ovat mahdollistaneet myös pelaajan liikkeiden käyttämisen ohjaintoimintoihin.

Tämän opinnäytetyön tavoitteena oli verrata ja arvioida käsiohjauksen (hand tracking) hyötyjä sekä haittoja perinteisiin ohjausmetodeihin verrattuna. Tässä työssä muunnettiin olemassa oleva virtuaalitodellisuussovellus perinteisestä ohjaimilla pelattavasta langallisesta sovelluksesta langattomaksi käsiohjausta hyödyntäväksi sovellukseksi.

Tämä opinnäytetyö ohjaa lukijansa käsiohjausmuutosprosessin läpi, sekä vertaa tästä aiheutuvia hyötyjä ja haittoja. Työssä käydään lisäksi läpi erilaisia käsiohjaukseen liittyviä teknologioita, pääpainona langattomat teknologiat.

Useimmissa tapauksissa käsiohjauksen huomattiin toimivan mahdollisena vaihtoehtona perinteisille ohjaimille, mutta sen heikkoutena ovat sovellukset, joissa pelaajan on liikuttava virtuaalitodellisuuden sisällä. Opinnäytetyössä esitetään vaihtoehtoinen tapa liikkua virtuaaliympäristössä käyttäen käsiohjausta, sekä kehitysprosessi, jolla siihen päädyttiin.

Koska käsiohjaus on suhteessa hyvin tuore teknologia, on pelialla vain vähän aiheeseen liittyviä standardeja. Tässä opinnäytetyössä tuodaan esille joitain ratkaisuja ongelmiin, joita käsiohjauksessa esiintyy, sekä esitetään mahdollista standardia käsiohjauksella liikkumiseen virtuaaliympäristössä.

ASIASANAT:

Virtuaalitodellisuus, käsiohjaus, langattomuus, langaton, yhdistetty todellisuus

CONTENTS

LIST OF ABBREVIATIONS

AI	Artificial intelligence	[A1]
AR	Augmented reality	[A2]
FPS	Frames per second	[A3]
MR	Mixed reality	[A4]
SDK	Software development kit	[A5]
UI	User interface	[A6]
UX	User experience	[A7]
VR	Virtual reality	[A8]
XR	Mixed reality	[A4]

1 INTRODUCTION

Virtual reality (VR) has been growing steadily for years now not only in gaming, but in educational and simulation applications as well, and with it comes a multitude of new technologies and advancements. Traditionally virtual reality applications have been used with gaming controllers or VR controllers, but hand tracking technologies are making their mark in the industry. This bachelor's thesis studies hand tracking technologies and compares their differences and advantages to traditional controller systems from the point of user experience through play testing, relevant technologies and their potential in the future virtual reality environments.

This bachelor's thesis focuses mainly on two hand tracking technologies: LeapMotion by UltraLeap used with Realmax Qian Mixed Reality headset and Oculus Quest by Oculus. Hand tracking gives a unique unobstructed VR experience when combined with mobile VR platforms and the main objective of this bachelor's thesis is on the immersive experience that hand tracking provides when combined with wireless VR platforms. This thesis explores the different technologies used in Mixed Reality (XR) systems in general, as well as from the point of hand tracking. The focus is also on user experience and what challenges or benefits hand tracking brings to user experience and how this is reflected in requirements analysis for VR applications using hand tracking.

Another focus point of this thesis is a practical part where an existing VR application, Fire safety simulation by Turku Game Lab, is converted from traditional VR application with controller inputs into a mobile VR application that uses hand tracking instead of controllers. The thesis takes the reader through a summary of the development process and points out all the challenges of converting an existing system to a hand tracking system, as well as advantages and disadvantages hand tracking offers compared to the original controller version of the app.

The methods used in this Bachelor's thesis are a combination of research on the topic of hand tracking in virtual reality and an iterative development process of the fire safety simulation application. The findings are presented in the form of reflective thinking about the usability of hand tracking technologies and their future development in the field of virtual reality.

2 TECHNOLOGIES IN MIXED REALITY

The purpose in this chapter is to go through different technologies used with Mixed reality (XR, sometimes referred as MR) applications. Mixed reality is the combination of Virtual reality (VR) and Augmented reality (AR), that blends physical and virtual world in a way that creates a realistic environment [1]. The different levels of XR are further explored below by using a simplified representation of a RV continuum (picture 1). When looking at technologies in XR environment, there are a few different categories to consider:

Virtual reality, which is an immersive method of interacting with a computer simulated environment. VR technologies normally consist of a VR headset, tracking beacons for user position tracking, and a computer where the software is being run. The user feels like they are part of the virtual world and the virtual environment is built around the player. This is very resource demanding for the computer and because of this, while being the most common use of XR technology, it is used mostly in high-end video gaming for entertainment purposes [2].

Another category is Augmented reality, a form of virtual reality where the user sees a clear view of the real world where additional virtual elements or information are added to it. Most AR applications are used with smart phones and tablet computers, but it is also possible to use a transparent head mounted display to present the AR content [3]. These technologies are mostly used for, but not limited to, practical applications and AR gaming. Unlike in VR where the player is inside the virtual world, in AR virtual objects are brought to the real world using the device camera and then rendering objects on the screen, making the virtual objects appear to be part of the real world.

The third category is mobile XR platforms, which are wireless standalone headsets or glasses that can be used without any additional equipment and are not limited by wires or external computational needs. XR headsets combine VR and AR hardware into a single headset, but usually must sacrifice some computational power as a tradeoff for versatility. These technologies are only made possible recently with miniaturization of computing and battery technologies.

The last category for XR technologies is haptics; different technologies that gives the user actual sensory feedback in VR environments. Normally XR sensory feedback is

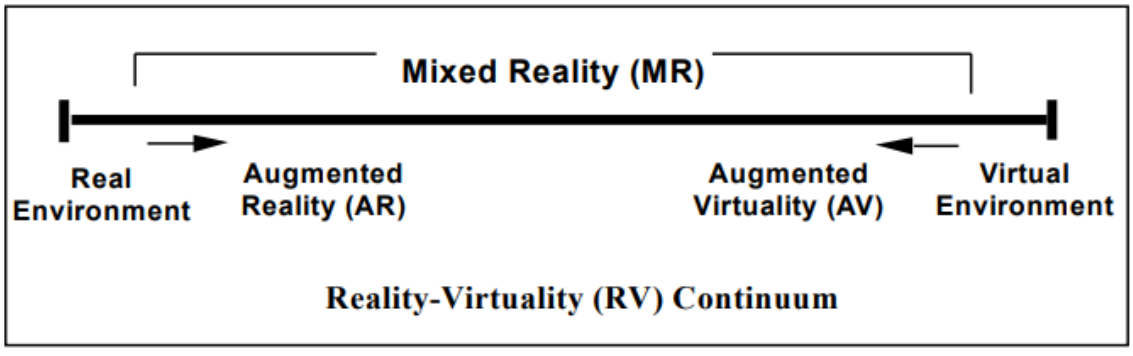
only limited to visual and auditory senses, but the purpose of haptic feedback is to bring non-verbal, touch-based communication to the virtual world. [4]

In XR, manipulating virtual environments are generally done in a few different ways. In most AR applications when using mobile devices such as a tablet computer, the inputs are handled with the device touch screen. In some cases, it is also controller by visually pointing the camera on objects, like reading QR-codes or using visual tracking. These methods offer very little feedback to the user and are generally not considered very immersive.

In VR, however, giving inputs to the application and receiving feedback is much more dynamic and immersive. All XR headsets come with some form of controllers for controlling various applications. Using controllers in VR gaming is the most common way of manipulating the virtual world.

The final method of inputs in XR is hand tracking or gesture controls. This is a new technology even in XR terms. This is a method of control that gives most of the benefits of using controllers, with next to no downsides that using controllers would bring.

When considering different input methods in mixed reality environments, it is possible to look at XR as having different levels of virtual environment immersion, which correspond to the input method in question. This can be represented with Reality-Virtuality continuum (picture 1).



Picture 1. Simplified RV continuum. [3]

The level of virtuality in an environment is increasing when moving through the RV continuum from left to right. In the left we have an environment consisting only of real objects that can be observed in person or through some kind of window or through some

kind of display. Next step in the RV continuum is augmented reality, in which we mostly use inputs through a mobile device, such a smartphone to manipulate the environment.

Next stop in the RV continuum is Augmented virtuality in which the environment is mostly if not completely virtual, having only some elements from the real physical world. These elements might simply be the control method of said environment. Hand tracking would fit in to this category as the environment would be completely virtual, but the input method is physical. Last part of the continuum is Virtual environment, where the whole environment is virtual including methods of user input. In this case using controllers would be considered as a virtual input method [3].

2.1 Hand Tracking

In XR environment capturing the movements of a player or objects is called Motion tracking. When some form of motion tracking is used to capture the movements and gestures of the player's hands, it is called Hand tracking. Gesture control in hand tracking is a way to control applications in XR environments through real life movements that gives the user higher levels of immersion through natural human interactions that would not be feasible with using controller inputs [5].

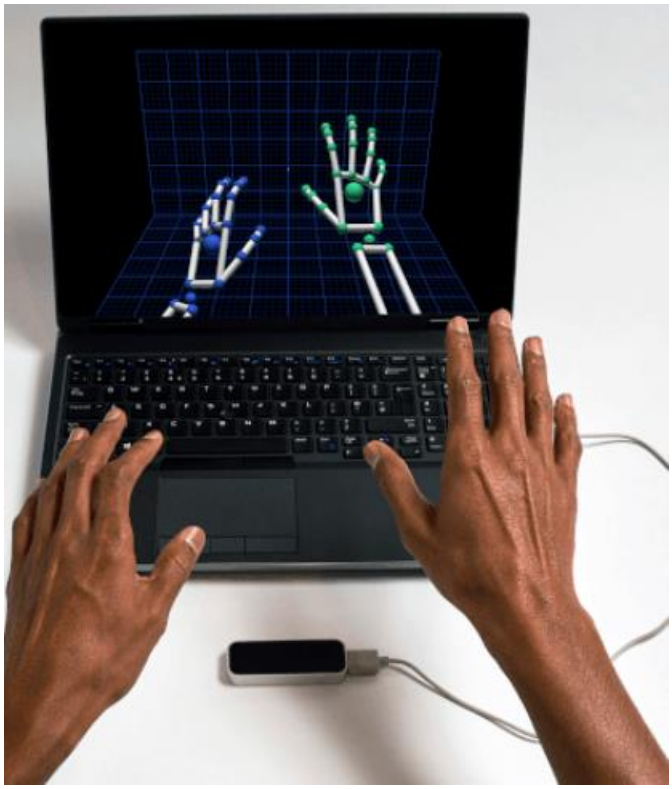
The technology behind hand tracking is nothing new on the hardware level; The only requirement is to have at least two cameras (picture 2) for depth perception and that is it. While most hand tracking hardware use more than two cameras for better depth perception and infrared cameras and infrared LEDs for better illumination outside of human visual spectrum, it is technically not a necessary requirement. The real technological part of hand tracking is the software behind it.

Picture 2. Oculus Quest hand tracking cameras. [15]

There are different ways of handling the tracking process and companies are guarding their technologies very closely, so detailed information about hand tracking algorithms is not readily available, but the one thing in common with all hand tracking technologies is that it is AI based, usually some form of machine learning. The cameras feed the tracking algorithms with visual data of hand position and orientation, dozens or even hundreds of pictures per second. Then from this data, a skeletal model of the hands and hand position is built on the computer screen (picture 3) [6].

UltraLeap describes their hand tracking process as follows:

”*Our software uses the images (from the cameras) to generate a virtual model of your hand movements. We model not just your palm or fingertips, but the joints and bones inside your hand. This means we can accurately guess the position of a finger or thumb, even if it’s hidden from view.*” [7]



Picture 3. Leap Motion hand tracking [7]

Hand tracking enables the use of hands as an input method on devices that support this technology. Currently there are only a couple of VR headsets with hand tracking support. Therefore, Leap motion, a hand tracking camera developed by Ultra leap, is very popular choice for hand tracking, as it enables almost any device to be controlled with gesture controls.

2.1.1 Leap Motion

The Leap Motion controller by UltaLeap is a small USB peripheral device designed to be placed on a physical desktop or mounted onto a VR headset. Using USB-C connector makes the peripheral compatible with basically every possible device capable of supporting VR. The device has two monochromatic infrared cameras and three infrared LEDs for movement and object detection. Because Leap Motion uses infrared light and cameras for hand tracking, it is not as susceptible to low light environments as a hand tracking using visible light-seeing cameras would be. The tracking area of the device is

very generous and can exceed the visual range of some VR headsets (e.x. Realmax Qian) making the hand tracking feel much more natural, as the device observes a 60cm diameter hemispherical area with 120*150-degree field of view. The cameras generate almost 200 fps of reflected data making the tracking feel smooth and latency free.

The data is analyzed by the Leap Motion software using “complex maths” in a way that has not been disclosed by UltraLeap. The device synthesizes 3D hand position by comparing 2D frames generated by the cameras. The overall average accuracy of the controller is 0.7 millimeters allowing for very precise hand gestures and operations. [8]

2.1.2 Occulus Quest

Oculus quest is a fully standalone VR headset created by Oculus VR, which is a division of Facebook Inc. The Quest is a completely wireless VR headset, and it is one of the only VR headsets, that comes with a built-in hand tracking. This combined with easy-to-use systems for both, end users and developers make the Quest easily the best mobile VR platform available at the time of writing this thesis.

For hand tracking, Oculus Quest relies on four wide angle monochrome cameras located at each corner of the device. The cameras capture images which are used to build a heatmap of the hands. Using the heatmap an algorithm adds keypoints to fingertips and joints in order to accurately fit a 3D model on top of the hand-skeleton.[6]

Oculus uses deep learning, a form of machine learning, to teach the computer where the finger and joint keypoints are supposed to be. Using examples of different hand gestures, the neural network will learn to predict the hand movements and gestures for the user even if the tracking is momentarily lost or if the hands are partially blocked from the cameras. According to Oculus, the keypoint estimation and predictions are a common problem with hand tracking that they solve using AI learning.[9]

2.2 Haptics

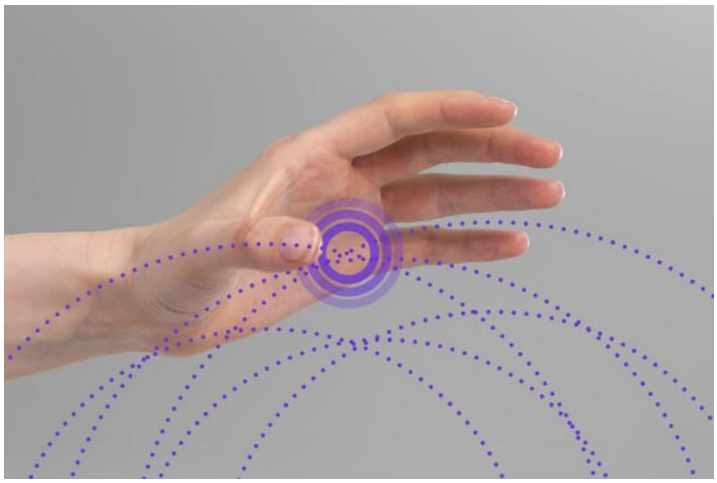
Haptic communication is a form of non-verbal communication that refers to the way people communicate with the sense of touch. Haptics offers an extra dimension in a VR environment which is essential for the feeling of true immersion. This is most found in uses of vibration or electrostatic shock in contact with skin [4].

Most common uses for haptics in VR are controllers giving vibration feedback to the user. This is the first way haptics were used in gaming and the first uses of this technology precedes its use in VR by almost two decades. Expanding on this same principle were the body suits used in VR. The difference to vibrating controllers is that the body suit would use small electric shock to communicate, for example, a hit on the body in a first-person shooter game (picture 4). The focus of this section, however, is the haptic feedback on hand tracking. The problem of course is that vibrations via a controller is impossible with hand tracking. This leaves two options for haptics: Gloves or ultrasound.

Picture 4. Haptic body suit [16]

The use of ultrasound in hand tracking is a still developing technology for haptics which shows a lot of potential. It works by having an array of small ultrasound emitting speakers in front or below of the user's hands. The soundwaves are too high frequency for humans to hear, but the waves can be felt as a pressure against the skin. Using complex algorithms different speakers in the array are triggered at a slightly different times, to compensate for the relatively slow speed of sound so that the soundwaves hit the same

spot at the same time, amplifying the feeling of touch and direction (picture 5), giving a stronger feeling of pressure or vibration. The system can accurately track the hands position and pinpoint the exact focal point where the sense of touch needs to be applied. Using these signals, it is possible to achieve the same level of haptic feedback as when using controllers or gloves, but it has the added benefit of not having to wear anything that might disrupt the feeling of immersion.



Picture 5. Ultrasound haptics. [17]

Haptics are an essential part of making a more immersive VR experience, but they come with some drawbacks. One of the biggest advantages of using mobile VR platforms with hand tracking is the lack of need for extra cables or controllers. Using any form of haptics require the user to pack more gear and to set up the haptics systems defeating the purpose of mobility and ease of use that comes from only having a headset with you. However, if the goal is highest possible level of immersion, which should be the case for most training or simulation software, then mobility surely can be sacrificed in favor of using haptics for better immersion.

3 USER EXPERIENCE IN VR

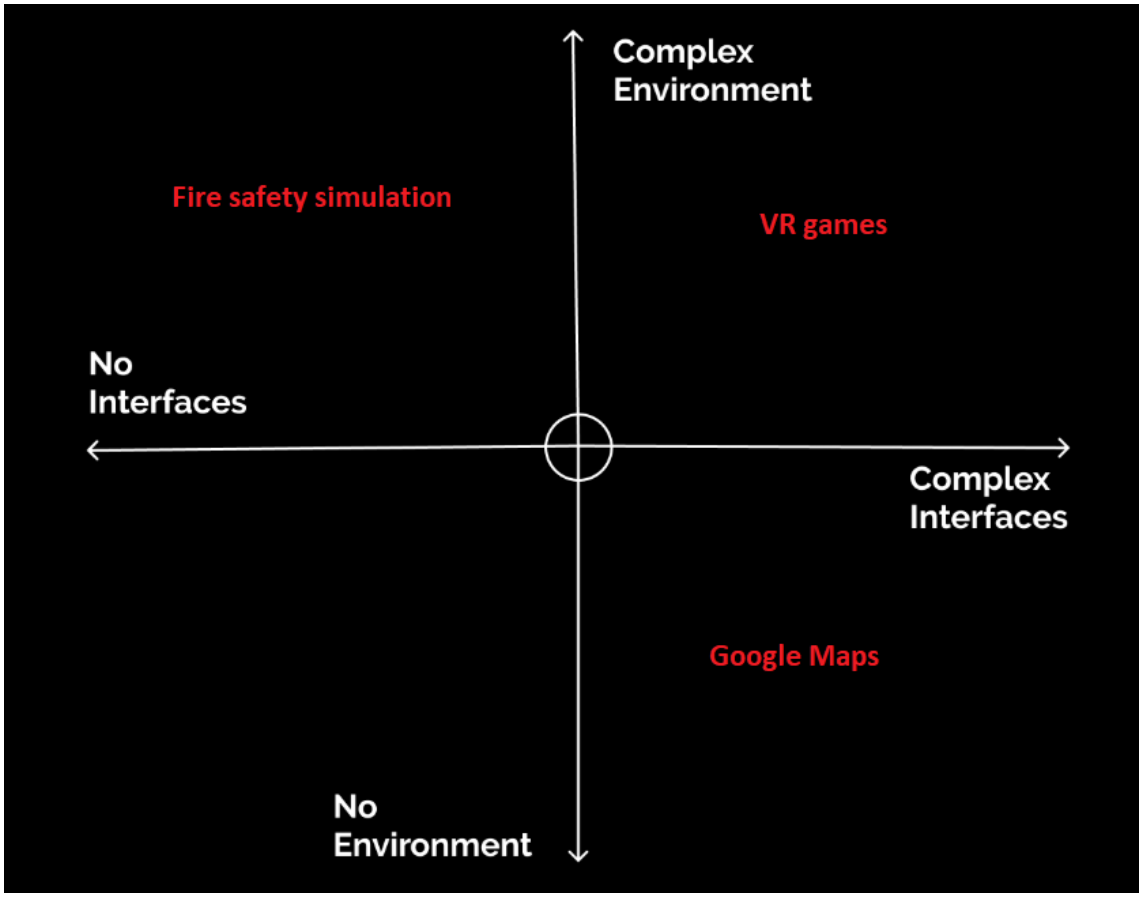
3.1 User experience in virtual reality applications

User experience, or UX for short, is the experience a person has using a product or service. UX is a combination of different disciplines, such as interaction design, visual design, usability, content strategy, user interface design, information architecture, user research and so on. In short, user experience is the combination of look, feel and usability. This chapter focuses on user experience in VR in the case of hand tracking as well as some generic UX in VR [10].

When designing VR applications, the apps are made up of two types of components: environments and interfaces. Environment can be thought to be the world where the user enters when they put on a VR headset, a virtual planet that can be explored or the hallway where the player walks during the fire safety simulation.

An interface is the set of elements that users interact with to navigate an environment and control their experience. All VR (and XR) applications can be positioned along two axes according to the complexity of these two components.[11]

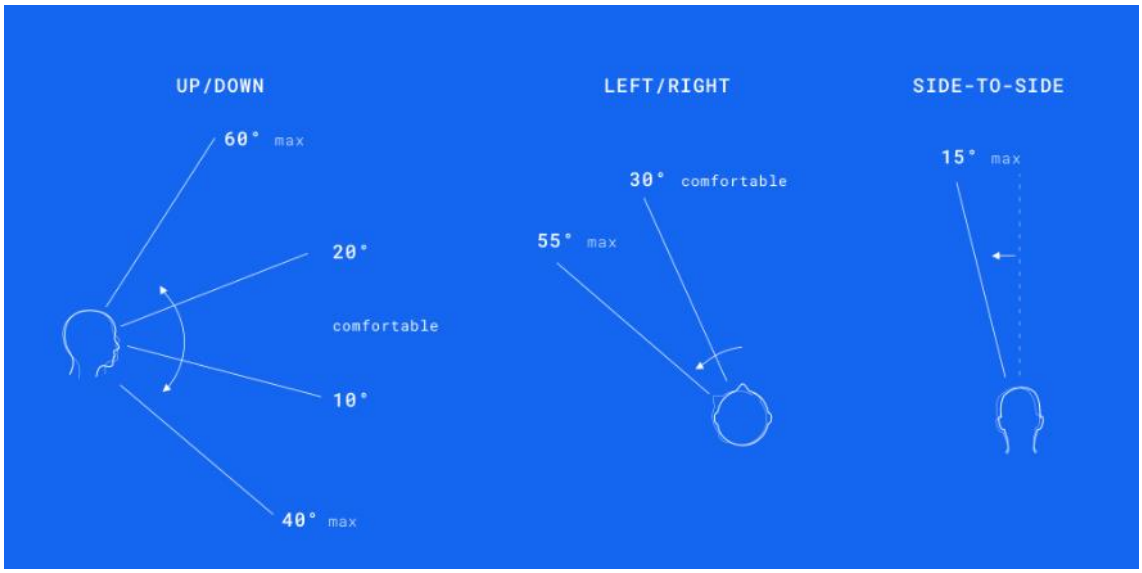
In the top left quadrant of picture 6 are XR applications, such as the fire safety simulation mentioned above. These apps have a fully designed environment but no visible interface at all. In this quadrant are also apps like theme park rides where the user is only looking around but otherwise locked in for the ride. In the opposite quadrant are apps that have a developed interface but basically very little or no environment at all. Here are apps such as Realmax Qian or Oculus Quest home screens, or informative AR apps such as Google maps for example. Complex VR games which have fully developed rich environments and complex controls with many layers of UI, Elder scrolls V: Skyrim (VR) for example, would be part of the top right quadrant.



Picture 6. Mixed reality UX design axes.

In addition to normal UX design, such as interface or environment, VR applications come with one other important design category when thinking about user experience: Ergonomics.

With head mounted AR and VR applications it is important to consider how much the users are required to move their heads for any experience longer than a few minutes. Google VR designer Mike Alger and Alex Chu of Samsung research claim that users' comfort level when rotating their heads horizontally is 30 degrees to each side with maximum rotation of 55 degrees. For vertical motion, rotation of 20 degrees upward is comfortable, with maximum rotation upward of 60 degrees. Rotation downward is around 12 degrees comfortably with maximum of 40 degrees as shown in picture 7.



Picture 7. Comfortable head rotations in VR. [18]

When defining comfortable head rotation zones for head mounted XR applications, it is also important to consider how the app will be used. Does the app require users' direct interaction, such as with hand tracking and gesture controller apps, or just point and click via touchpad or controllers? With direct interaction, it must be considered how that can be achieved comfortably, especially if the app is intended for extended use. As more and more XR applications are utility based, such as training simulations, this consideration will become more important [12].

3.2 User experience in hand tracking

When considering UX in a VR application user inputs, and how those input controls are taught to the user are an important part of the UX. The user must be made aware how to navigate the VR environment and how to interact with it. This is usually done by presenting the user an image of the controller configuration or the user is familiarized with the controls one step at a time.

This process becomes much simpler with hand tracking. Gesture controls must be taught to the user similarly to how they would be taught if using controllers, but basic functionality and simple interactions, like grabbing, pushing or throwing do not need any special tutorials and designers can assume that most people are familiar with the

functionality of their own hands. For example, the simple task of pushing a button in VR, when using a controller, requires the user to know that they need to move the controller on top of the button in the virtual world and then press some button in the controller to activate it. With hand tracking such tutorial is not needed as the user would push the button just as they would in the real world.

Picture 8. Pushing a button with hand tracking.

Another considerable UX advantage when using hand tracking over controllers is that the user is aware, and can see, their own hands. This helps with multitude of UX problems compared to using controllers. The simple first UX improvement is that the user will always have their hands with them. This is an especially beneficial with mobile XR headsets as no additional peripherals are needed when hand tracking is used. Additionally, body language is important and can be communicated via hands much more effectively than with controllers, where hand gestures are limited to pre-programmed gestures that the user executes by pressing a button in the controller.

Interacting with hands and fingers feels good and intuitive, thus improving the user experience. Users seeing their hands reinforces scale, usability and physical presence therefore interacting with virtual objects feels natural. Hand position can always be sensed perceptually, seeing your virtual hand mimic the motions of your real hand feels real and reinforces the sense of immersion [6].

The problems with hand tracking from the perspective of UX become apparent when the application requires numerous or complex inputs in order to perform a function. In order to perform a specific function outside of simple physical interaction, there are two ways of executing them with hand tracking: either simply pushing a button in some on-screen UI, which would destroy immersion, or by doing some hand gesture. These kinds of inputs are much simpler to perform with a controller as the user would only be required to press a button in the controller.

Player movement is also a huge UX problem with hand tracking. The most common way of movement in VR is teleporting from one point to the other. This method of movement is preferred because artificial linear movement in VR is the most common way to get motion sickness with VR headsets. However simply teleporting from point A to point B does not induce motion sickness nearly as often. Teleporting is normally done with controller by pushing one button in order to draw the point where to teleport and then pushing another button to execute the jump. This would obviously be impossible to do with hand tracking, so some other movement methods must be implemented.

When developing an application for VR it is often better to decide if player interaction and inputs are done via controllers, hand tracking or some other method of control. Most VR applications are either developed for hand tracking or for controller use and can rarely be switched from one to the other by the user preference. The control systems are so different from each other that they present such huge UX problems that the applications would in essence have to be rebuilt to use hand tracking over controllers, or vice versa.

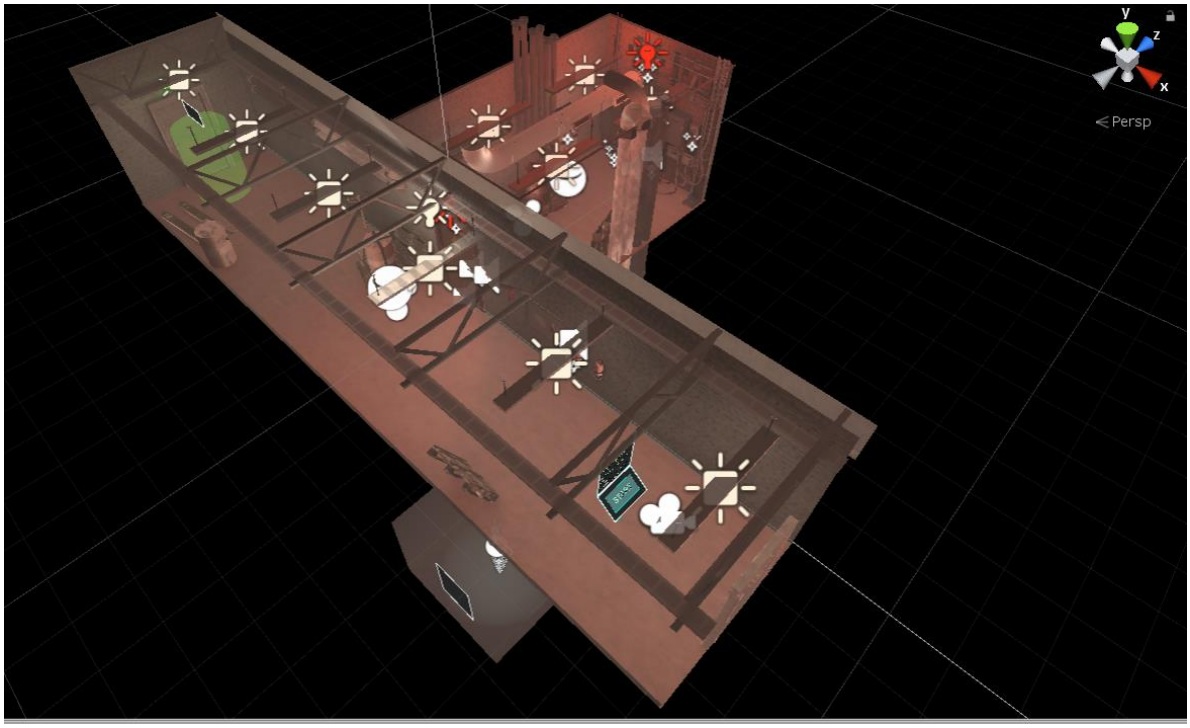
4 REQUIREMENTS ANALYSIS

This chapter will consist of a general requirement analysis for modifying the Fire safety simulation application to be playable with hand tracking. The app has been developed to be used with dedicated VR platform and a PC. The application will be modified to be playable using mobile VR/XR platforms with player controls being done with hand tracking. While performance and graphical adjustments will be part of the requirement analysis, the focus of this chapter will be on adding hand tracking support to the application.

4.1 Overview – Fire safety simulation

The Fire safety simulation is a VR application developed by Turku Game Lab. The app is meant to be played using HTC Vive VR headset and Vive controllers. The purpose of the practical part of this bachelor's thesis is to remove the dependency of controllers and add hand tracking as the method of control and player input. Additionally, the whole app will be made to work with wireless VR headsets for added immersion and freedom of movement. These changes will be discussed further in sections 4.2 and 4.3 as well as in chapter 5. The following is a rundown of what the player experiences when playing through the fire safety simulation:

When the simulation starts, the player is placed at one end of a hallway greeted by a tutorial UI and a start button. At the far side of the hallway an exit door can be seen and in the middle of the hallway there is a side door. Next to the side door on the wall there is some fire extinguishing gear. (picture 9).



Picture 9. Bird eye view of the firesafety app layout.

The player will see smoke coming out of the side door, from the electrical cabinet. The player then needs to pick up the correct tool to use from a few different extinguishers that are mounted on the wall. Then the player must open the side door and enter the cabinet room, avoiding all the smoke, extinguish the fire from the electrical cabinet and exit the room. After leaving the room and closing the door, the player must hit the fire alarm and then exit the hallway. At the end of the simulation, the player receives a numerical grading according to their performance.

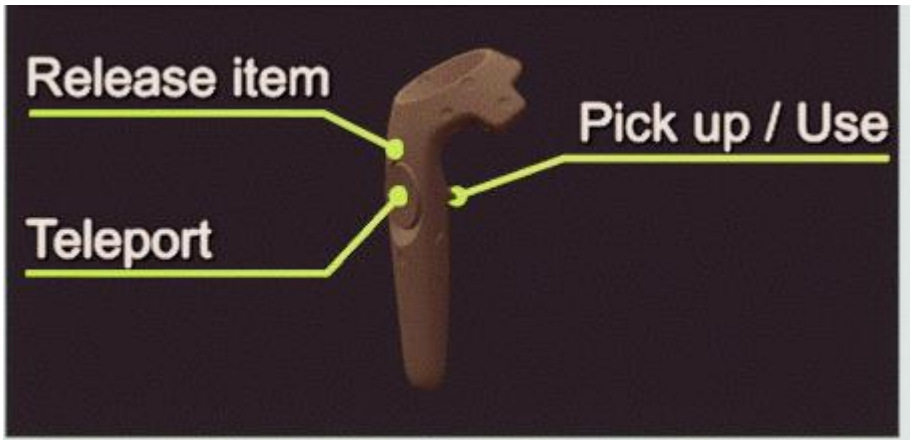
The purpose of the application is to teach the correct way to react to an electrical fire and exit the area safely. The player's performance is measured with several different parameters, including time taken, choosing the correct tool, staying out of the smoke, etc. The simulation feels otherwise realistic, but the use of controllers breaks the immersion and requires training in the use of the controllers. This is the main reason for adding the use of hand tracking to the app.

4.2 Hand tracking

With hand tracking, the main requirements are responsiveness, accuracy and immersive feel. The first two are achieved with the hand tracking hardware and the SDK, while the immersion is achieved by having the environment respond to the player in a similar way that is expected in the real world. Because there is very little that can be done to accuracy or responsiveness with the actual hand tracking technology withing the scope of this bachelor's thesis, the requirement analysis must be focused on immersion and usability of hand tracking when compared to using controllers.

4.2.1 Controls

There are a few different actions the player can do in the simulation, which are: pushing a button, grabbing game objects, opening and closing a door, using the extinguisher. These actions are executed by pressing buttons in the controller as seen in picture 10.



Picture 10. Controller button layout.

With hand tracking, the player must be able to perform these same actions but without any difficult hand gestures and if possible, as naturally as can be done. Pushing a button needs to work like pushing a button in real world. With more complex actions, such as grabbing something, the player needs to execute a gesture. For picking up an object with your hand, the gesture should be similar to the corresponding action in real life, a simple pinch gesture with the index finger and thumb. Unlike with the controller, where the player would pick up an object by pressing the trigger button and then drop it by pressing a thumb button, there should not be different gestures for picking up and dropping objects

then using hand tracking. After picking up an object, it should be held in hand only if the pinch gesture is being sustained. When the player lets go of an object, it should drop down like in the real world.

4.2.2 Player movement

The required player movement actions in fire safety simulation are walking around and crouching. Ideally, player movement should be done by simply walking around. While wearing a mobile VR headset the player is not restricted by wires or a small play area. For maximal immersion, the player should never have to think how they move around in VR. However, this type of movement requires a lot of space and because of that, an alternative way of moving around is needed.

With the controller, player movement is done by pushing a button with your thumb (picture 8) which paints a ray from the controller to the environment. When the player lets go of the button, the player is moved to the location where the ray was pointing. Similar way of movement should be an efficient way to move even while using hand tracking, but now this teleportation action needs to be executed with a hand gesture.

The gesture needs to be simple and easy to remember and there should not be any way for misfires, as that would make hand tracking confusing and it would break immersion. At the time of writing this bachelor's thesis there is no standardised movement system that utilizes hand tracking or gesture controls nor are there any guidelines for building such a system. Also, the lack of literature, conference papers or research articles on this topic makes it clear that we are still not close to standardising movement with hand tracking.

After a lot of prototype designing, planning and some early testing with hand tracking, an elegant and somewhat humorous way to execute the movement action was found: The method of movement would be to use a ”finger pistol” gesture. This is an easy to do gesture that should never be done by accident, which makes it a good gesture for the movement action. The ray can be painted from the tip of the index finger when the “finger pistol” – gesture is initiated and the teleport action would take place after “firing” the finger pistol, or in other words, moving the thumb from upward position to down.

4.3 Performance

The original fire safety simulation was developed to be played with a pc that is capable of supporting VR. This means that the pc where the simulation is supposed to be run, is powerful enough to support VR, thus powerful enough for the application. Because of this there are no graphics options required. This creates a problem when ported from pc to mobile VR platforms.

The platforms chosen for this project were LeapMotion and Oculus Quest for their mobility and immersion potential. More in-depth overview of these systems can be found in chapter 2.1.1 and 2.1.2 respectively. LeapMotion is a hand tracking peripheral and it needs a VR headset to function properly. In this project, Qian by Realmax was chosen for this purpose. Qian is a very light weight mobile AR/VR headset and while it is comfortable to use, the small size of the device comes with a trade off in performance and graphical processing power.

The fire safety simulation is not very demanding application as far as VR applications go, but it has some lighting and particle effects that may affect the performance of the simulation in the above-mentioned VR systems. Low frame rate or sluggish performance can cause nausea or vertigo in players and should be avoided when possible. Sufficiently smooth performance needs to be considered when developing for less powerful platforms and graphical fidelity must be sacrificed in favour of acceptable frame rates. Good target FPS for VR applications is 90 FPS as mentioned in VR guidelines by Oculus [13].

5 IMPLEMENTATION

Before starting to implement hand tracking to the Firesafety simulation, the whole control system had to be removed. The Firesafety simulation was originally developed to be played with HTC Vive VR system using the Vive controllers. This meant, that the application used Vive SDK and Steam VR as a platform for player movement and interaction. This means that the whole interaction system would need to be rebuild and customized to work with hand tracking instead of controllers.

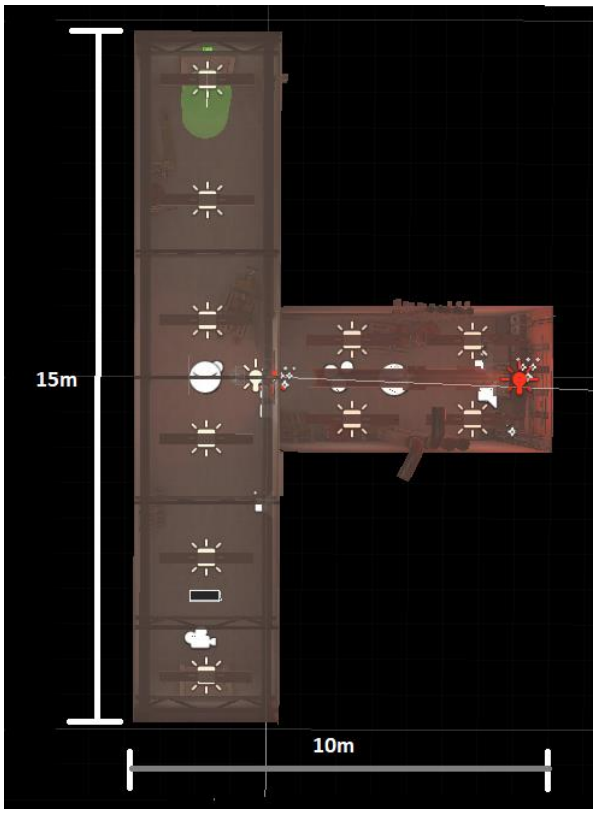
The chosen platforms for hand tracking implementation were Realmax Qian combined with Leap motion and Oculus Quest (as discussed on chapter 4: Requirements analysis). This means that there will be two different versions of the Firesafety application with hand tracking, one for Qian and one for Quest. The plan is to start the development using Qian and LeapMotion and then later port it over to Oculus Quest. The idea is that development should be faster and simpler when using Qian with Realmax's custom SDK for LeapMotion. The SDK is very straight forward with customization and while it doesn't have many features, building custom systems and features on top of the simple SDK framework should let for more streamlined code. Another reason is that when building all interaction and movement systems by oneself it is much easier to add new functionality and modify these systems. When moving the project to Oculus Quest having a working base framework makes it a lot faster to integrate the very robust and feature rich Oculus SDK with the existing Firesafety hand tracking application.

When implementing hand tracking to the Firesafety simulation there are three major system to design: Movement system, UI and button interactions, and object grabbing and manipulation. The SDK already has a basic button interaction system built in, so customizing that to work on this particular app is a simple process. However, the movement system and object interaction systems must be completely built from nothing.

5.1 Movement system

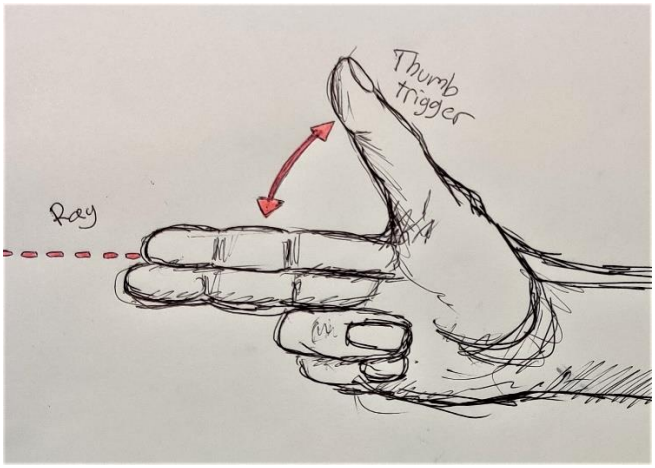
As specified in the requirement analysis, ideally, the biggest benefit with mobile, wireless VR systems, like the Realmax Qian, is that player movement can be implemented to work by simply walking around. After some initial tests with the Qian SDK, it was apparent that the SDK realistically tracked movement in the VR world.

To make movement speed and walking around in VR feel realistic, it is important that the VR world scale is correct. The fire safety simulation was already made in 1:1 scale (one unity unit corresponds to one meter) so additional scaling was not needed. In ideal situation this is all the movement system required in this application, granted that there is enough empty space to walk around. The fire safety simulation requires about 15m by 10m play area (picture 11) if movement is done with walking around in the real world. Because most of the time VR players are going to be stationary while playing, it is crucial to have an additional movement system in place. As specified in the requirement analysis a teleportation system for movement is needed.



Picture 11. Firesafety simulation VR world size.

The way the teleportation system was supposed to work is as follows: When doing a ”finger pistol” gesture, a ray is painted from the tip of the index finger. When ready to teleport, use the thumb to initiate the teleport as if firing the finger pistol (picture 12). Then the player’s position would be moved to where the ray was hitting when fired.



Picture 12. Teleportation gesture concept.

Ideally a gesture system that recognizes different gestures would be preferred, but since the SDK didn’t include any framework for a ready system and because the number of needed gestures was low, it was better to use ”triggers” to recoginze different hand positions and gestures instead. The ”triggers” are invisible game objects which fire off an event or an algorithm when they collide, for example having a trigger in the tip of your thumb and in the tip of the metacarpal bone of the index finger would fire off the teleportation event when they collide.

Implementing this system proved to be a bit of a challenge. The hands done in VR by LeapMotion were created in runtime, so it was not possible to modify the hands directly from Unity editor. Instead, the triggers and the ray had to be implemented also in runtime, after the SDK created the hands. The LeapMotion hand skeleton has predefined bone positions so inserting the triggers directly to the bones was possible as done in this code snippet (picture 13). The trigger position is updated each frame after the hand position is updated, this ensures that the triggers are always found on the correct positions.

```
void TeleportInit()
{
    // Initialize teleport ray
    GameObject go = new GameObject();
    go.name = "TeleportRayHolder";
    go.tag = "QianTeleport";
    teleportRayPoint = go.transform;
    GameObject tp = Instantiate(qianTeleportController, teleportRayPoint);
    qianTeleportController.transform.localRotation = teleportRayPoint.rotation;

    // Initialize teleport triggers
    GameObject trigger1 = GameObject.CreatePrimitive(PrimitiveType.Sphere);
    trigger1.name = "FingerTrigger";
    GameObject trigger2 = GameObject.CreatePrimitive(PrimitiveType.Sphere);
    trigger2.name = "PalmTrigger";
    trigger1.transform.localScale = new Vector3(fingerColliderSize, fingerColliderSize, fingerColliderSize);
    trigger2.transform.localScale = new Vector3(palmColliderSize, palmColliderSize, palmColliderSize);
    SphereCollider collider1 = trigger1.GetComponent<SphereCollider>();
    SphereCollider collider2 = trigger2.GetComponent<SphereCollider>();
    TeleportTrigger teletrigger = trigger2.AddComponent<TeleportTrigger>();

    collider1.isTrigger = true;
    collider1.radius = 0.5f;
    collider2.isTrigger = true;
    collider2.radius = 0.5f;
    Rigidbody rigidbody = trigger2.AddComponent<Rigidbody>();
    rigidbody.isKinematic = true;
    fingerTrigger = trigger1.transform;
    palmTrigger = trigger2.transform;

    tp.GetComponent<QianTeleport>().trigger1 = trigger1;
    tp.GetComponent<QianTeleport>().trigger2 = trigger2;
    tp.GetComponent<QianTeleport>().teleportTrigger = trigger2.GetComponent<TeleportTrigger>();
```

Picture 13. Teleport triggers initialization function.

The teleportation system worked nicely and moving around felt as natural as can be expected. The trigger system was working perfectly and moving around felt at least somewhat immersive. However, the accuracy and fidelity of the teleportation mechanism left a lot to be desired. After several rounds of testing there were some problems found that were not caused by bugs in the code. The greatest issue was that while the hand tracking is very accurate in ideal conditions, sometimes when the cameras do not see the actual position of the fingers, the predictive AI gets confused and the fingers start convulsing. This makes accurate teleporting completely impossible. This also explains why no hand tracking SDK comes with a standardized movement controls. The hand tracking cameras simply do not track thumb movements accurately enough to utilize thumb as a trigger gesture.

The iterative process for the teleportation system was long and had many iterations. The most promising systems were the original finger pistol system and a system where the ray would be fired off from the player's head and the teleportation was triggered by making a fist gesture.

The problem with the head mounted ray was that the ray was always in the field of view of the player and the whole system felt very un-immersive. On the other hand, the functionality of this system was perfect. The ray was very accurate, and the trigger mechanism worked without fail. The compromise was to combine the immersiveness

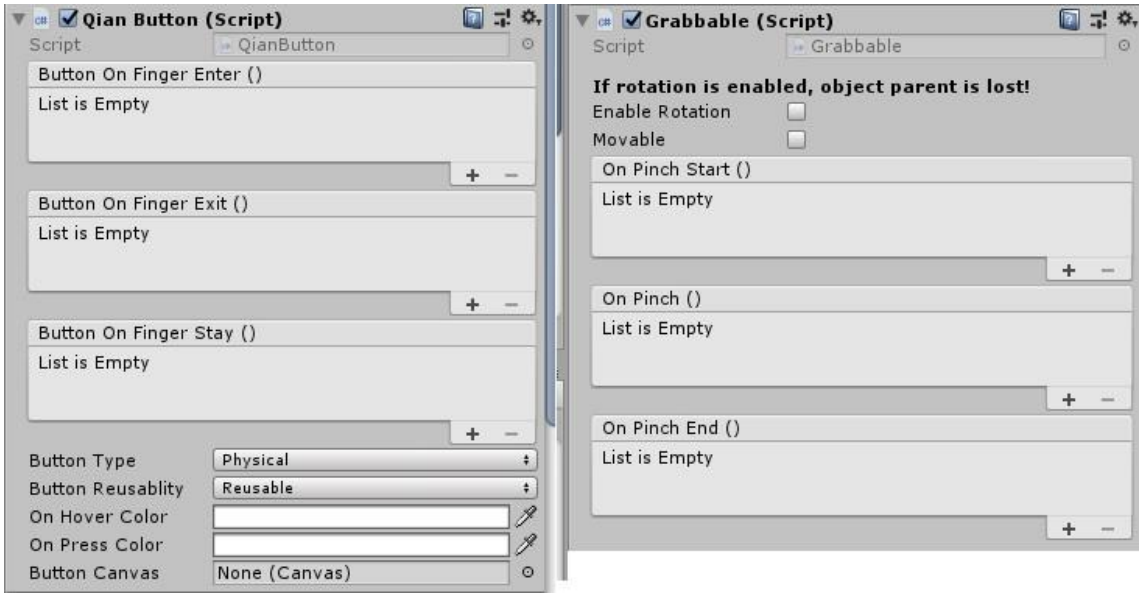
and ease of use from the finger pistol system to the fidelity of the head mounted ray system.

The final iteration was successful. The ray would be fired from the wrist of the player, which was always tracked accurately despite of the hand position. The teleport action was triggered by making a fist gesture which also was accurately tracked (most of the time) even if the hand position was such that the hand tracking cameras didn't see all fingers simultaneously. This iteration of the teleportation system was the one that combined all the best attributes of previous iterations and is accurate and reliable while not sacrificing usability.

5.2 Object manipulation system

The object handling system consists of two main parts: the more intricate grabbing system and simpler button-pressing system. LeapMotion SDK came with basic pinch gesture handling system which was simple to modify to fit the specification and requirements of the fire safety simulation app. After the hand gesture event system was working, scripts to handle object grabbing and manipulation and button presses were needed. These scripts and systems had to be completely built from nothing which was a long task. The grabbing system went through many iterations before working adequately, however, the button script was a simpler process.

Handling button presses was a straightforward task. Whenever players fingers would enter the button's trigger area, the button is pressed. Implementing this functionality was also straight forward and the basic functionality worked as intended. There are two types of buttons to press in the app: UI buttons and physical buttons. The button script also needs to distinguish whether the button is reusable, or one time use only. Because the button script was duplicated to many objects, it was better to make the script easily customizable without changing the code every time. As seen in picture 14, all the button parameters can be changed from the script inside Unity editor without the need to even see the code anymore.



Picture 14. Button and Grabbable scripts in Unity editor.

Additional to the two button types, there is a third unique case as well: The door handle. It is treated as a physical button, but it is not triggered with a button press gesture, instead the door works like it does in real life, by grabbing the handle. This required an extra script to tie together the object grabbing system and button system, but from the code's perspective, the door handle is essentially a button.

The real challenge was to develop a system for grabbing and moving game objects. The SDK came with a simple script to handle picking up objects. It works by changing the position of the picked-up object to the position of the hands and updating the position each frame. The way it works in-game is whenever the player does the grabbing or pinching gesture, the object would appear to be held in hand and when the pinch is released the object's position is no longer being updated. This works in principle but has a few obvious problems. First, the object would only be held in hand if the pinch gesture is being executed. This means that if there were any problems with tracking or the hand position was lost for a fraction of a second, the pinch would break, and the object would drop. The second problem was that the system did not distinguish between multiple objects. If the player tries to pick up one object from a pile of objects, the one being picked seems somewhat random. This was because the script would choose an object to be picked up based on the hand colliding with an object. However, if the hand collides with multiple objects at the same time, or some object colliders are within each other, the selection is random.

These problems were corrected after a long iteration process, but the final version of the grabbing script works very well. The problem with dropping objects because momentarily losing tracking is fixed by making the picked-up object to be a child object of the virtual hands for the duration of the grab. A child object is a game object that inherits its scale, rotation and position parameters from its parent object. In other words, if you move a parent object, the child object moves with it. Now the picked-up object's original parent is stored in a variable, then the object is made as a child of the VR hands. When the player is ready to drop the object and lets go of it, a ”grab end” event happens and the object is dropped and its original parent is restored from the variable.

Now it doesn't matter if tracking of the grabbing gesture is lost momentarily, which happens often when the player teleports or turns around, because the object is still being held on the players hand until the player is ready to let go of it. Additional upside is that the grabbable script and the button script looks very similar as they both use an event-based system to handle their functions (picture 14). Now it is easier to customize the app or continue its development for someone else than the original author.

The second problem of distinguishing between two close by game objects when grabbing was rectified by adding a small triggers on the tip of the VR fingers to help with precision (similarly to how the teleport triggers were done in picture 13). Now the collision is not being detected for the whole hand but instead only for the fingertips. Also, a highlight was added to the object that is going to be picked up with the pinch gesture. Now the player can know which object is being targeted for the pickup before the action (picture 15).

The movement system and the object pickup system worked beautifully in a vacuum, but there was a problem when they were put together. Sometimes when holding an object in hand, the movement system would think that a teleport gesture was being executed. This resulted in some confusing and seemingly random teleports in the game. The fix for this unwanted feature was to disable the ability to teleport while holding an object. Additionally, the teleport could only be initiated when targeting the ground. This is shown to the player by changing the teleport indicator from blue to black if the player was targeting an unallowed location for the teleport, as seen in picture 15.

Picture 15. Removing the pin from the extinguisher.

5.3 Porting the application from Realmax Qian to Oculus Quest

The app was originally developed for pc platform using HTC vive for the VR. When the hand tracking was added the project was also modified to be playable in mobile platforms. The workflow was to first make the app playable with Realmax Qian with Leapmotion hand tracking and then make a version with Oculus Quest. The idea being that if the app is playable with a mobile platform, it would also be playable with a more powerful mobile gaming VR headset. Indeed, the only changes needed to implement in order to port the app to Quest was to remove any Qian and Leapmotion dependencies from the app and change them to work with Oculus Quest SDK.

Because most of the functionality was done with this change in mind the logic for movement and hand gestures was done with custom scripts and the SDK would only be responsible for the actual hand tracking. This means that while the hand tracking for the app took many weeks to implement, the changes required to make it work on Quest only took a fraction of that time.

The only real problem with Quest was that the system used a different approach for object detection when choosing an object for the pinch gesture. The leapmotion version

had a system where the object would be identified by colliders. With Quest, the system chooses which object to grab by calculating the distance from the colliders of nearby objects to the VR hands. For example, if there were two objects within grabbing distance when the grabbing gesture is executed, say, one object being 11mm from the hand and the other being in 26mm distance, the first object would be chosen for the grab. This system works flawlessly as long as the objects are not overlapping. However, this becomes a problem when trying to remove the pin from the extinguisher (picture 13). The collider of the extinguisher surrounds the collider of the pin and because of that, the distance to the pin is always greater than the distance to the pin.

The solution was to simply use the exact system that is in use for the Qian/Leapmotion version. The only downside being that the Quest hand tracking SDK already has a robust gesture recognition framework which could not be used with this app. This was more of a downside from the developer point of view instead of an actual functionality or usability problem.

In the end the switch from one platform to the other was a smooth process and from the user’s point of view there is little to no distinction between the Qian or Quest versions.

6 TESTING AND FINDINGS

The development process for the hand tracking implementation was iterative and the app went through testing on a daily basis. Most of that testing was finding bugs and fixing broken systems as well as simply experimenting on different ideas. The purpose of the testing was mostly to find the best working solution for hand tracking. Throughout the development process this was the only type of testing done and user tests were only done after the project was finished. This worked as a self-evaluation tool on how well the hand tracking implementation succeeded and as a metric for future hand tracking development. However, because the testing was done after completing the project, it did not affect the development in any way and the user testing could not be taken as feedback to improve upon.

While doing this bachelor's thesis the world was going through a global pandemic which affected not only the thesis work, but it made testing a great deal more difficult. For this reason, the user testing was conducted in about 2-month period with only 14 participants. Luckily, there was enough of test data to see an obvious trend with the testers.

The testing was carried out with each tester playing through the application two times, once where they familiarize themselves with VR and hand tracking as well as the goals of the app and a second time where they would run through it completely by themselves without any hand holding or comments.

The test group consisted of only one tester who had tried hand tracking before, three testers who were familiar with virtual reality and had VR gear of their own, three other testers had tried VR at least once before and the rest of the test group had no experience with VR or hand tracking at all.

The testing was conducted using Realmax Qian with UltraLeap Leapmotion for hand tracking. Even though there were two versions of the app, one with oculus hand tracking and one with leapmotion, the purpose of these tests was not to compare the hand tracking hardware, but the feasibility and usability of hand tracking as an alternative to a controller based version. From the testers point of view the different hand tracking versions behave similarly and thus have no effect to the test results. Additionally, the user feedback was to be focused on hand tracking UX, not the possible hardware limitations.

The test group played through the application using only hand tracking. For a better comparison it would have been better to have a second test group for testing the controller version of the firesafery simulation, or have two separate tests with the same group, one with controllers and one with the hand tracking version and then compare the results. This, however, could potentially cause bias towards hand tracking for testers who are not familiar with using controllers and vice versa for testers who have experience using VR with controllers. Additionally, having two test groups was not possible due to the pandemic restrictions that were in place at the time of the testing.

After playing the application the testers filled a questionnaire where they would give an answer between 0 and 10, where 0 was the most disagreeing or a negative experience and 10 the most positive. The questionnaire was loosely based on the User experience Questionnaire (UEQ). The UEQ is part of standardized methods used to evaluate user experience. The user answers the questionnaire giving a numerical value for each question, these values are then collected and presented in a graph for easier and more reader friendly format for analysis [14].

Graph 1. User testing results.

From the testing data (graph 1) can be seen that the lowest user satisfaction was with the usability of hand tracking, only 6,2 out of 10 average. This was mostly due of having

only one question on the usability of the hand tracking. Instead, the usability should have been divided in to two questions, where player movement would have been excluded from the rest of hand tracking usability. Moving around turned out to be the most frustrating part of hand tracking and the most common complaint with testers.

Interestingly people who were familiar with VR gave much lower score on usability and usefulness than people who were using VR the first time. The most logical explanation for this phenomenon is that while hand tracking is good, is not extremely accurate, especially when compared to using controllers. People with VR experience have most likely been using controllers for a long time which would make them biased against hand tracking. Such bias did not exist on testers who had never used VR before, so they were much more willing to give a higher score. On overall satisfaction there were next to no variance between different testers. However, there were a few who had an obvious distaste for any gaming related technology, and this could explain the overall low scoring in every question for these testers.

7 DISCUSSION

From the test data and tester interviews it can be concluded that testers were mostly positive about hand tracking even though some people had doubts about the technology not being quite ready to replace controllers in VR. Of course, hand tracking should not replace controllers in the first place, but to be a more immersive experience when the gameplay includes touch-based inputs.

From the experience of working on this project as well as doing research about hand tracking, the author agrees with the consensus of the test group. The greatest caveat being that apps which utilize hand tracking should be developed with hand tracking in mind right from the start. Discussions with the test group suggest that the most immense challenge with hand tracking is always going to be player movement and with apps where the movement is carried out with controllers, it is very challenging to replace that movement system while retaining the immersion from hand tracking.

When considering input systems for applications there will always be some inputs that are better to do with controllers than with hand tracking, mainly the player movement system. Before the industry comes up with some standardized movement system for hand tracking applications, it is easy to make the statement that controllers will simply be better for movement compared to hand tracking. There will be situations when controllers are superior choice for an app and situations where hand tracking will be supreme. These are choices that must be considered case by case when developing VR applications. Undoubtedly when developing a new application there could be workarounds for the limited player movement of hand tracking. For example, simply remove the reason to move inside the VR environment, instead after performing a task at one location, teleport the player to the location of the next task and so forth. This would also allow the player to remain sitting while playing, removing any negative feelings some players experience in VR, such as vertigo. This type of movement was out of the question for the Fire safety simulation because the goal of the project was to convert the app to be played with hand tracking and compare the feasibility of such conversion. The above-mentioned movement system would have drastically changed the gameplay of the application making any such comparison impossible.

From all the hand tracking hardware tested, the Oculus Quest gave overall the most satisfying experience. The reason was in large part the smoothness and accuracy of the

hand tracking and how simple it was to use. From Oculus' point of view, the most notable downside with this project was that because of compatibility issues it was not feasible to use all the features in the Oculus hand tracking SDK, mainly the gesture system. On the other hand, in the fire safety simulation there were only really two gestures needed, one for picking up game objects and one for teleportation, and because in the perfect situation movement is done with actual walking instead of teleporting it can be said that pickup/grabbing gesture is the only one needed. This means that using the robust gesture system and gesture libraries would have been redundant anyway as setting up the system might be even more work than just simply coding in one gesture, especially because this was already done with the Qian version.

From a player's perspective, the author finds very little difference between Qian and Quest when it comes to hand tracking. Qian hand tracking however did not seem quite as smooth or accurate as the Quest. It is possible to speculate whether this is because of what LeapMotion can do, the possible limitations of Qian's LeapMotion SDK or that the version of LeapMotion in the SDK was slightly old in software development terms. Most frequently this happened as a total momentary loss of tracking if the hand was in a "weird" orientation. One possible reason for this could be the number of tracking cameras used in LeapMotion, which is lower than the number of cameras Oculus Quest utilizes but this is complete speculation by the author.

From a developer's viewpoint credit should be given to Qian because the SDK and hand tracking systems were easy to modify to fit the needs of the project. Also, the method how Qian approaches pinch gesture recognition and handles grabbing objects that are close to each other was simply very good. It gave much more consistent results when grabbing objects compared to the method Oculus Quest is using and gave overall more consistent user experience.

8 CONCLUSION

This chapter expresses the opinions of the author for this Bachelor's thesis and working on the project as well as presents the overall conclusion from the author's point of view.

The author finds the general the experience of working in this this project as positive. There is still some room for improvement, like smoothing out the movement system, but overall, the author is pleased with the results. When doing research about movement systems in hand tracking VR, it was surprising to find that no standard for it exists, indeed, there were some articles mentioning the lack of such system and how a solution is needed. It is hoped that the testing and iteration conducted in this project could be a part of the work needed to realize such standards.

The fire safety simulation with hand tracking works very well as a proof of concept or a demonstration of what can be achieved with hand tracking and the general capabilities of what these technologies can offer. Additionally, the work carried out in this bachelor's thesis was also used as a base for a conference paper titled *Hand Tracking in Fire Safety - Electric Cabin Fire Simulation* by Mika Luimula, Mazin Al-Adawi and Jere Ranta, presented at the 11th IEEE International Conference on Cognitive Infocommunications, 2020 in Mariehamn, Finland. [19]. The project also offers a ready base of scripts and code as well as already working systems, such as the grabbing system, which can work as a starting point when developing future apps with hand tracking. Of course, it is possible that many of these systems will be included in the hand tracking SDKs in the future when the systems get more standardized.

When reflecting on the work carried in this project or what methods were used, very little could be changed or implemented differently within the project. Of course, some of the methods for both research and development could be refined, but when it comes to the results it is hard to say if there is anything specific that the author feels should have been approached in some other way. This project was set to demonstrate the feasibility of hand tracking in a simulation or training software and that is what this bachelor's thesis does. The application still needs a lot of polish and small tinkering and if development was to be continued, further polishing would be the next step. There are some systems that could be coded slightly more efficiently and the overall performance could be improved. Mainly it would be desirable to increase the FPS to at least 60 instead of the 30 it is now. This is not completely a development problem, but rather a hardware

problem which might be impossible to solve with the current generation hardware. This problem should solve itself as new and more powerful mobile XR headsets are released.

REFERENCES

[1] Microsoft documentation: Mixed Reality, 2019 (https://docs.microsoft.com/en-us/windows/mixed-reality/)

[2] Das, S; Franguiadakis, T; Papka, M; DeFanti, T.A; Sandin, D.J, A genetic programming applications in virtual reality, Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference, 1994, p.1985-1989 vol.3, Orlando, FL, USA.

[3] Milgram, P; Takemura, H, Augmented reality: A class of displays on the reality-virtuality continuum, Proceedings of SPIE - The International Society for Optical Engineering 2351, 1994.

[4] Balladres, A, 2017, Understanding haptics for VR, [Cited April 2020] https://virtualrealitypop.com/understanding-haptics-for-vr-2844ed2a1b2f

[5] Virtual Reality: What is VR hand tracking, 2020, [Cited April 2020] https://linuxhint.com/vr_hand_tracking/

[6] Oculus hand tracking deepdive 2019, [Cited April 2020] https://www.youtube.com/watch?v=gpQePH-Ffbw

[7] UltraLeap tracking, 2020, [Cited March 2020] https://www.ultraleap.com/tracking/

[8] LeapMotion Datasheet, 2019, [Cited March 2020] https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf

[9] Han, S, 2019, Using deep neural networks for accurate hand tracking on oculus quest, [Cited May 2020] https://ai.facebook.com/blog/hand-tracking-deep-neural-networks

[10] User Experience design, Interaction Design Foundation, [Cited October 2020] https://www.interaction-design.org/literature/topics/ux-design

[11] Purwar,S, 2019, Designing user experience for virtual reality applications, [Cited October 2020] https://uxplanet.org/designing-user-experience-for-virtual-reality-vr-applications-fc8e4faadd96

[12] Mealy, P, Dummies: Designing augmented reality apps, [Cited April 2020] https://www.dummies.com/software/designing-augmented-reality-apps-comfort-zones-interfaces-and-text/

[13] Oculus, Guidelines for VR performance optimization, [Cited March 2020] https://developer.oculus.com/documentation/native/pc/dg-performance-guidelines/

[14] Díaz-Oreiro, I; López, G; Quesada, L; Guerrero, L, Standardized Questionnaires for User Experience Evaluation: A Systematic Literature Review, Presented at the 13th International Conference on Ubiquitous Computing and Ambient Intelligence UCAmI 2019, Toledo, Spain, 2–5 December 2019.

[15] Sam U, 2020, What is VR hand tracking?, [Cited April 2020] https://linuxhint.com/vr_hand_tracking/

[16] Teslasuit, 2019, [Cited October 2020] https://teslasuit.io/

[17] Stratos Explore, UltraHaptics, 2020, [Cited October 2020] https://www.ultraleap.com/product/stratos-explore/

[18] Alger, M, (September 2015), Visual Design Methods for Virtual Reality, [Cited October 2020] http://aperturesciencellc.com/vr/VisualDesignMethodsforVR_MikeAlger.pdf

[19] Luimula, M; Ranta, J; Al-Adawi, M, Hand Tracking in Fire Safety - Electric Cabin Fire Simulation, Presented at: 2020 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Mariehamn, Finland, 23-25 September 2020.

References for abbreviations

[A1] Artificial Intelligence, 2010, TechTerms, [Cited November 2020] https://techterms.com/definition/artificial_intelligence

[A2] Augmented Reality, 2016, TechTerms, [Cited November 2020] https://techterms.com/definition/augmented_reality

[A3] FPS, 2015, TechTerms, [Cited November 2020] https://techterms.com/definition/fps

[A4] Mixed Reality, 2019, [Cited November 2020] TechTerms, https://techterms.com/definition/mixed_reality

[A5] SDK, 2010, TechTerms, [Cited November 2020] https://techterms.com/definition/sdk

[A6] User Interface, 2009, TechTerms, [Cited November 2020] https://techterms.com/definition/user_interface

[A7] User Experience, 2019, TechTerms, [Cited November 2020] https://techterms.com/definition/user_experience

[A8] Virtual Reality: Definition and Requirements, NASA, [Cited November 2020] https://www.nas.nasa.gov/Software/VWT/vr.html