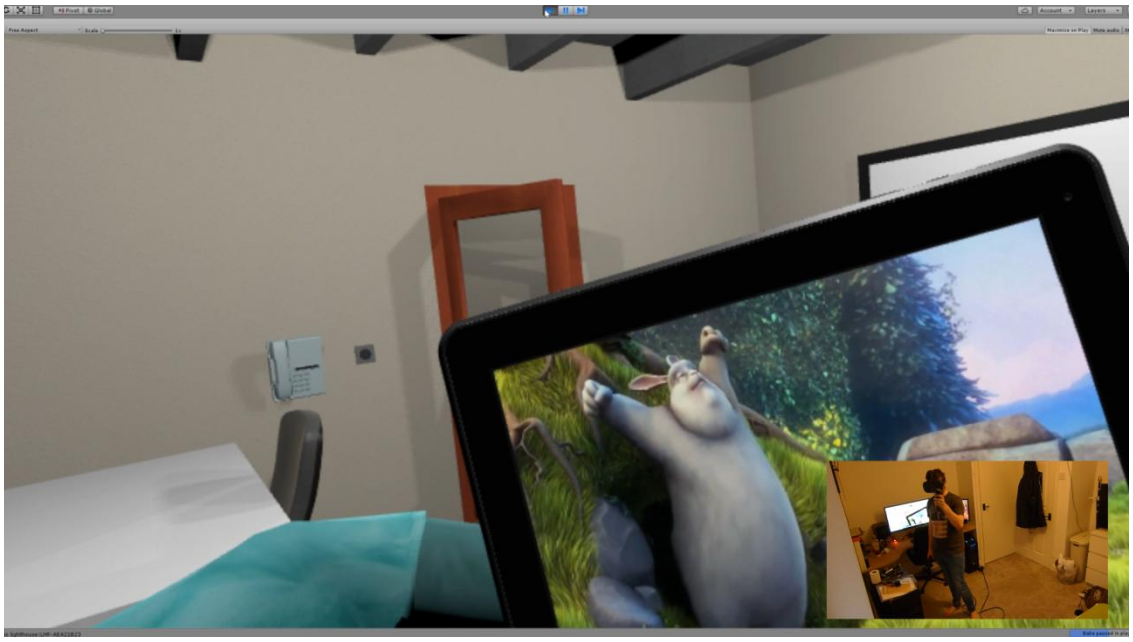
USABILITY AND IMMERSIOIN:

A VIRTUAL REALITY STUDY



Author: [redacted]

Degree Title: Bsc (Hons) Computer Games Technology

Supervisor: Huw Lloyd

DECLARATION

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Signed: [redacted]

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Huw Lloyd, as well as Leah Greene and Phillip Chandler from the department of nursing for all of their support and guidance throughout this project.

ABSTRACT

Virtual Reality is a relatively new technology and as such it is important for VR software to be accessible and easy to use. VR also allows for an amount of immersion and presence in virtual worlds that hasn't been possible with traditional hardware. This project aims to examine the relationship between usability and immersion in VR as well as find out if they are competing concepts.

CONTENTS

DECLARATION i

ACKNOWLEDGEMENTS i

ABSTRACT i

CONTENTS ii

TABLE OF FIGURES iv

INTRODUCTION 1

Project Background 1

Aim 2

Objectives 2

LITERATURE REVIEW 3

Background 3

What is VR? 3

History of VR 4

VR in the modern world 6

What makes an immersive VR application? 7

Autonomy 7

Interaction 8

Presence 8

Assessing presence in real world VR applications 9

DESIGN 10

Required Resources 10

Head Mounted Display 10

Steam VR 11

Game engine 11

Tool Kit 13

Level Design 14

IMPLEMENTATION 17

Setting up the project 17

Setting up VRTK 17

Setting up interactable objects 19

Creating the Simulation Lab room 20
Custom scripts 21
Dimmer switch 21
Replacing the phone after a small amount of time 21
Running water script 23
Displaying the tablet 24
Playing the video 25
Better rotation script 27
TESTING 28
Aim 28
Testing methodology 28
The “Usable” application 28
The “immersive” version 29
Selecting participants 31
Protocol 32
Questionnaire questions 33
Hypothesis 35
Results and evaluation 35
PROJECT EVALUATION 43
Improvements 43
CONCLUSION 43
REFERENCES 44
APPENDICES 47

TABLE OF FIGURES

Figure 1 Example of an early stereoscope. Figure 2 An image to be viewed in the stereoscope. 4
Figure 3 The Sword of Damocles in use 5
Figure 4 Zeltzer's cube 8
Figure 5 Vive Lighthouses 10
Figure 6 Example of the play area 11
Figure 7 Unity's CameraRig prefab 12
Figure 8 NewtonVR camera rig 13
Figure 9 Simulation Lab angle 1 14
Figure 10 Simulation Lab angle 2 15
Figure 11 Plan of the simplified simulation lab 15
Figure 12 Tablet Concept 16
Figure 13 A blank unity project 17
Figure 14 Setting up the SDK for VRTK 18
Figure 15 Controller scripts game object with controller events and simple pointer scripts attached 18
Figure 16 A fully implemented interactable object 19
Figure 17 Completed simulation lab 20
Figure 18 The video playing 26
Figure 19 How control information is displayed in the usable version 29
Figure 20 The hand model in the immersive version 30
Figure 21 the control screen on the tablet in the immersive version 30
Figure 22 position of the grip button on the vive controller (number 8) 31
Figure 23 Control information usability results 35
Figure 24 Control information immersion results 36
Figure 25 Control information preference results 36
Figure 26 Interaction button usability results 37
Figure 27 Interaction button immersion results 37
Figure 28 Interaction button preference results 38
Figure 29 Controller representation usability results 38
Figure 30 Controller representation immersion results 39
Figure 31 Controller representation preference results 39
Figure 32 Movement system usability results 40
Figure 33 Movement system immersion results 40
Figure 34 Movement system preference results 41
Figure 35 Which version was more usable results 41
Figure 36 Which version was more immersive results 42
Figure 37 Overall best version results 42

INTRODUCTION

Project Background

As early as the 1830's stereoscopic viewers have been used for "virtual tourism" and the same technology is still used today for low budget VR such as *Google Cardboard* but, since the popularisation of videogames in the 1980's, the idea of a totally immersive virtual world became realistic instead of science fiction.

Films such as *Tron(1982)* and *The Lawnmower Man(1992)* show how ingrained this idea was in popular culture, and so, videogame companies started to experiment with Commercial Virtual Reality. Both Nintendo and Sega announced that they were developing Virtual Reality headsets in the early 1990's, however, Sega's hardware ran into technical issues and was never released. Nintendo's *Virtual Boy* was highly anticipated by consumers but the primitive graphics and uncomfortable viewing position meant that it was discontinued only a year after release.

It was obvious that the technology needed further advancements before Virtual Reality could be considered immersive in any way and games companies shifted their focus back to traditional consoles and games. The release of *The Matrix* in 1999 showed that there was a continued interest in Virtual Reality but it would take 15 more years for consumer Virtual Reality to become viable.

In June 2012 Palmer Luckey used the group funding site *Kickstarter* with the hope of funding his head mounted display named the *Oculus Rift*. *John Carmack*, founder of *Id Software,* then showed off a version of *Doom 3 BFG* running on a prototype of the Rift at E3, it was this that sky-rocketed the Oculus Rift Kickstarter to over 2.4 million dollars and also started the resurgence of consumer VR.

Today, many head mounted displays (or HMDs) exist, such as: *Google Cardboard* - which allows users to view 360 degree photos and play simple VR games and the recently released Playstation VR which seeks to bring VR to consoles. This project will focus on VR for PC, specifically the *HTC Vive*.

The HTC Vive uses two infra red sensors the track the location of the HMD and the motion controllers meaning that players can walk around the game world and interact with objects by simply picking them up. Games like *Rec Room* and *Job Simulator* use the Vive to make their experiences more immersive and unique. This new input method, while immersive, brings up questions in regards to the usability of the software as people who are used to more conventional input methods might find it hard to adapt and novice users could get confused.

Many developers are now facing the issue of trying to develop games that can immerse users in new and exciting worlds without making it frustrating and difficult to navigate and interact with objects. This project will explore the problem stated above and attempt to come up with a solution that allows games to find the right balance between immersion and usability.

Aim

The aim of this work is to analyse how immersion and usability affect player enjoyment in VR by creating two versions of an application for the HTC Vive, one which includes features to make it as usable as possible and one that includes features that make it as immersive as possible.

Objectives

- Research game engines and choose one that best meets my needs for this project.
- Create two scenes that can be used as the levels for the game.
- Implement mechanics that let players interact with objects and the game world in fun and interesting ways.
- Research ways to make these mechanics both usable and immersive.
- Evaluate the game play on each version by conducting a survey and analysing usability and immersion

LITERATURE REVIEW

Background

For many years VR (Virtual Reality) has been a sought after technology in a range of contexts, including entertainment and teaching, which has lead to the wide variety of research being conducted about the subject. A majority of the research covers possible usage in training, for example: simulators for surgeons, as well as the concept of presence in the virtual world and how to make users feel immersed in the experience. Both of these areas are integral to this work and will be covered in this section of the report. Due to the nature of this work it is imperative that the techniques used to make users feel present in the virtual world as well as how to make the application usable is fully understood so that each version of the application is fulfilling its purpose. The software that is produced will be used to teach medical students, therefore understanding how to successfully convey ideas and impart knowledge using virtual environments will be useful. This section will also attempt to describe what elements are involved in creating proficient VR software which will enable a smoother design and development phase.

Much of the research that has been done in the field of VR has been purely hypothetical until recently. The release of HMDs in recent years such as the oculus rift, HTC vive and playstation VR have shown the power of VR, but it is Google cardboard, which allows any Smartphone to become a VR headset, which has brought the technology to the mainstream. Moore (2016) explains that these technologies are iterations of devices that have existed for many years; however with global Smartphone users estimated to reach 2.1 billion by the end of 2016 *(*Number of smartphone users 2016), it is the vast availability of the technology that has allowed it to flourish. The high entry cost of PC and console based systems has meant that widespread adoption has been slow, however, due to their better tracking and higher fidelity they are ideal for use in treatment of physical or mental issues as well as for training and simulators, they are also much cheaper than older bespoke solutions that were used. (Merchant et al., 2014)

What is VR?

In a modern context a solid definition for virtual reality can be hard to pin down. Brooks (1999) writes that a virtual reality experience can be any which effectively immerses the user in a responsive virtual world and which allows for dynamic control of the viewport, while this is a good starting point for understanding the technology the definition can be said to include most modern videogames and other immersive applications.

Bishop and Fuchs' (1992) definition, of real time interactive graphics with three-dimensional models that gives the user immersion and direct manipulation in the virtual world, more closely

resembles the direction that modern VR has taken. Input in VR is one of the main areas which differentiates it from more traditional three-dimensional computer games and applications, control methods such as location tracked controllers or gloves allow users to directly interact with virtual objects as opposed to conventional control schemes which would let users interact with objects by pressing a button or key.

VR is, as Gigante (1993) states the illusion of participation in a synthetic environment rather than external observation of such an environment, this definition captures the other crucial component of VR that differs from other three-dimensional applications, which is the head mounted display. Computer monitors and televisions provide a monoscopic two-dimensional view of a three-dimensional world whereas a HMD contains two panels and so can create a stereoscopic image to allow users to feel present in the virtual world. Gigante (1993) goes on to say that VR relies on a head tracker display, which means it is required that real world body and head movements are reflected in the virtual environment.

History of VR

As stated above, VR was originally seen as any system that immerses users in a virtual environment. Panoramic paintings from as early as the 19th century fill the viewer's field of vision to make them feel present at a fictional or historical location. Alongside the development of panoramic paintings Charles Wheatstone demonstrated, in 1838, that the brain processes a separate image from each eye and combines them into a single three-dimensional view, he would go on to adapt this into the stereoscope which became extremely popular with the release of *the View-Master* in 1939. It is these two concepts, filling the field of view and presenting the image in three-dimensions that form the basis of all VR technology. (History of virtual reality. 2015)

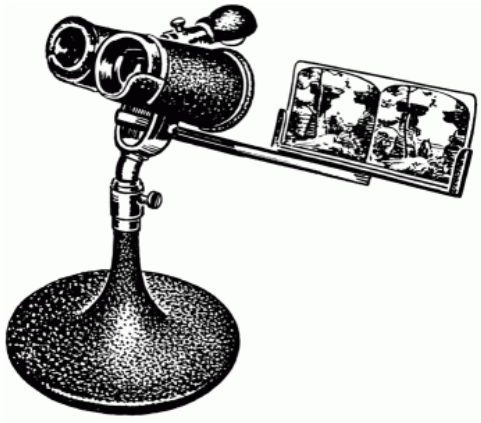


Figure 1 Example of an early stereoscope.

Figure 2 An image to be viewed in the stereoscope.

The 1960's brought with it a multitude of advancements for VR starting with Morton Heilig's *Sensorama*; a film which was presented in colour and stereoscopy as well as featuring binaural sound, scent, wind and vibration effects. Immersive elements such as were offered by the *Sensorama* would become important aspects of future VR systems even though it had no interactive elements. *Headsight,* the first precursor to the modern HMD, was developed by two Philco Corporation engineers in 1961. The system didn't include any virtual elements but was the first device to include motion tracking and was used to allow for remote viewing of dangerous situations by the military. (Mazuryk and Gervautz 1996) (*History of virtual reality*. 2015)

Ivan Sutherland described the *Ultimate Display* in 1965. His vision was of a system that could simulate the real world to such a degree that it was indistinguishable from reality. He described a HMD which would display a virtual world with 3D sound and tactile feedback, it would be powered by hardware which could create and maintain the virtual world in real time and give users the ability to interact with objects in the virtual world in a realistic way. Sutherland, along with his student, Bob Sproull, created the *Sword Of Damocles* in 1968. It was the first VR HMD that was connected to a computer, the headset was so huge that it had to be suspended from the ceiling, it was only capable of producing wireframe rooms and objects. (Mazuryk and Gervautz 1996) (*History of virtual reality*. 2015)

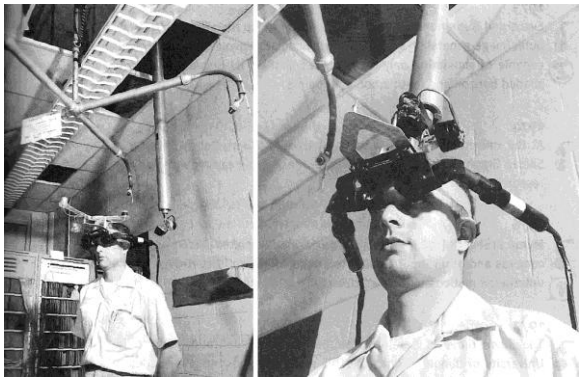


Figure 3 The Sword of Damocles in use

The term "virtual reality" wasn't used until 1987 when the founder of the visual programming lab, Jaron Lanier, coined the phrase. Lanier would go on to create a range of virtual reality headsets and VPL would be the first company to sell them to the public but the high price of the hardware meant that few could afford the technology. Because of the immense cost, personal ownership of VR still wasn't viable, however with the rise of arcades, the public could experience VR for the first time. The Virtuality group released a range of VR arcade games

starting in 1991. Players would use a HMD to experience real-time immersive 3D visuals, some units also allowed for networked multiplayer games. (*History of virtual reality*. 2015)

The push to sell VR directly to consumers started in 1993 when SEGA announced a VR headset for their Genesis home console. The HMD would feature head tracking, stereo sound and LCD stereoscopic displays. Technical difficulties meant that the product would never get released. Nintendo was able to release a portable VR gaming system in 1995, called the *Virtual Boy,* which was marketed as the only portable system that could display true 3D graphics at the time. The system was extremely large and uncomfortable to use and displayed games in only black and red. Both of these systems were massive failures for their respective companies and showed a need for advancements in graphics processing, display and input technology before VR could be viable for consumers. (*History of virtual reality*. 2015)

VR in the modern world

Originally Virtual reality systems were grouped into 3 categories known as the 3 levels of immersion in VR systems. Mazuryk and Gervautz (1996) explain that Desktop VR is the lowest level of immersion, also called "Window on World" systems; they use a conventional monitor to display a monoscopic view of the world. Fish Tank VR, a technique which uses head tracking with a traditional monitor to give the illusion of depth, is more immersive than desktop VR systems but still lacks many qualities necessary for fully immersive VR. It is named "Fish Tank" because users are not inside the virtual world but are looking into the 3D environment, like looking into a fish tank. Immersive systems use a HMD that supports a stereoscopic view of the virtual world and allows realistic representations of head movements using motion tracking. Immersive systems may also be enhanced by the use of 3D audio, haptics or motion tracked controllers.

Today there is a wider divide between VR systems and 3D applications displayed on a monitor, therefore, only products that classify as immersive systems would be considered virtual reality. This shift in thinking means that a new way to categorize VR systems was needed. Over the last few years, as more and more systems release, two clear categories of VR have emerged.

Mobile VR uses a shell of a HMD with minimal hardware into which a Smartphone can be placed that will deal with all of the graphics processing as well as displaying the image. These systems are completely self contained and portable as well as extremely cheap as most people already have a Smartphone and the housing can cost as little as a few pounds. Due to the limits in processing power of Smartphones the applications of these devices are limited to 360 degree pictures and videos as well as simple games; complex environments cannot yet be rendered. Mobile VR have no way to track a user's location only rotation using the phones internal gyroscope is supported, additionally there is no native input so a separate device is required.

Desktop VR uses a HMD that is connected to a PC or games console. The HMD contains a display for each eye as well as other technology such as gyroscopes and motion tracking hardware. The PC handles all of the graphics processing so the detail and complexity of the virtual world is only limited by graphics processor in the user's PC. Location and positional tracking is supported as well as dedicated input methods which allow for greater immersion and interaction with the virtual world. These systems are not at all portable as they are wired into the PC and rely on it to function, the wire can also cause users to lose immersion if they trip or get tangled in it, although, HTC has recently announced a wireless add on for its Vive HMD (Mendelsohn 2016). The price of the HMD as well as a PC powerful enough to run VR applications is the biggest downfall of Desktop VR.

What makes an immersive VR application?

While all of the VR systems described above are designed to immerse users as fully as was possible at the time of creation, it is up to the software that runs on these systems to make use of their features as well as any other techniques that will keep the user engaged with the virtual world. Zeltzer's (1992) work on telepresence, the feeling of being elsewhere, gives valuable insight on this subject. He proposes that as well as the VR system, game/graphics engine and any peripherals, there are three key components of any virtual environment:

1. A set of models of objects and processes to be simulated.
2. Some way to modify the state of the models throughout the simulation.
3. Channels that allow the user to experience the simulated events through one or more sensory modalities.

These components are called autonomy, interaction and presence respectively.

Autonomy

At one end of the spectrum a virtual environment might contain completely passive models that don't react to the user, or each other, at all, characters and objects are simply placed and rendered, they have no scripting of any kind to allow for interaction. At the other end of the spectrum are virtual actors and items capable of complex actions and interactions such as reactive planning , for example complex AI or a torch catch fire if exposed to a flame. Autonomy then, is a qualitative measure of the ability of a computational model to act and react to simulated events and stimuli, ranging from 0 for the passive geometric model to 1 for the most sophisticated virtual environment.

Interaction

In this context, interaction means the amount of access the user has to model parameters at runtime for example allowing users to pick up objects inside the simulation or throw an object and have it knock over other objects. The range is from 0 which represents a completely static environment with no user-model interaction, to 1 for comprehensive, real-time access to all model parameters.

Presence

Presence must first and foremost be separated from immersion. Immersion describes the system's ability to provide a highly detailed environment and the tools to interact with it. Presence is the extent to which a user of a virtual environment allows them self to be convinced by it. The range is from 0, the user is fully aware they are in a simulation and unable to convince them self otherwise, to 1, the user is entirely present in the virtual world.

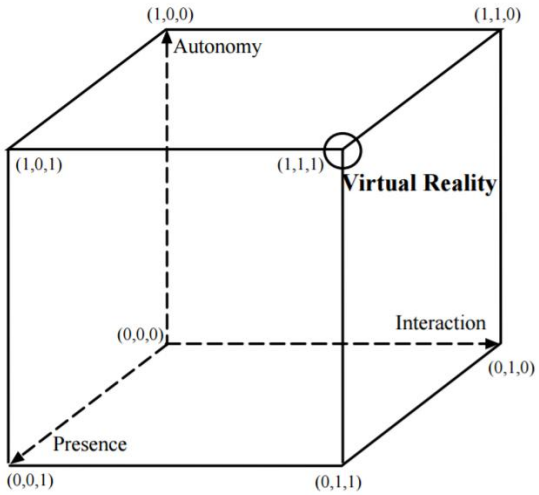


Figure 4 Zeltzer's cube

As shown in figure 3 a high amount of all 3 components is needed to create a convincing virtual reality experience. Position (1,0,1) on the diagram can be said to be a system where the user feels present in a complex Environment, in which models can react to each other, but the user has no control over, or interaction with, the system, a good example of this is Valve's dota 2 VR spectator mode which allows users to watch people play the game in VR. (0,1,1) would be a system where users could move around the virtual environment and pick up or knock over objects but no other more complex actions or reactions are possible. Clearly both previously described systems still fall under the umbrella of Virtual reality, however, position (1,1,0) could describe any 3D application or game with complex scripting and some sort of user input, therefore, presence is the most important factor when it comes to creating convincing VR

Assessing presence in real world VR applications

Steed et al. (2016) conducted an experiment to understand what affects presence in VR systems. They created a mobile VR application for Samsung gear VR and Google cardboard which features a singer in a bar. Three different pairs of conditions were tested; self-avatar versus no self-avatar, user asked to tap along versus user not asked to tap along and eye contact versus no eye contact, at the end of the experience a small box is dropped onto the user's avatar. A questionnaire was used to collect data about the users' experiences.

The results showed that participants' feeling of presence was not affected by the singer looking at them. Possibly the low resolution of the HMD that was used hindered the users' ability to see the singer's eye gaze, but, the actual reason for this is unknown and perhaps worthy of extra research.

The singer asking the participants to tap along negatively affected question 3 which was "During the time of the experience, which was strongest on the whole, your sense of being in virtual bar, or of being in the real world? 1 being the real world, and 7 virtual bar." It is theorized that because the participants hand motion was out of synchronization with the avatars motion it actually drew attention to the fact that they were in a virtual environment instead of making the experience feel more interactive as was the intention.

Participants that had an avatar felt like they could have been hurt when the box fell in VR which suggests that the self avatar was having an important impact on their experience.

DESIGN

Required Resources

This project requires a range of components to function and each contains multiple competing resources, this section will describe these components and explain why choices were made in regards to which was chosen in each category.

Head Mounted Display

As described in the literature review the head mounted display or HMD contains the panels which display the 3D graphics and allow users to view the world in 3D, they also contain hardware that tracks the users' location and rotation to accurately reflect their movements in virtual reality.

Obviously the choice of HMD will drastically affect this project as each work differently and so could affect usability or immersion, however, as the software being produced is for use in the nursing department, the HTC Vive had to be used as it is the only HMD that the department has access to.

The Vive has the advantage of being the only HMD to feature room scale VR as default, this means that users can physically walk around the environment, it is also the only HMD to provide tracked controllers which means that the controllers movements are reflected in VR, both of these features open up more possibilities and potential problems for usability and immersion. To achieve room scale VR the Vive uses two "light houses" which emit infrared light to detect the position of the HMD and tracked controllers within a space, this space is called the play area.

Figure 5 Vive Lighthouses

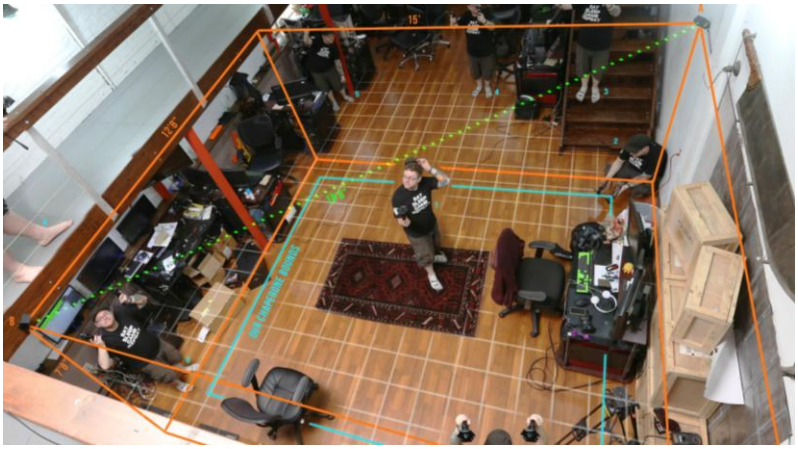


Figure 6 Example of the play area

Steam VR

The Steam VR SDK allows for the development of Virtual Reality games and experiences that will work on all major Virtual reality headsets. It comes packaged with many newer versions of game engines and if not can be downloaded as a plug-in which can be included in Virtual reality projects. It includes prefabs for things like virtual reality cameras, which, when dragged into a scene become the default camera allowing the use of a HMD including position and rotation tracking. The SDK also includes scripts to display tracked objects in the virtual world.

Steam VR requires a space of 2 meters by 1.5 meters and a PC with the following specs; Intel™ Core™ i5-4590 or AMD FX™ 8350, NVIDIA GeForce™ GTX 1060 or AMD Radeon™ RX 480, 4 GB RAM or more

Game engine

A game engine is required to create the world and mechanics for the application. At the time of this project three popular game engines support VR development for the HTC Vive: Unity, Unreal and Cryengine. Each engine uses the steam VR SDK differently so it is important to experiment with each one and find which is the easiest and most efficient to use so that the development phase goes as smoothly as possible.

Unity was the first engine to have official support for the Steam VR plug-in which means that official and unofficial tools and information are widely available to help streamline and improve VR development. Unity documentation is clear and easy to find, it supports both C# and Java programming languages and Microsoft visual studio integrated development environment (IDE).

The Steam VR plug-in doesn't come packaged with this engine but is easily downloaded for the unity asset store and can be included in any project. In Unity the SteamVR plug-in comes with a CameraRig prefab which, when dragged into a scene sets up the HTC Vive's play area as well as

maps the location of the controller models to the real world location of the controllers within the play area.

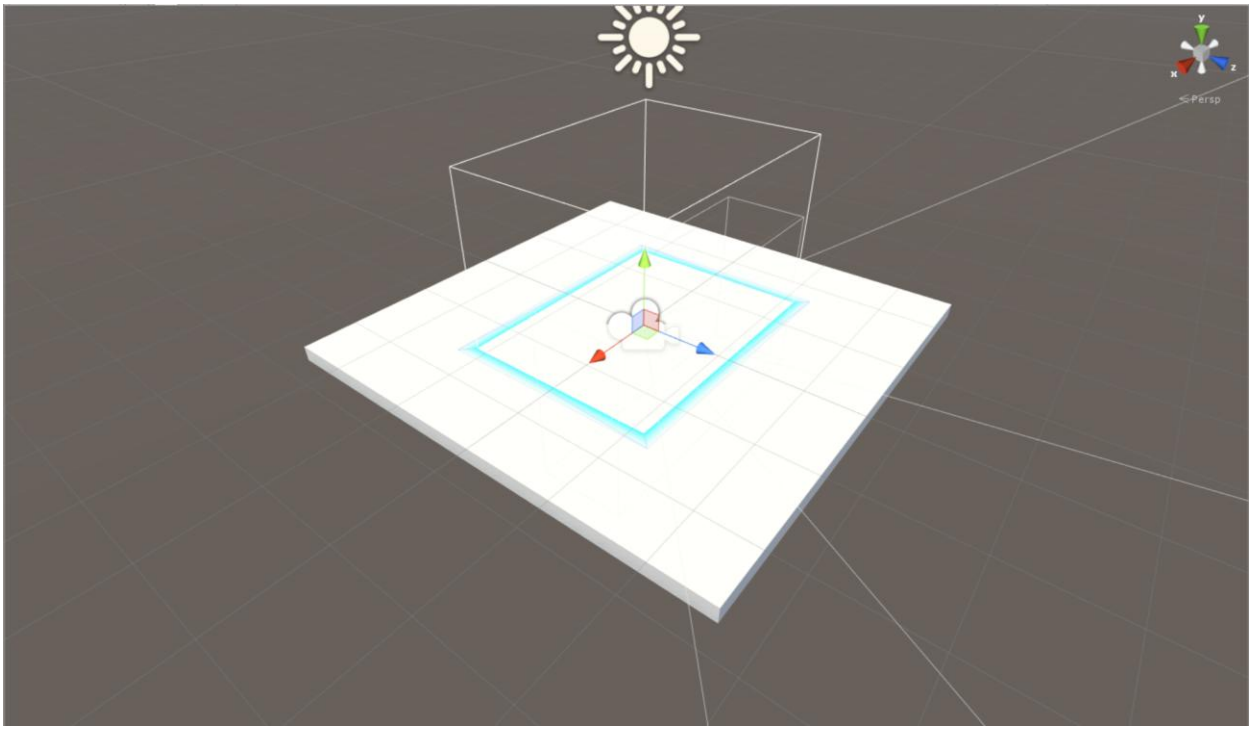


Figure 7 Unity's CameraRig prefab

Unreal engine 4 is an extremely advanced game engine that produces some of the most technologically impressive modern games. It supports C++ development as well as its blueprint based visual scripting system which could simplify and speed up some aspects of development. The documentation for unreal is isn't as comprehensive as unity and can get quite confusing as aspects of the engine change between versions so some solutions no longer work.

The Steam VR plug-in is built directly into every version of unreal since version 4.8 but example scenes and blueprints didn't come with the engine until version 4.13, as with the documentation information on VR development in the engine isn't as widely available as unity and there are less third party tutorials.

Cryengine is similar to Unreal in that it is extremely advanced and can create stunning experiences, unfortunately its documentation is even more lacking than unreal engines. In general Cryengine is not very user friendly; it is a relatively newly released to the public and as such needs some usability improvements. In testing it also seemed to have stability issues and would crash quite often.

Cryengine requires a plug-in file to be added to the project and configuration files to be edited to enable VR support. There is one example VR scene but no information on how to recreate it

or expand on it. Overall this engine is still in the early stages of VR support and doesn't allow for key Vive features such as room scale or tracked controllers

For this project Unity was chosen as at the time that the project was started the VR support was much more comprehensive that the other options. Unity also has a large range of tools available on its asset store such as VRTK and Newton VR that aid VR development on the platform. Unreal has made many improvements to its VR support over the last few months and if this project was to be redone that would probably be the engine of choice.

Tool Kit

As motioned above, Unity gives developers access to many tool kits to aid with development. These are collections of scripts and prefabs that add functionality that the engine doesn't come with natively. It is important for this project to use one of these tool kits as the focus of the work is to analyze and assess interaction and locomotion implementations, not to create these systems, so using one of these tool kits allows for a more streamlined development process so that plenty of time is available for testing and evaluation. In terms of VR two main tool kits exist for unity Newton VR and VRTK.

Newton VR is an extremely comprehensive set of scripts and prefabs that deal with interactions in VR. It allows for many different forms of interaction with objects within the virtual world, for example, basic pickup using a button on the controller, dial rotation by rotation the controller and object reorientation on pickup to allow for items like swords to always be picked up at the handle. A custom camera rig prefab is used with this tool kit which shows the scale of the player in relation to objects in the scene, this could cause problems as moving and changing variables in the camera rig is a bit different using this tool kit.

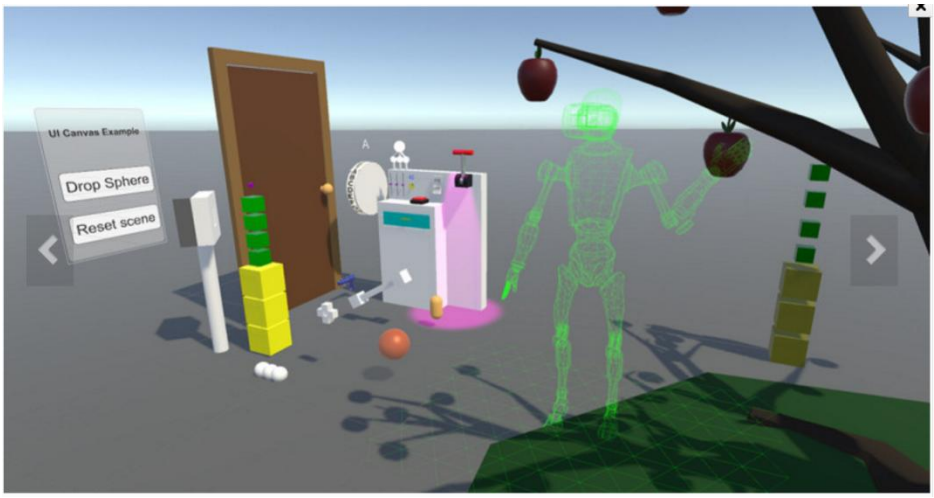


Figure 8 NewtonVR camera rig

Unfortunately this tool kit doesn't come with any locomotion options at all so all interaction has to take place inside of the initial play area. During testing another tool kit was used alongside Newton VR called "HTC Vive Teleportation System with Parabolic Pointer" which allows for movement around a larger space than just the play area, unfortunately this caused issues and if an item was picked up and the user teleported the item would disappear.

VRTK or Virtual Reality Tool Kit doesn't have as many or as detailed interactions as Newton VR, with only the basic methods such as, press button to pick up and move the controller in a circular motion to rotate but it contains many locomotion systems from teleporting to using the controller track pad to move the play area. Because all these systems are built within the same tool kit they work extremely well together and allow for movement arount the world while interacting with items at the same time. For these reasons VRTK was chosen for this project, however the author feels that there is a chance that the interactions in Newton VR may be easier to understand and more usable, as such Newton VR might have better fit the needs of this project if it wasn't vital that users are able to move around the game world.

Level Design

The application created for this project is also to be used in the nursing department as a teaching aid for medical students so it is important that the level resembles the nursing departments simulation lab as closely as possible, because of this many photographs were taken of the simulation lab in order to faithfully recreate all aspects of it.

Figure 9 Simulation Lab angle 1

Figure 10 Simulation Lab angle 2

With information about the lab collected the room needed to be simplified. It isn't practical to create every piece of equipment in the room as it would take a lot of time and could also cause problems during testing, as participant could experience sensory overload if there are too many things to do and interact with. A plan needed to be created that only includes some essential features of the room that could be used as interaction points during testing.

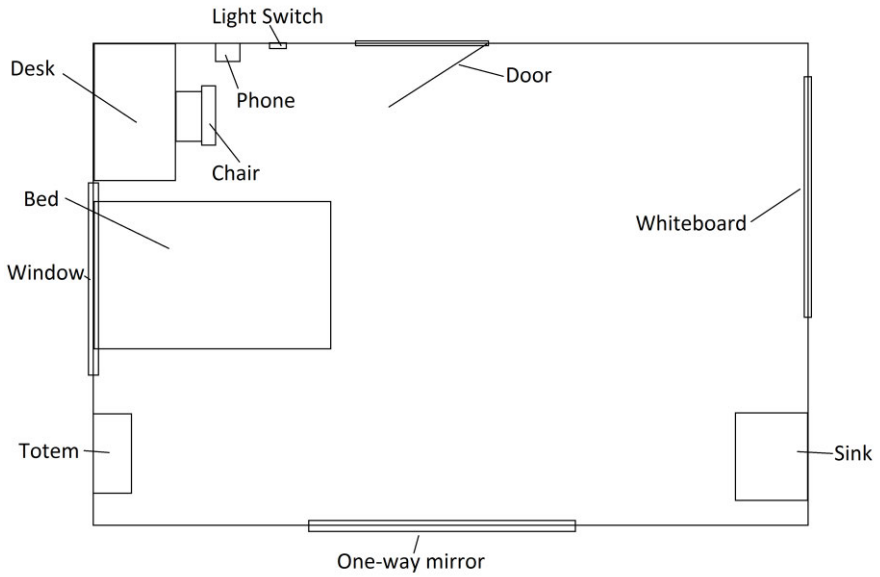


Figure 11 Plan of the simplified simulation lab

As figure 11 shows the plan contains different types of interactions, such as a door, a dimmer switch and tap that need to be rotated, a chair which can only be moved along the floor and a phone that can be picked up freely, these varying ways of interacting with objects will hopefully highlight usability problems for the participants. The interactable objects were chosen not just because of their varying interaction styles but also because of their location in the room, placing objects around the edges of the room will force to participants to explore the whole room meaning that they will use the locomotion system a decent amount whereas if all of the interactable objects were inside the play area there would be no reason to use the locomotion system at all.

For the applications real world use the player would have to be given information such as controls and how to use pieces of equipment, so a concept was created to allow users to see a virtual tablet which would give them that information.

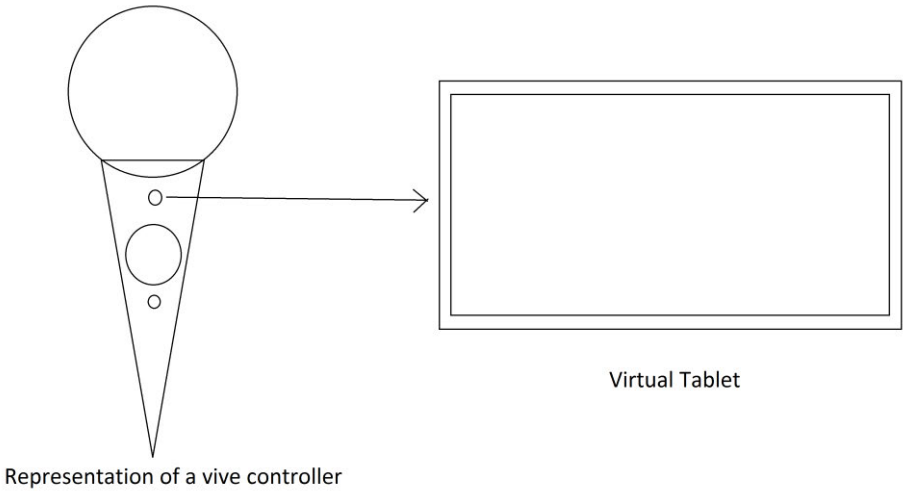


Figure 12 Tablet Concept

The idea is that at any point players can press the menu button on the controller and the controller would disappear and be replaced by a tablet. The tablet would feature a menu system that would allow users to bring up a screen that shows the controls for the application or view videos about each piece of equipment in the lab. Another system would be implemented that would vibrate the controller when a user got close enough to a piece of equipment, this would indicate that a video is available for the equipment and when the user presses the menu button to bring up the tablet the corresponding video would automatically play. This concept also allows for a more immersive way of presenting control information that can be used in the evaluation of the application.

IMPLEMENTATION

Setting up the project

Before the level is created and objects are added some setup is required to ensure that the VR components are functioning correctly and SteamVR and VRTK are working. A new scene can be created by right clicking inside the assets folder and selecting create > scene by double clicking the newly created scene it is loaded and it only contains a camera and directional light.

Figure 13 A blank unity project

This default camera can be deleted as a VR camera rig, as mentioned previously, is needed to view the scene with the HTC Vive. The SteamVR plug-in needs to be downloaded from the unity asset store and imported into the project, now the camera rig prefab can be dragged into the scene.

Setting up VRTK

The VRTK plug-in needs to be downloaded from the asset store and imported into the project. VRTK requires that an empty game object, called VRTK, be created that will contain scripts for setting up the system. Dragging the VRTK_SDK manager script onto the empty game object allows the SDK for the project to be chosen so that VRTK knows which HMD it is dealing with, for this project that will be SteamVR. Below the SDK selection is a series of boxes that need to be populated with game objects but simply pressing the auto populate button will find these game objects in the SteamVR CameraRig prefab.

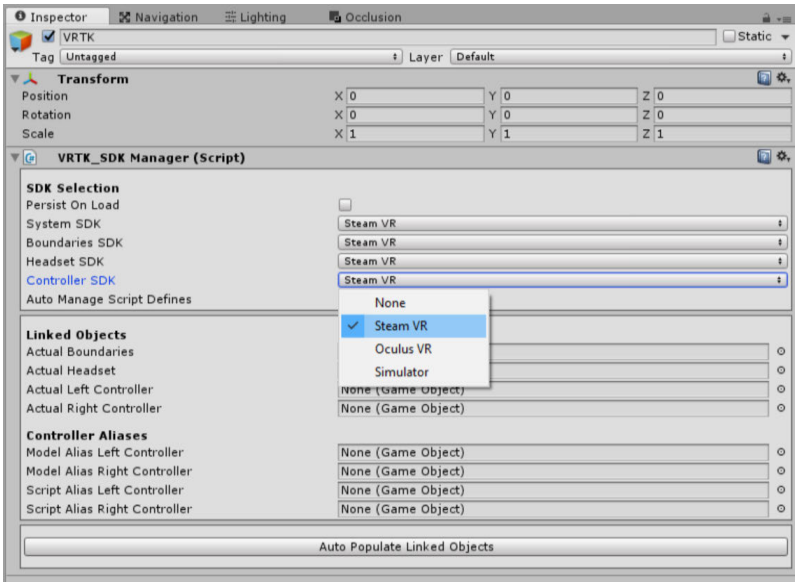


Figure 14 Setting up the SDK for VRTK

Creating two more empty game objects as children of the VRTK empty game object will allow scripts for the controllers to be used such as listening for controller events or allowing for a beam to be projected out of the controller for teleporting. Attaching the interactGrab script to these two controller scripts game objects allows objects that are interactable to be picked up.These two empty game objects need to be dragged into the script alias section of the VRTK SDK manager script that is attached to the VRTK game object.

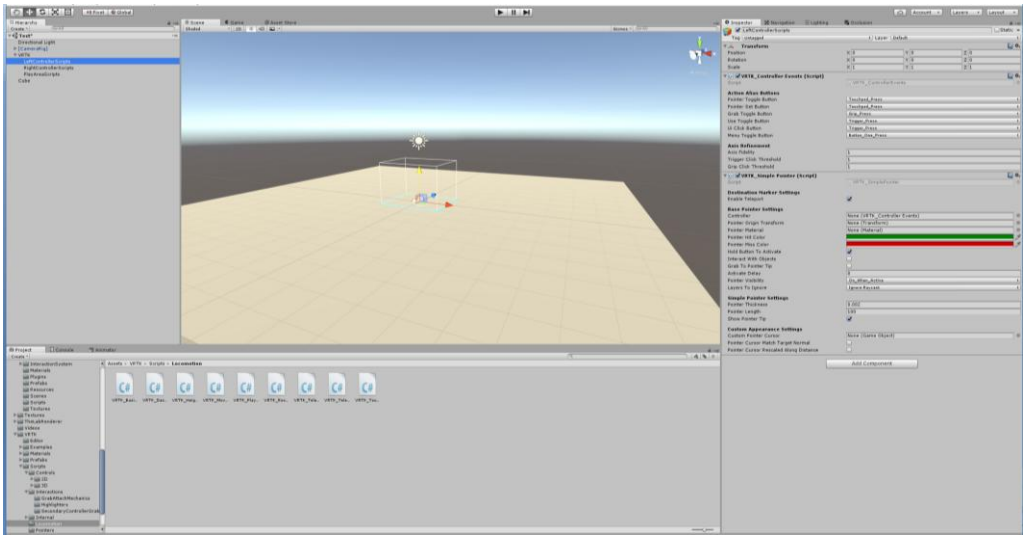


Figure 15 Controller scripts game object with controller events and simple pointer scripts attached

Adding a third empty game object and naming it play area scripts allows for a locomotion script to be added to the scene for this project the basic teleport script was used. All the systems are now implemented to allow for teleporting around the scene.

Setting up interactable objects

Every interactable object in VRTK has to have a rigidbody, meaning that it is a physics object and can be push or hit by other objects in the world. Interactable objects must also have a collider attached to them so that VRTK knows when the controller is touching the object and it is able to be picked up.

Attaching the VRTK interactable object script to an object allows it to be touched grabbed and used. Touching an object simply means moving the controller inside of that object this could have the effect of changing the objects colour or making the controller vibrate. Grabbing the object means picking it up usually by touching it first then holding a button on the controller. Using an object means first grabbing the object then pressing a different button on the controller to activate some sort of use, for example, firing a bullet out of a gun that is being held.

After the interactable object script has been attached 2 attach mechanics have to be chosen for the object, one determines what happens when the object is first picked up, this could be a fixed joint that connects the controller and object with a joint to allow the object to be moved or a rotator track that applies rotational force to the object in the direction that the controller moves. The other attach mechanic deals with situations in which the object is already being grabbed but then is also grabbed by the other controller this could swap the object to the other controller or it could allow the object to be scaled by pulling both controllers apart.

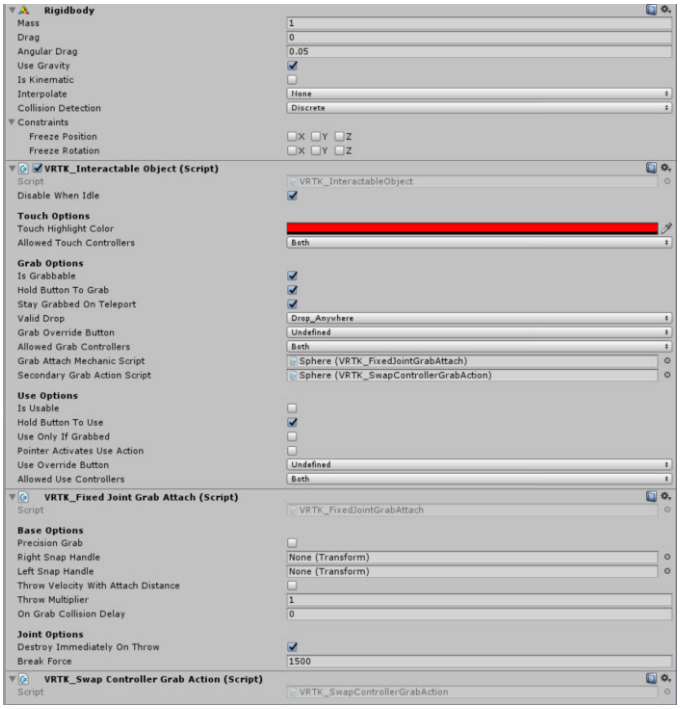


Figure 16 A fully implemented interactable object

Creating the Simulation Lab room

To eliminate any issues that might interfere with the evaluation of the application, such as performance problems, it was decided that any walls or features that are only ever viewed from one side could be implemented as planes instead of 3D geometry. The same precautions were taken when selecting assets to use in the level and low poly assets were used where available; most were acquired for free from Turbo Squid, available at - https://www.turbosquid.com.

Figure 17 Completed simulation lab

The model of the man lying in the hospital bed was created in Autodesk character creator and uses animations supplied by Kevin Tan. The hands and animations used in the immersive version of the application were bought from the unity asset store and imported into the project. Figure 17 shows that the totem that was included in the plan (figure 11) isn't present in the final version of the application, the object, which includes features like a heart rate monitor and air canisters, was custom made to fit the space so there were no 3D models available to use, on top of this the types of interactions required for it are extremely complex. It was decided to leave out the totem so that development of it didn't eat into time that could be used for the evaluation.

A problem immediately became clear when the basic walls and floor of the room were implemented. Any object with a collider could be teleported to meaning that users could teleport to the wall and end up outside the room which could be disorienting and immersion breaking. A script called VRTK_PolicyList was used to specify valid teleport locations using tags, the floor was given the tag Include and it was now the only thing players could teleport to, however, it was still possible to teleport into the extreme corner of the room which would cause the same issues stated above. Cubes without a mesh renderer were added all around the edge of the level so that users could not teleport all the way into the corner.

Custom scripts

All of the basic interactions such as picking up objects or opening doors can be achieved just using VRTK scripts applied to objects, but, for more complex interactions additional custom scripts were required to achieve the desired effect.

Dimmer switch

```
public class DimmerScript : MonoBehaviour {
    public Light[] lightArray;
      void Update () {

            foreach (Light light in lightArray)
            {
                light.intensity = this.transform.eulerAngles.z/180;
            }
    }
}
```

This script is used on the light switch and allows the light intensity to be increased and decreased by rotating the dimmer switch. An array of lights in the scene is created, when the script is attached to a game object the size of the array can be entered, which corresponds to the number of lights in the scene then each light can be dragged in so that the script affects all lights at the same time. Every update the script changes the intensity of every light in the scene to match the value of the Euler angle of the dimmer switch divided by 180, this is because light intensity in unity goes from zero to eight so even a slight movement of the dimmer switch would set the intensity to max if it wasn't scaled down. An invisible cube is used to make sure that the dimmer switch can't be rotated below 0 degrees as this would cause the scene to go from completely dark to max brightness straight away.

Replacing the phone after a small amount of time

The phone is hung on the wall in the corner of the room and uses a simple pick up mechanic which includes an additional feature of rotating and positioning the phone correctly for each hand so it feels more like picking up a phone in real life. The phone can be thrown across the room and is easily lost so a system had to be put in place to put the phone back on the holder after a few seconds if it isn't being interacted with.

```
public class replaceAfterGrab : MonoBehaviour
    {
        public Rigidbody rigidbody;
        public Transform Pos;
        private int time = 0;
        private bool grabbed;
        private bool firstTimeGrab;
        private VRTK_InteractableObject interact;
        // Use this for initialization
        void Start()
        {
            firstTimeGrab = false;
            rigidbody.isKinematic = true;
            interact = this.GetComponent<VRTK_InteractableObject>();
        }

        // Update is called once per frame
        void Update()
        {
            grabbed = interact.IsGrabbed();
            if (grabbed == true)
            {
                firstTimeGrab = true;
                time = 0;

            }
            if (grabbed == false && firstTimeGrab == true)
            {
                rigidbody.isKinematic = false;
                time += 1;

            }
            if (this.transform.position != Pos.position && grabbed ==
false && time > 1000)
            {
                time = 0;
                this.transform.position = Pos.position;
                this.transform.rotation = Pos.rotation;
                rigidbody.isKinematic = true;
                firstTimeGrab = false;
            }
        }
    }
}
```

The script takes in the rigidbody of the phone which allows physics to be turned off and on for the object, it also takes in a transform called Pos which is the position that the phone will return to if it isn't being interacted with. An integer called time is declared that counts up and when it hits a certain value the phone will return to the location of Pos. The script uses two Booleans grabbed and fistTimeGrabbed, the reason this is necessary is that time counts up when the phone isn't grabbed but when the application is started the phone isn't grabbed so the timer will start to count up straight away and when the user picks up the phone it will teleport back to its starting location as soon as the user lets go of it. rigidbody.isKinematic sets weather the phone is affected by physics or not, it is set to true when the phone is on the holder so that it doesn't just fall to the ground and is set to false at all other times.

Running water script

```
public class waterRun : MonoBehaviour {
    private ParticleSystem water;
    private AudioSource sound;
    public GameObject tapLeft;
    public GameObject tapRight;
    private float rateR;
    private float rateL;
    private float finalRate;
    // Use this for initialization
    void Start () {
        water = GetComponent<ParticleSystem>();
        sound = GetComponent<AudioSource>();
        sound.volume = 0;
    }

     // Update is called once per frame
     void Update () {
        var em = water.emission;
        rateR = tapRight.transform.eulerAngles.y;
        if (tapRight.transform.eulerAngles.y > 355)
        {
            rateR = 0;
        }
        rateL = tapLeft.transform.eulerAngles.y;
        if (tapLeft.transform.eulerAngles.y > 355)
        {
            rateL = 0;
        }
        finalRate = (rateL + rateR) * 5;
        sound.volume = (rateL + rateR)/ 100;
```

```
        if (finalRate > 0 && finalRate < 20)
        {
            finalRate = 0;
        }
        em.rate = finalRate;
    }
}
```

The waterRun script creates particles and sounds when the taps are turned on in the sink. The script is attached to the particle system so it only needs to take in the left and right taps so that it can access their rotations. The rotations of both taps are taken into account so that the flow rate or amount of particles is affect by both at the same time. The sound volume is controlled by the same value however like with the dimmer switch the value had to be scaled down to match the scale of the volume variable in unity.

Displaying the tablet

```
public class showTablet : MonoBehaviour
    {
        public GameObject tablet;
        public GameObject otherController;
        private int count = 0;

        // Use this for initialization
        void Start()
        {
            tablet.active = false;
        }

        // Update is called once per frame
        void Update()
        {
            count++;
            if
(this.GetComponent<VRTK_ControllerEvents>().buttonOneTouched == true
&& tablet.active == false && count > 45 &&
otherController.GetComponent<VRTK_ControllerActions>().IsControllerVis
ible() == true)
            {

this.GetComponent<VRTK_ControllerActions>().ToggleControllerModel(fals
e, null);
                tablet.active = true;
                count = 0;
```

```
            }
            if
(this.GetComponent<VRTK_ControllerEvents>().buttonOneTouched == true
&& tablet.active == true && count > 45)
            {

this.GetComponent<VRTK_ControllerActions>().ToggleControllerModel(true
, null);
                tablet.active = false;
                count = 0;
                this.GetComponent<playVideo>().Start();
            }
        }
    }
}
```

This script hides the controller model and sets the tablet game object to active. The script is attached to the controller so two instances of this script are active at all times, one for the left controller and one for the right, this was done to improve usability so that weather users are left or right handed they can use the tablet. A usability problem was encountered early on in the development of this script where users could display the tablet on both controllers and be unable to locate the button to hide it again, as the controller model was no longer being displayed, leavening them unable to use the controller. The script now finds out if the other controller is visible before displaying the tablet, if the other controller is invisible then the tablet is being displayed on that controller so there is no need to display it on this one too. The script originally just listened for the menu button to be pressed but this meant that if the button was held the tablet would constantly appear and disappear over and over again so a timer was implemented to limit how many times the tablet can be displayed to twice a second.

Playing the video

As stated in the planning section a series of videos were to be implemented to explain pieces of equipment to students in an immersive way. A script was written to allow for this functionality but only one video is available in the final version of the application.

```
public class playVideo : MonoBehaviour {
    public GameObject screen;
    public MovieTexture movie;
    public Material videoMat;
    public Material defaultMat;
    public GameObject play;
    public GameObject replay;
    public AudioSource sound;
    private bool playedOnce;
```

```
    // Use this for initialization
    public void Start () {
        replay.SetActive(false);
        play.SetActive(true);
        screen.GetComponent<Renderer>().material = defaultMat;
        playedOnce = false;
        movie.Stop();
        sound.Stop();
    }
    public void Play() {
        screen.GetComponent<Renderer>().material = videoMat;
        screen.GetComponent<Renderer>().material.mainTexture = movie
as MovieTexture;
        sound.clip = movie.audioClip;
        movie.Play();
        sound.Play();
        play.SetActive(false);
        replay.SetActive(false);
        playedOnce = true;
    }
    // Update is called once per frame
    void Update () {
        if (!movie.isPlaying && playedOnce)
        {
            movie.Stop();
            sound.Stop();
            replay.SetActive(true);
            screen.GetComponent<Renderer>().material = defaultMat;
        }
     }
}
```

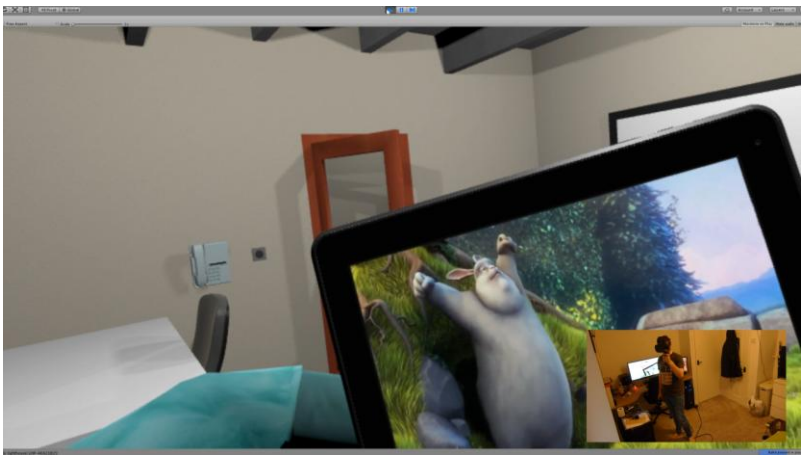


Figure 18 The video playing

The script makes use of the UI system that is built in to unity and VRTK to change the material of the tablet screen to a video material when the play button is pressed. VRTK allows the beam used for teleporting to also be used to interact with UI elements in unity and unity allows UI elements to run functions within scripts when the UI element is pressed. The play button is set up to run the Play() function within this script when it is clicked.

Better rotation script

By default, when rotating objects using VRTK, users have to move the controller in a circular motion. In reality using a tap or a dimmer switch requires rotating the hanfd not moving it in a circular motion. This disconnect between the virtual world and the real world caused problems in early tests of the application and users had trouble adapting to using that method.

```
public class grabRotateX : VRTK_TrackObjectGrabAttach
    {
        public GameObject applyforce;

        public override void ProcessFixedUpdate()
        {
            Vector3 rotateForce = new Vector3(trackPoint.rotation.z -
initialAttachPoint.rotation.z, 0,0);

grabbedObjectRigidBody.AddForceAtPosition(rotateForce/1000,
applyforce.transform.localPosition, ForceMode.VelocityChange);
        }

        protected override void SetTrackPointOrientation(ref Transform
trackPoint, Transform currentGrabbedObject, Transform controllerPoint)
        {
            trackPoint.position = controllerPoint.position;
            trackPoint.rotation = controllerPoint.rotation;
        }
    }
```

The script uses the current rotation of the controller minus the rotation of the controller when it first started interacting with the object to add force to that object. It allows users to simply rotate the controller and the object will reflect that rotation.

TESTING

Presence and usability, to some extent, are subjective in nature. Experience or just personal taste can affect what users find appealing and what immerses them in VR. If the author tested the application themselves it would only reflect what they find to be usable and immersive and as an experienced VR user it would not give insight into how novices experience the application, which is why testing for this application needs to be done with participants who can try the application and give their opinions on the mechanics and features of the game.

This section will discuss the testing methodology used for this project as well as displaying and analyzing the results.

Aim

To test how usable and immersive two versions of the same application are and to compile feedback about what techniques to use in the future and improvements that could be made.

Testing methodology

Two version of the application were created, one that represents a usable application and one that represents an immersive application. Both versions are exactly the same in terms of content and features but contain key differences to create the desired level of immersion and usability. Each features its own way of presenting information about the controls, a unique button used to pick up and interact with objects, a 3D model that represents the controller and a movement system that allows users to navigate the space.

The "Usable" application

This version uses a 3D model that is as close as possible to how the controller looks in real life; it will even reflect button presses and realistically compress the trigger the same amount as the user is pressing the trigger on the controller. It should be easy for participants to find buttons on the controller using this version as all they have to do is look at the controller in VR. It could, however, sacrifice immersion as users are always aware that they are in VR because they can see the controllers at all times.

Each button on the controller has a Label floating in mid air with arrows pointing to them; the labels explain what the buttons do. Again, this technique should make it very easy for the users to understand the controls but could get in the way later on when they are used to the controls and might end up becoming distracting.

Figure 19 How control information is displayed in the usable version

The trigger is used to interact with objects in the usable version. When holding the vive controller the index finger rests on the trigger, making it easy to push and keep compressed, meaning that it is less likely that users will drop objects while they are trying to move them around. The index finger also never has to move away from the trigger, so users won't be fumbling around trying to locate the button on the controller, which could happen if the action was mapped to one of the face buttons.

Basic beam teleportation is used for getting around the room in this version. The user can hold down the big track pad on the controller (which can be seen in figure 19) to display a red beam, pointing the beam at the ground turns it green, the user can let go of the button when the beam is green to teleport to that location. This is the most popular method of locomotion in VR so if users have played any VR games or experiences before it will feel very familiar. Admittedly, it could be slightly harder for complete novices to use.

The "immersive" version

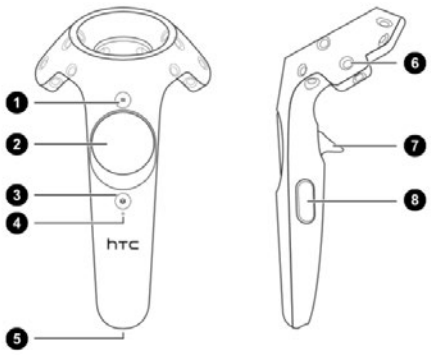
A hand model was used for this version that included a grabbing animation for when the user presses the interaction button. The hands should help users to feel more present in the simulation lab. newer users might find it difficult to remember where buttons are located on the controller as they have no way to look at it, In this case the hands might actually break immersion as the user gets frustrated and can't do what they want to do.

Figure 20 The hand model in the immersive version

When the user presses the menu button to bring up the tablet, the first screen that is shown is a controls screen, whereas, in the useable version the screen is blank. This is a way to present control information without it being intrusive, if the user wants to access the information they just need to tap a single button and the fact that it is displayed on a tablet adds to the immersion f this feature.

Figure 21 the control screen on the tablet in the immersive version

The grip button is used for interacting with objects in this version as the motion of pressing the button is similar to a grab action this could make picking up objects feel more immersive as users might feel more like they are using their own hands to grab things in VR. The grip button is on the side of the controller and could easily be pressed by accident, it is extremely recessed to help deter this but this makes it hard to notice and find for new users.



1	Menu button
2	Trackpad
3	System button
4	Status light
5	USB charging adapter
6	Tracking sensor
7	Trigger
8	Grip button

Figure 22 position of the grip button on the vive controller (number 8)

The immersive version uses a locomotion system called room extender. A circle is drawn where the player is stood and within that circle any movements made by the user are completely normal. If the user steps out of that circle their movements and multiplied by a set amount. This method lets users simply walk around the room even though the room they are in is much smaller than the virtual room. This system is extremely immersive as moving around the space is effortless, however, long sessions could cause VR sickness as the movement is slightly unnatural and that would be considered a usability problem.

Selecting participants

At first it was decided that the testing would be conducted by students from the nursing department as they would be the target audience of the application, however, this project is about VR as a whole not just this application. The target audience for VR in general is everyone so it was decided that getting a more diverse range of participants would be [redacted]

The number of participants was set as 10 because the testing could take quite a while to conduct for each participant, so any more than 10 participants could have greatly extended the time needed to perform the tests. The participants were chosen so that there would be a large age range as well as a variance in their familiarity with VR and videogames. The wide range of participants means that the data collected reflects both the opinions of people familiar with VR as well as people who have never used VR before. No personal information was collected from the participants

Protocol

From this point on the usable version will be referred to as version 1 and the immersive version will be referred to as version 2. For each participant the version they tried first was swapped so participant 1 would use version 1 first and participant 2 would use version 2 first. The reason for the swapping was so that if users were just getting used to the way things worked in the first version they played and struggling to adapt to the second version they played, the results wouldn't just show that everyone liked version 1 and didn't like version 2, they would show that 50% of participants prefer version 1 and 50% of participants prefer version 2.

Before the participant enters the room their participant number would be recorded as well as which version they are going to play first. The version that the participant was going to play first would then be set up making sure that the participant is unable to see which version they are playing in order to avoid any bias. The same protocol was used for each version so the participant would follow the instructions in the first version and do the exact same things in the next version.

The instructions for the participants were as follows:

- Find and press the menu button to display the tablet
- Press the play button on the tablet to play the video
- Try exploring the space using this versions locomotion system
- Move over to the door
- Open the door
- Move to the light switch
- Try turning the light up and down
- Pick up the phone
- Move the desk chair around
- Move over to the white board
- Pick up the pens
- Move over to the sink
- Turn the taps on and off
- Look at yourself in the mirror
- Move over to the man lying in the bead
- Place the controller on his arm to feel his pulse

After the protocol had been completed for both versions the participants would answer a questionnaire.

Questionnaire questions

Usability and immersion in a VR experience

A survey to analyze the importance of both usability and immersion in VR and how they affect user enjoyment

Participant number:

Which version did you play first?

On a scale of 1-5 how experienced are you with playing video games?

On a scale of 1-5 how experienced are you with Virtual Reality?

In version 1 was information about the controls easy to find and understand?

In version 1 was information about the controls distracting; did it take you out of the experience?

Any additional thoughts about how Version 1 displayed control information?

In version 2 was information about the controls easy to find and understand?

In version 2 was information about the controls distracting, did it take you out of the experience?

Any additional thoughts about how Version 2 displayed control information?

Which way would you prefer to get control information in the future?

In version 1 on a scale of 1-5 how easy was it to pick up/ interact with objects?

In version 1 on a scale of 1-5, when picking up or interacting with objects, how often did you forget that you were using a controller?

Any additional thoughts about Interacting with objects in Version 1?

In version 2 on a scale of 1-5 how easy was it to pick up/ interact with objects?

In version 2 on a scale of 1-5, when picking up or interacting with objects, how often did you forget that you were using a controller?

Any additional thoughts about Interacting with objects in Version 2?

Which button would you prefer to be used for interacting with objects in the future?

In version 1 was the way in which the controller was represented helpful in using the app?

In version 1 on a scale of 1-5 do you think that way in which the controller was represented made you feel more present in the virtual world?

Any additional thoughts about how the controller was represented in Version 1?

In version 2 was the way in which the controller was represented helpful in using the app?

In version 2 on a scale of 1-5 do you think that way in which the controller was represented made you feel more present in the virtual world?

Any additional thoughts about how the controller was represented in Version 2?

How would you prefer the controller be represented in the future?

In version 1 on a scale of 1-5 how usable was the movement system?

In version 1 on a scale of 1-5 how immersive was the movement system?

Any additional thoughts about the movement system in Version 1?

In version 2 on a scale of 1-5 how usable was the movement system?

In version 2 on a scale of 1-5 how immersive was the movement system?

Any additional thoughts about the movement system in Version 2?

Which movement system would you prefer to be used in the future?

In general which version is more usable?

In general which version is more immersive?

Which version would you prefer to use?

Hypothesis

- Participant that are more used to using video game controllers will prefer the hand model because of the added immersion it provides. Participants who aren't as experienced with videogames will prefer the controller model because they can look at it in order to find where the buttons are.
- Most participants will prefer to be given control information on the tablet as it is unintrusive but still provides all necessary information.
- Most participants will prefer the trigger for interacting with objects because it is much easier to use than the grip button and provides more tactile feedback.
- Participants that have used VR before will prefer the walking locomotion system because it's effortless but participant who aren't familiar with VR will prefer teleporting because walking around causes them VR sickness.

Results and evaluation

All participants completed the protocol for both versions of the application and completed the questionnaire. The participants had a good range of video game experience with one person saying they never play video games, three people saying they hardly ever play, three people saying they occasionally play and three saying they play all the time. In regards to Virtual reality experience three people had never used VR before, three had used it a few times, two used it occasionally and two had used it quite a lot. The lack of VR experience could be put down to VR still being in its infancy as a consumer product and I might have been hard to find a few more participants who are more experienced.

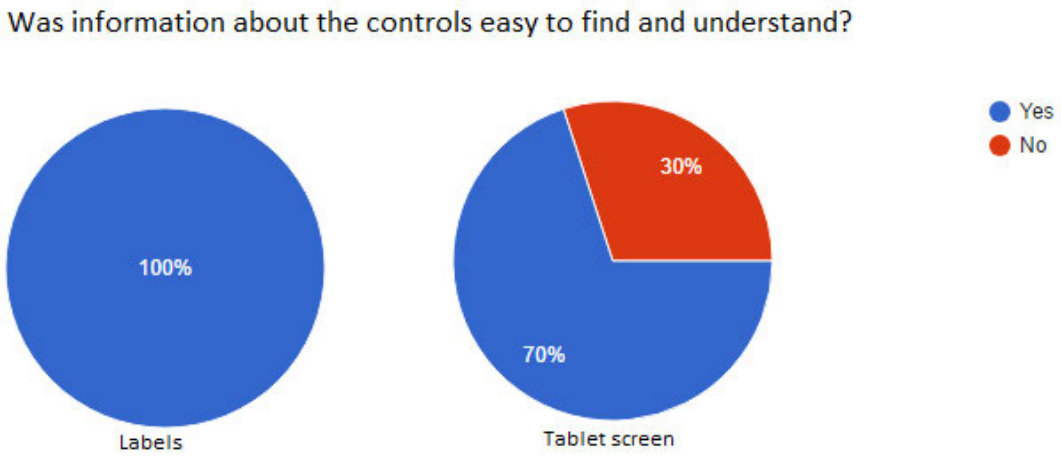


Figure 23 Control information usability results

Figure 23 show that all of the participants felt that using the labels on the controller model was easy to understand and use, whereas, three of the participants felt that the tablet screen approach wasn't. Feedback from the participants suggests that having to press a button to receive control information could cause problems if someone isn't there to explain that. All participants that expressed these concerns said that the information was clear and easy to understand once they had been told which button to press.

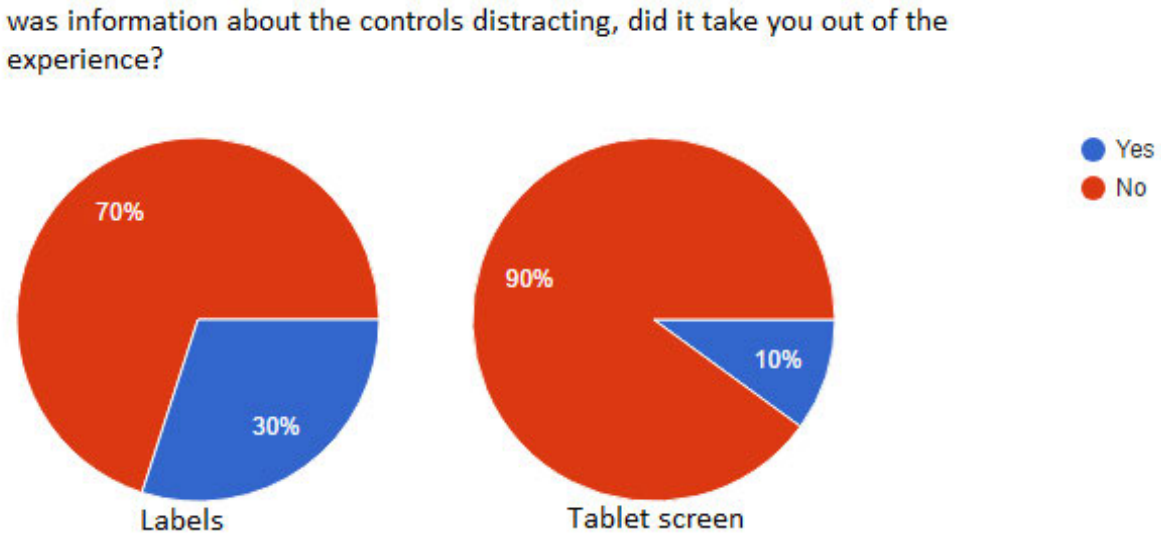


Figure 24 Control information immersion results

Figure 24 shows that participants clearly felt that receiving control information from the tablet was less distracting and therefore more immersive than the labels. One participant suggested that making the labels fade away when the user wasn't looking directly at the controller might help to improve immersion when using that method and this would defiantly be something to look into if the project were revisited.

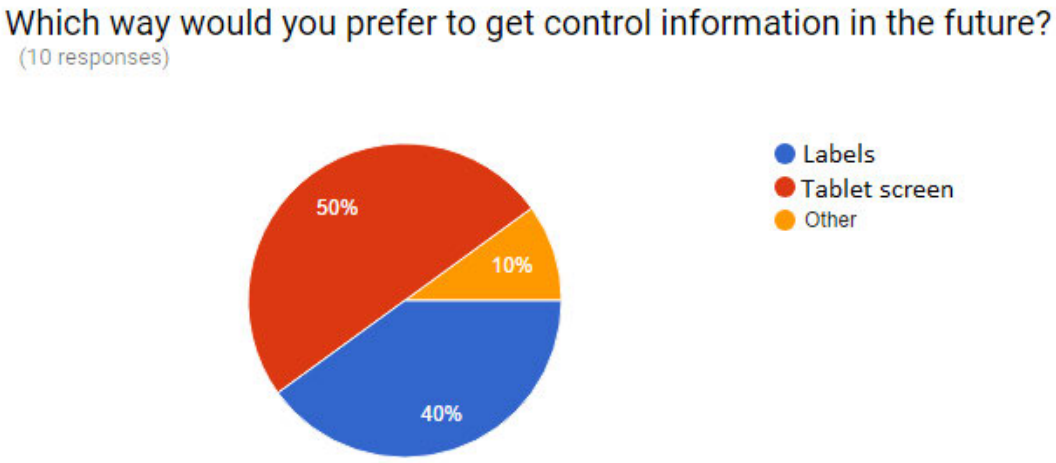


Figure 25 Control information preference results

The result of how participants would prefer to get control information in the future is inconclusive. Five participants preferred the information on the tablet, 4 participants preferred the labels and the last participant would prefer the labels if they disappeared when not looking at the controller. Interestingly this result does not align in any way to participants experience level with videogames or VR. Both control information methods got votes from participants with lots of experience with videogames and participants with none what so ever.

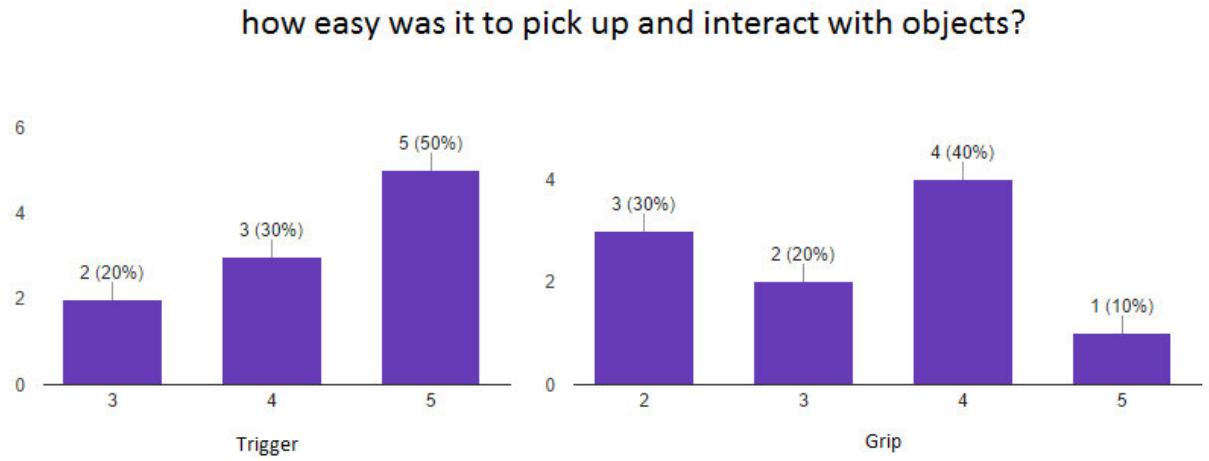


Figure 26 Interaction button usability results

When assessing how easy it was to interact with objects no participants scored the trigger less than three, and half of the participants scored it a 5. In comparison half of the participants scored the grip button three or less clearly showing that the trigger is easier to use. Even so, 4 participants scored the grip a 4 and one scored it a 5, this could be due to many of the participants' lack of exposure to VR and video games, so they had no experience with traditional controller input, making it easier for them to adapt to this new control scheme.

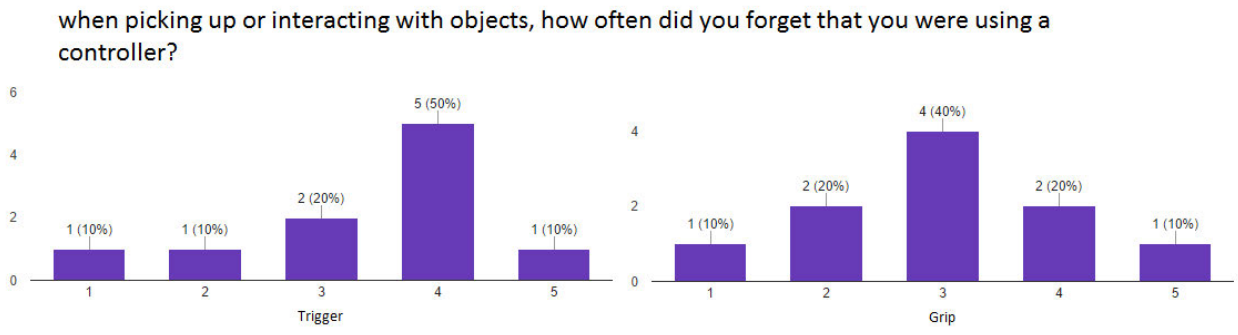


Figure 27 Interaction button immersion results

The results for how immersive the trigger and grip buttons are to use is quite surprising. 60% of participants scored the trigger a 4 or higher whereas 70% of participants scored the grip a 3 or less. Feedback from the participants suggests that the grip button could be hard to find and consistently press reliably making it more obvious that a controller is being used not just hands. The same thought process could be applied to the trigger button, even though a realistic grab motion isn't being performed when using the trigger button its prominent position on the controller makes it effortless to press and therefore allows the user to forget about the controller and focus on the experience.

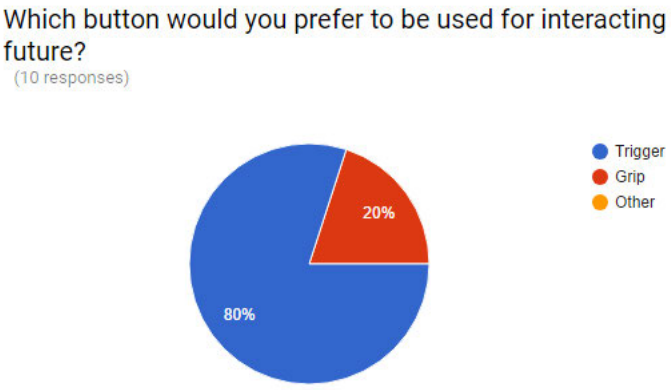


Figure 28 Interaction button preference results

The results for which interaction button was preferred fall in line with the results from the other two questions about the interaction buttons and eight out of 10 participants would prefer to use the trigger button.

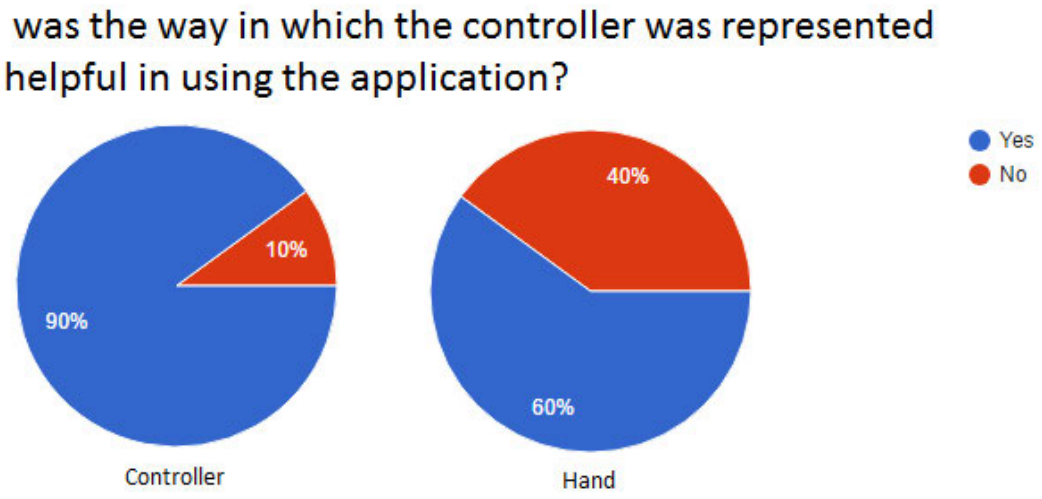


Figure 29 Controller representation usability results

In terms of how the controller is represented, participants obviously felt that the controller model was very helpful, presumably because it meant that they could look down and locate buttons that they wanted to press easily. Only 60% of participants said that the hand model was helpful while playing the game, once again this could be because of a lack of experience using controllers so being able to see the controller is necessary.

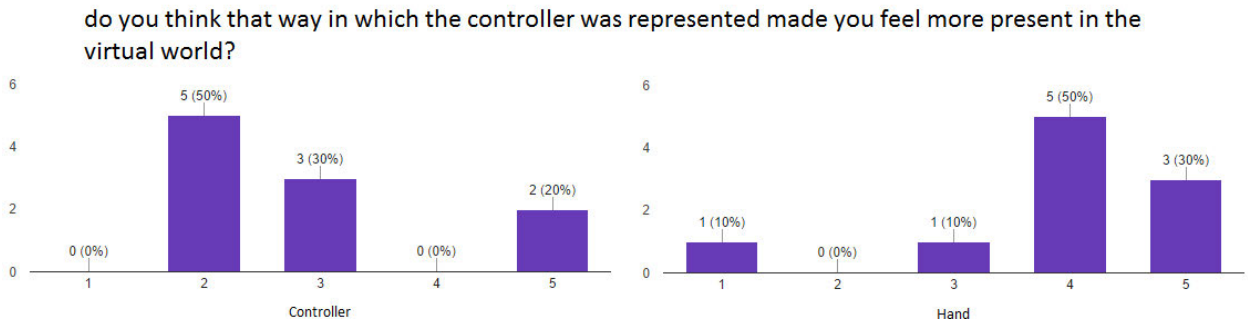


Figure 30 Controller representation immersion results

Most participants agreed that the hand model made them feel more present in the simulation lab and the controller model broke immersion or didn't give the same sense of presence. A few points were raised that contradict these results however; one participant claimed that because they knew they were holding a controller in their hand seeing in VR made the experience feel more real. A participant said that the hands were too cartoony to be believable but if the hand model was too realistic it might cross the line into uncanny, which is equally as immersion breaking. The fact that the hands are floating in mid air was also given as a reason why the hand model isn't immersive, this is a technological problem with current VR equipment, as to realistically simulate arms, each joint in the arm and shoulder would have to be tracked which isn't currently possible with consumer VR.

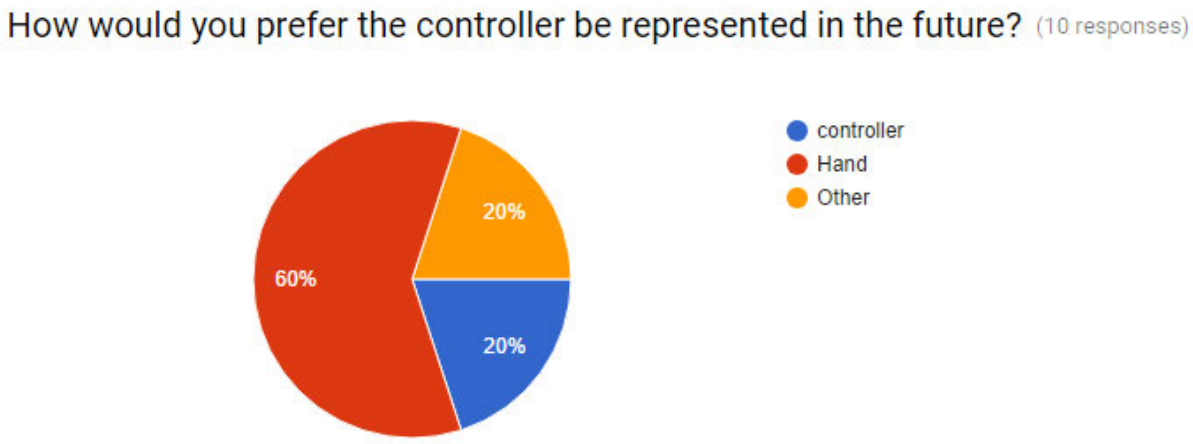


Figure 31 Controller representation preference results

Even though concerns were raised about not being able to see the buttons on the controller, 60% of participants said that they would prefer to use the hand model in the future. 2 participants voted for other because they wanted a menu in which the 3d model could be swapped at will, allowing new users to see the controller and more experienced users to swap to the hands.

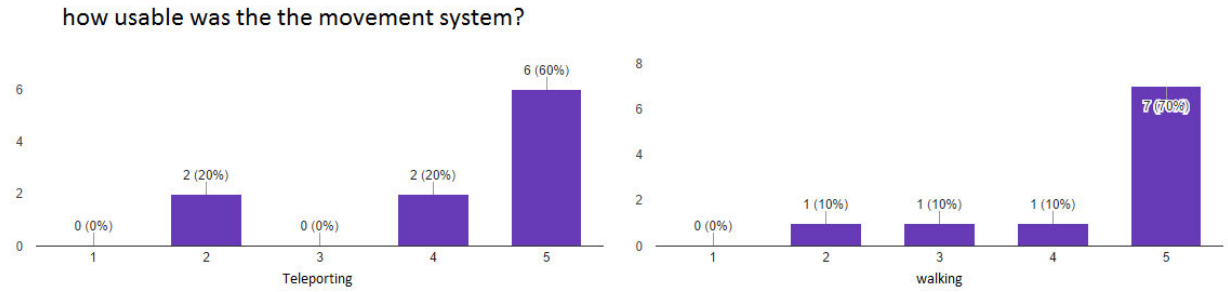


Figure 32 Movement system usability results

This result was extremely close with 60% of participants scoring teleporting a 5 and 70% scoring walking a 5. This could be an issue with this test as the amount of time each participant spent in VR was only about 10 minutes, perhaps longer amounts of time using the movement systems would have revealed usability problems that the participants didn't notice such as sickness or dizziness.

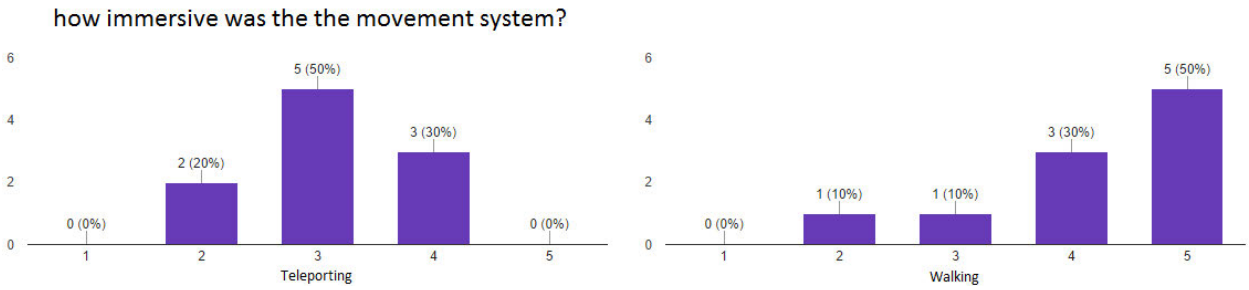


Figure 33 Movement system immersion results

Figure 33 clearly shows that participants thought that the walking locomotion technique was more immersive than teleporting with 50% giving walking a 5 while 50% gave teleporting a 3. Some participants mentioned that because of the small size of the play area used for the testing, they were worried they would bump into things in the real world while walking around which stopped them from getting immersed in the experience.

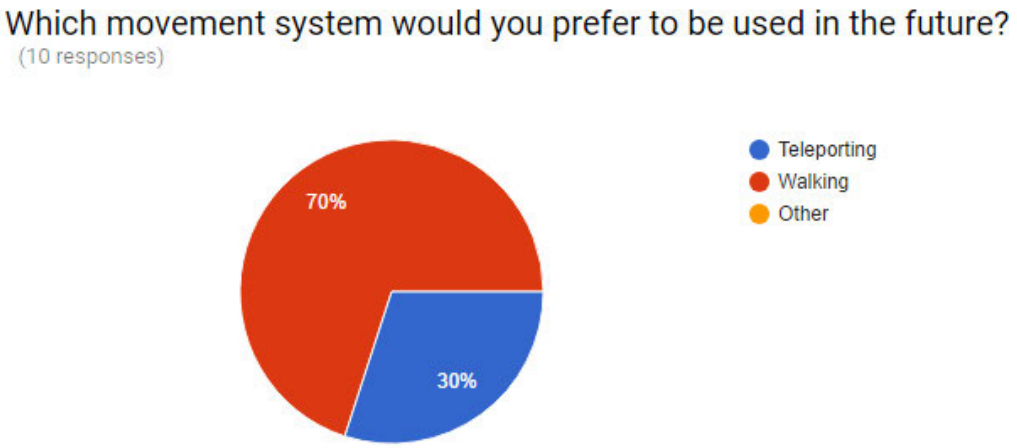


Figure 34 Movement system preference results

The result in figure 34 is in line with the rest of the movement system results. Walking was rated the most usable and the most immersive so it makes sense that the participants would prefer to use that system.

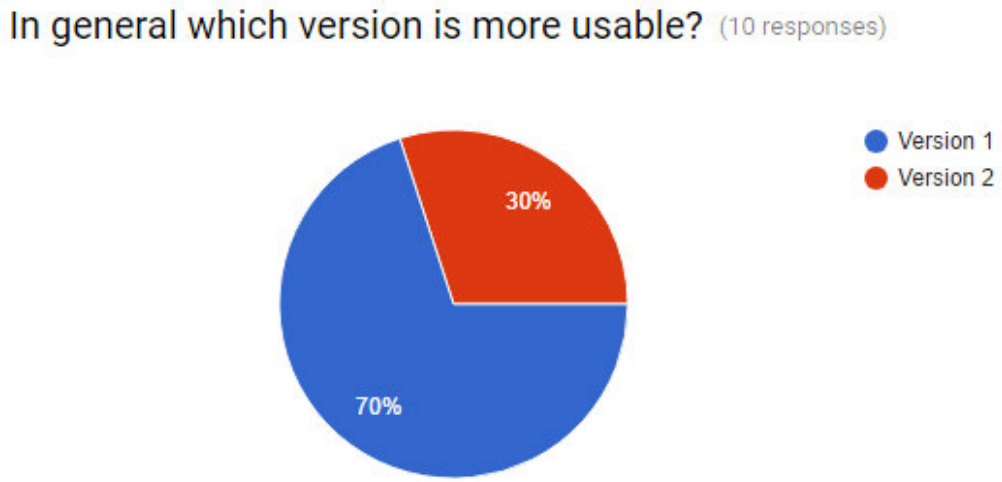


Figure 35 Which version was more usable results

Version 1 was created to be the usable version and the fact that most participants agree that it is the most usable shows that it achieved the aims that were set out for the version.

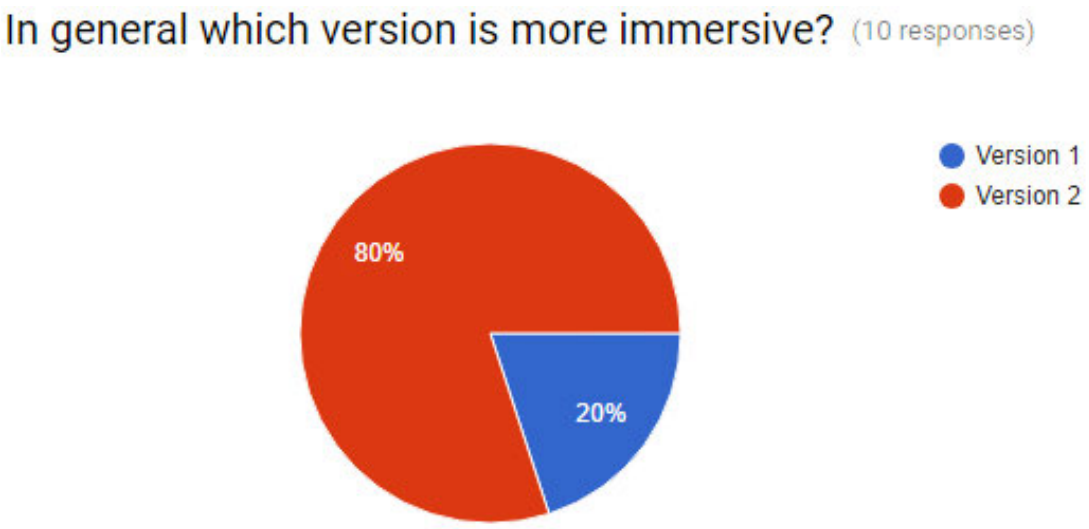


Figure 36 Which version was more immersive results

As with the more usable version, version 2 was created to be as immersive as possible and this result shows that it achieved the aims set out for it.

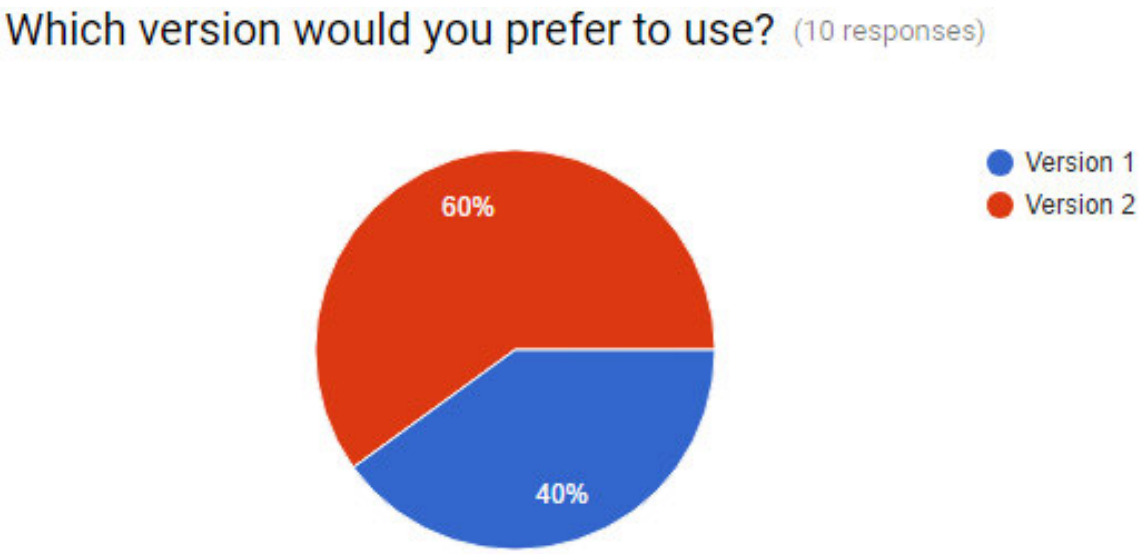


Figure 37 Overall best version results

This result is surprising because while version 2 had a lot of easy to use and immersive features like the walking system and the hands, many of the participants also had issues with these features. The features in version 1 weren't as controversial as features from version 2 but they also perhaps weren't as immersive and exciting which might explain my more participants chose version 2 in this question

PROJECT EVALUATION

The first aim of this project was to create two versions of the same VR experience. Both versions of the application were completed on time and include a large range of interactions as well as their own 4 main features which would be analyzed.

The second aim was to test the two versions with participants who would give their opinions on what was immersive and usable in each version. 10 participants tested the applications and gave feedback in and extensive questionnaire which provided valuable insight into immersion and usability in VR.

Improvements

More interactions could have been created for the application to incentivize participants to test the application for a longer period of time making it more likely that issues would be caught that might only become apparent after extended use , such as dizziness or sickness.

The questions on the questionnaire could have been simpler as a lot of participants were having trouble figuring out what the questions were asking them.

Participants should play a version first and then answer questions about that version before playing the second version. Many participants were getting confused about what happened in each version because they played both before answering any questions.

CONCLUSION

To conclude this project I would like to come back to a question raised in the abstract of this report, are usability and immersion competing concepts when it comes to virtual reality? This research has shown that sometimes they can be, but other times they are one in the same. The interaction button was the main example of this where the trigger button is so easy and responsive to use participants didn't even have to think about the controller allowing them to get more immersed in the experience. The grip button was on paper the more immersive choice as the action of pressing it is similar to a grabbing action but its poor positioning and unresponsive feel leads to frustration which actually lowers immersion in the game.

On the other end of the spectrum is how the controller is represented, the usable version is not at all immersive and the immersive version is not at all usable. It seems the best way to tackle this issue is to just include both and let users choose which one they feel better suits their needs. In the end there needs to be a balance between immersion and usability in order to ensure that the maximum amount of people are catered for when creating virtual reality applications.

REFERENCES

Virtual Reality Society. (2015) *History of virtual reality*. Available at: http://www.vrs.org.uk/virtual-reality/history.html (Accessed: 19 October 2016).

Alex (2015) *Oculus rift history - how it all started*. Available at: http://riftinfo.com/oculus-rift-history-how-it-all-started (Accessed: 19 October 2016).

Data protection (2016) Available at: https://www.gov.uk/data-protection/the-data-protection-act (Accessed: 19 October 2016).

Number of smartphone users worldwide 2014-2020. (2016) Available at: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (Accessed: 23 November 2016).

Projected virtual reality headsets unit sales worldwide in 2016 (in million), by device. (2016)

Available at: https://www.statista.com/statistics/458037/virtual-reality-headsets-unit-sales-worldwide/ (Accessed: 23 November 2016).

forensic VR (2015) Available at: http://forensic-vr.com/VR (Accessed: 24 November 2016).

History of virtual reality. (2015) Available at: http://www.vrs.org.uk/virtual-reality/history.html (Accessed: 23 November 2016).

Simanek, D.E. (no date) *Stereo view card pictures for cross-eyed viewing*. Available at: https://www.lhup.edu/~dsimanek/3d/stereo/3dgallery3.htm (Accessed: 23 November 2016).

Mazuryk, T. and Gervautz, M. (1996) *Virtual Reality: History, Applications, Technology and Future*. Available at: https://www.cg.tuwien.ac.at/research/publications/1996/mazuryk-1996-VRH/TR-186-2-96-06Paper.pdf (Accessed: 24 November 2016).

Ripton, J. and Prasuethsut, L. (2015) *The VR race: What you need to know about Oculus rift, HTC Vive and more*. Available at: http://www.techradar.com/news/world-of-tech/future-tech/the-vr-race-who-s-closest-to-making-vr-a-reality-1266538 (Accessed: 24 November 2016).

Mendelsohn, T. (2016) *HTC Vive goes wireless with pricey upgrade kit*. Available at: http://arstechnica.co.uk/gadgets/2016/11/htc-vive-vr-wireless-upgrade-kit/ (Accessed: 24 November 2016).

Porter, J. (2016) *Oculus touch controller review*. Available at: http://www.techradar.com/reviews/oculus-touch-controller (Accessed: 25 November 2016).

Steed, A., Frlston, S., Lopez, M.M., Drummond, J., Pan, Y. and Swapp, D. (2016) 'An "In the Wild" Experiment on Presence and Embodiment using Consumer Virtual Reality Equipment', *IEEE Transactions on Visualization and Computer Graphics*, 22(4), pp. 1406–1414. doi: 10.1109/tvcg.2016.2518135.

Moore, C. (2016) 'The virtual yellow house: Experimental tangling with virtual reality', *IEEE Consumer Electronics Magazine*, 5(4), pp. 103–104. doi: 10.1109/mce.2016.2590219.

Brooks, F.P. (1999) 'What's real about virtual reality?', *IEEE Computer Graphics and Applications*, 19(6), pp. 16–27. doi: 10.1109/38.799723.

Heineman, D.S. (2016) 'Porting game studies research to virtual reality', *New Media & Society*, 18(11), pp. 2793–2799. doi: 10.1177/1461444816661711.

Bishop, G. and Fuchs, H. (1992) 'Research directions in virtual environments', *ACM SIGGRAPH Computer Graphics*, 26(3), pp. 153–177. doi: 10.1145/142413.142416.

Durlach, N.I., Mavor, A.S., Council, N.R., Staff, N.R.C. and Computer Science and Telecommunications Board (1994) *Virtual reality: Scientific and technological challenges*. Washington, D.C.: National Academies Press.

Sherman, W.R. (2002) *Understanding virtual reality: Interface, application, and design*. Edited by Sherman Josepha and Alan Craig. San Francisco, CA: Morgan Kaufmann Publishers.

M. Gigante (1993) Virtual Reality: Definitions, History and Applications. Edited by M. Gigante,H.Jones, R.A Earnshaw. Academic-Press, ISBN 0-12-22-77-48-1

Burdea, G.C. and Coiffet, P. (2003) *Virtual reality technology*. 2nd edn. New York, NY, United States: J. Wiley-Interscience.

Merchant, Z., Goetz, E.T., Cifuentes, L., Keeney-Kennicutt, W. and Davis, T.J. (2014) 'Effectiveness of virtual reality-based instruction on students' learning outcomes in K-12 and higher education: A meta-analysis', *Computers & Education*, 70, pp. 29–40. doi: 10.1016/j.compedu.2013.07.033.

Rajesh Desai, P., Nikhil Desai, P., Deepak Ajmera, K. and Mehta, K. (2014) 'A review paper on Oculus rift-a virtual reality Headset', *International Journal of Engineering Trends and Technology*, 13(4), pp. 175–179. doi: 10.14445/22315381/ijett-v13p237.

Dempsey, P. (2016) 'The Teardown: HTC Vive virtual reality headset', Engineering & Technology, 11(7), pp. 80–81. doi: 10.1049/et.2016.0731.

Zeltzer, D. (1992) ‘Autonomy, interaction, and presence’, Presence: Teleoperators and Virtual Environments, 1(1), pp. 127–132. doi: 10.1162/pres.1992.1.1.127.

APPENDICES

Usability and immersion: a virtual reality study terms of reference

Student: [redacted] **Supervisor:** Huw Lloyd

Course-Specific Learning Outcomes: on successful completion of this unit, the student will be able to:

1. Study the history of computer games, game genres, game structures and game design principles and to use the skills acquired to specify and evaluate new game applications.
2. Become knowledgeable in the use and development of computer graphics software/game middleware tools and to be able to apply this knowledge to the implementation of real-time interactive systems.
3. Learn structured approaches to computer programming.
4. Learn how the theories and techniques of behavioural systems can be used to enhance the playability and sophistication of computer games.
5. Study a range of mathematical tools and techniques and to be able to use these, in particular, to solve problems in the area of game modelling and animation.
6. Gain an appreciation of the multidisciplinary environment in which commercial games are designed and produced and to acquire skills in project management and team working.

Project Background:

As early as the 1830's stereoscopic viewers have been used for "virtual tourism" and the same technology is still used today for low budget VR such as *Google Cardboard* but, since the popularisation of videogames in the 1980's, the idea of a totally immersive virtual world became realistic instead of science fiction.

Films such as *Tron(1982)* and *The Lawnmower Man(1992)* show how ingrained this idea was in popular culture, and so, videogame companies started to experiment with Commercial Virtual Reality. Both Nintendo and Sega announced that they were developing Virtual Reality headsets in the early 1990's, however, Sega's hardware ran into technical issues and was never released. Nintendo's *Virtual Boy* was highly anticipated by consumers but the primitive graphics and uncomfortable viewing position meant that it was discontinued only a year after release.

It was obvious that the technology needed further advancements before Virtual Reality could be considered immersive in any way and games companies shifted their focus back to traditional consoles and games. The release of *The Matrix* in 1999 showed that there was a continued interest in Virtual Reality but it would take 15 more years for consumer Virtual Reality to become viable.

In June 2012 Palmer Luckey used the group funding site *Kickstarter* with the hope of funding his head mounted display named the *Oculus Rift*. *John Carmack*, founder of *Id Software,* then showed off a version of *Doom 3 BFG* running on a prototype of the Rift at E3, it was this that sky-rocketed the Oculus Rift Kickstarter to over 2.4 million dollars and also started the resurgence of consumer VR.

Today, many head mounted displays (or HMDs) exist, such as: *Google Cardboard* - which allows users to view 360 degree photos and play simple VR games and the recently released Playstation VR which seeks to bring VR to consoles. This project will focus on VR for PC, specifically the *HTC Vive*.

The HTC Vive uses two infra red sensors the track the location of the HMD and the motion controllers meaning that players can walk around the game world and interact with objects by simply picking them up. Games like *Rec Room* and *Job Simulator* use the Vive to make their experiences more immersive and unique. This new input method, while immersive, brings up questions in regards to the usability of the software as people who are used to more conventional input methods might find it hard to adapt and novice users could get confused.

Many developers are now facing the issue of trying to develop games that can immerse users in new and exciting worlds without making it frustrating and difficult to navigate and interact with objects. This project will explore the problem stated above and attempt to come up with a solution that allows games to find the right balance between immersion and usability.

Aim: the aim of this work is to analyse how immersion and usability affect player enjoyment in VR by creating two versions of an application for the HTC Vive, one which includes features to make it as usable as possible and one that includes features that make it as immersive as possible.

Objectives:

- Research game engines and choose one that best meets my needs for this project.
- Create two scenes that can be used as the levels for the game.
- Implement mechanics that let players interact with objects and the game world in fun and interesting ways.
- Research ways to make these mechanics both usable and immersive.
- Evaluate the game play on each version by conducting a survey and analysing usability and immersion.

Problems:

Hardware failure was initially a big problem for this project as only one HTC Vive was available to use so if that one failed the project would not be able to continue. The nursing department has a HTC Vive and allowed its use for the project which solved this problem.

Data collection could be another problem this project could face. Evaluation of the final product is required which means participants are needed and data will have to be collected from them. This project will adhere strictly to the data protection principles to ensure participants data is not abused or held unnecessarily.

Required Resources:

- HTC Vive
- Unreal Engine
- Unity Engine

Timetable:

Deliverable	start date	end date
Terms of reference	23-09-16	21-10-16
Game engine research	23-09-16	21-10-16
Ethics Form	21-10-16	28-10-16
Create prototype	21-10-16	11-11-16
Literature Survey	28-10-16	25-11-16
expand prototype	25-11-16	26-12-16
Product design	25-11-16	16-12-16
Evaluation design	16-12-16	03-02-17
create questionnaire	13-01-17	20-01-17
Testing and evaluation	20-01-17	30-01-17
Report outline	03-02-17	24-02-17
Draft Slides	24-02-17	10-03-17
Practice Presentation	N/A	14-03-17
Submit Report and Product	N/A	24-04-17
Final Presentation	N/A	25-04-17

References:

1. Virtual Reality Society. (2015) *History of virtual reality*. Available at: http://www.vrs.org.uk/virtual-reality/history.html (Accessed: 19 October 2016).

2. Alex (2015) *Oculus rift history - how it all started*. Available at: http://riftinfo.com/oculus-rift-history-how-it-all-started (Accessed: 19 October 2016).

3. *Data protection* (2016) Available at: https://www.gov.uk/data-protection/the-data-protection-act (Accessed: 19 October 2016).