



# Multiplayer VR Archery Game

Ryan Golden

N00150620

Supervisor: Joachim Pietsch

Second Reader: Cyril Connolly

Year 4 2022-23

DL836 BSc (Hons) in Creative Computing

## Abstract

The overall aim of this project was to learn how to properly implement multiplayer functionality into a virtual reality game. To this end a multiplayer archery game was developed in virtual reality allowing players to hone their skills and compete against other players. As a relatively new medium, VR has seen rapid growth in the past decade due to advancements in technologies and ease of access.

Creating video games is very accessible in game engines like Unity and Unreal. The development of VR is an engaging challenge due to it being the newest gaming medium. The game will be developed using the Unity Game Engine and will be primarily coded using C#, the default scripting language for Unity. The multiplayer component will be developed using the Normcore Framework, a plugin built with VR multiplayer games in mind.

The steps involved during development were the creation of a research document, requirements gathering and design discussion which includes both UI and game design. Following this, the game was tested thoroughly, and the results of the tests were used to polish and improve the game before the final finished product was submitted.

The game could be developed further by implementing more game modes for both single and multiplayer, adding additional player customization options and implementing more maps that could be played on.

## Acknowledgements

I would first like to express my thanks to my project supervisor, Joachim Pietsch, and my secondary reader Cyril Connolly for their constant advice and guidance for the duration of the project. They both ensured that the applications development progressed smoothly, and I am incredibly grateful for their help. Joachim's keen interest in game development pushed my application further and encouraged me to create something I could be proud of, and Cyrils outside perspective allowed me to develop the game with those having no VR experience in mind, ensuring the controls and gameplay was easy to learn and understand.

I would also like to thank the individuals who participated in my user survey and interviews, as well as the people involved with testing the application.

**The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.**

If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/collaboration is acceptable, you should consult your lecturer or the Course Director.

**WARNING:** Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not leave copies of your own files on a hard disk where they can be accessed by others. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended.

Plagiarism is considered to be an act of fraudulence and an offence against Institute discipline.

Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.

**The following is an extract from the B.Sc. in Creative Computing (Hons) course handbook. Please read carefully and sign the declaration below**

*Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.*

**DECLARATION:**

I am aware of the Institute's policy on plagiarism and certify that this thesis is my own work.

Student :

Signed

Ryan Golden

Failure to complete and submit this form may lead to an investigation into your work.

## Table of Contents

1	Introduction .....	1
2	Research.....	3
2.1	Introduction .....	3
2.2	Digital Reality .....	3
2.2.1	Virtual Reality.....	3
2.3	Accessibility.....	4
2.3.1	Visually Impaired.....	4
2.3.2	Hearing Impaired .....	5
2.3.3	Physically Impaired .....	5
2.4	Conclusion.....	6
3	Requirements.....	7
3.1	Introduction .....	7
3.2	Requirements gathering .....	7
3.2.1	Similar applications .....	7
3.2.2	Interviews.....	12
3.2.3	Survey.....	13
3.3	Requirements modelling.....	15
3.3.1	Personas.....	15
3.3.2	Functional requirements.....	17
3.3.3	Non-functional requirements .....	17
3.3.4	Use Case Diagrams.....	18
3.4	Feasibility .....	18
3.5	Conclusion.....	19
4	Design.....	20
4.1	Introduction .....	20
4.2	Program Design.....	20
4.2.1	Technologies .....	21
4.2.2	Structure of Unity.....	21
4.2.3	Application architecture (1 page) .....	25
4.3	User interface design .....	26
4.3.1	Wireframe .....	27
4.3.2	User Flow Diagram .....	29
4.3.3	Style guide.....	30

4.3.4	Storyboard .....	32
4.3.5	Level Design .....	32
4.3.6	Environment.....	33
4.4	Conclusion.....	34
5	Implementation .....	35
5.1	Introduction .....	35
5.2	Technologies .....	35
5.3	Development environment.....	35
5.4	Sprint 1.....	36
5.4.1	Goal .....	36
5.5	Sprint 2.....	37
5.5.1	Goal .....	37
5.5.2	Bow and Arrow Prototype .....	37
5.6	Sprint 3.....	38
5.6.1	Goal .....	38
5.6.2	Multiplayer Framework Implementation .....	38
5.6.3	Multiplayer Object Synchronisation .....	38
5.7	Sprint 4.....	39
5.7.1	Goal .....	39
5.7.2	Multiplayer Scoreboard .....	39
5.7.3	Multiplayer Score Syncing.....	40
5.7.4	Multiplayer Point Acclamation .....	40
5.8	Sprint 5.....	41
5.8.1	Goal .....	41
5.8.2	Single Player Bow and Arrow .....	41
5.8.3	Single Player Score System .....	41
5.9	Sprint 6.....	42
5.9.1	Goal .....	42
5.9.2	Creation of the Main Menu .....	42
5.10	Sprint 7.....	43
5.10.1	Goal .....	43
5.10.2	Initial Level/Environment Design .....	43
5.10.3	Enhanced UI .....	44
5.11	Sprint 8.....	45
5.11.1	Goal .....	45
5.11.2	Finalising the Multiplayer Environment.....	45

5.11.3	Creation and Completion of the Single Player Environment .....	46
5.12	Conclusion.....	47
6	Testing.....	48
6.1	Introduction .....	48
6.2	Functional Testing.....	48
6.2.1	World Navigation .....	48
6.2.2	Menu Navigation.....	49
6.2.3	Gameplay .....	50
6.2.4	Multiplayer.....	51
6.2.5	Discussion of Functional Testing Results .....	51
6.3	User Testing .....	51
6.4	Conclusion.....	52
7	Project Management .....	53
7.1	Introduction .....	53
7.2	Project Phases.....	53
7.2.1	Proposal .....	53
7.2.2	Requirements.....	53
7.2.3	Design.....	53
7.2.4	Implementation .....	54
7.2.5	Testing.....	54
7.3	Project Management Tools.....	54
7.3.1	Trello .....	54
7.3.2	GitHub .....	55
7.4	Reflection .....	55
7.4.1	Your views on the project .....	55
7.4.2	Completing a large software development project.....	55
7.4.3	Working with a supervisor .....	56
7.4.4	Technical skills.....	56
7.4.5	Further competencies and skills .....	56
7.5	Conclusion.....	57
8	Conclusion.....	58
9	References .....	59
	Appendix .....	61
	Interview Transcript.....	61

## 1 Introduction

This project is a virtual reality archery game where the player is attempting to score as many points as they can either in a single player environment, or against another player in a multiuser room. Players will need to use the bow and arrows at their disposal and accurately fire at targets downrange much like you would in real life. Each target is worth varying amounts of points depending on the level of skill required to hit it accurately, and players can choose to start a timer and attempt to score as many points as they can in the allocated time.

The game was developed with satisfaction and enjoyment in mind, and the game mechanic of firing the bow and arrow was a key focus. Players will accurately see their skill improve as they continue to play the game, achieving a higher score with more practice. This is due to the increasing speed with which the player handles the bow, and the accuracy that he or she achieves when firing.

The project was developed using the Unity Game Engine, and all scripting and code was written using Visual Studio Code in the C# language. The game utilizes Unity's XR Interaction Toolkit for its VR functionality, and the Normcore Framework plugin for multiplayer purposes, discussed more thoroughly in the design and implementation sections of the document.

Throughout development, the project source code was uploaded to GitHub using the Github Desktop Application, and for a period Trello was also used to aid in project management allowing more time to be spent on sections that needed more work.

This document is split into four main sections:

### **Requirements**

The Requirements section discusses similar applications and explains the positive and negative attributes of each. User surveys and interviews were also discussed and recorded in this section, and the results of which were used to develop personas and decide on the functional requirements for the application itself.

### **Design**

The Design section goes over the core design and aesthetics of the application. The projects UI and environmental design are finalized, and various figures were created to help visualize how users would use the application. Additionally, the technologies being used to develop the game are chosen, and the inclusion of wireframes and paper prototypes aid in its finalization.

### **Implementation**

The implementation section describes how each sprint during the development cycle was tackled, describing what happened and what was added to the game. Additionally, the development environments are described in more detail, and the technologies utilized are also described more coherently.

### **Testing**

Finally, the testing section explains how the application was tested for discrepancies and errors by utilizing both functional and user testing. These tests were conducted to further improve and polish the game before the projects conclusion and were also used to gather feedback from players and see what can be changed if necessary.



Following this is a brief discussion on how the project was managed, where I will discuss how certain areas could be improved and what aspects were either added or removed in order to produce the finished product.

## 2 Research

A research paper was produced with an emphasis on accessibility options in digital reality, before the project had been fully decided on. As the project evolved, the focus diverged from digital reality and accessibility, however relevant information from the document has been included, with several edits, to focus more wholly on virtual reality alone.

### 2.1 Introduction

“Digital reality refers to the wide spectrum of technologies and affordances that include Augmented Reality, Virtual Reality and Mixed Reality that simulate reality in various ways.” (Katharina Finger & Elvedin Mesic., 2019). The rapid progression of technology around the world has led to substantial breakthroughs in many fields, one such being the creation and distribution of various Virtual, Augmented and Mixed reality platforms. While the general populace can enjoy these new, futuristic forms of entertainment, effort needs to be made to ensure that everyone can participate. As they are relatively new developments, many Digital Reality mediums lack the proper tools to habilitate those with impaired physical skills, or who suffer from poor eyesight or hearing.

This literature review will first address what exactly the primary type of Digital Reality platform is available. This section will discuss in depth what Virtual Reality refers to.

The next segment will discuss accessibility and the various options made available to people with various disabilities that utilize this platform. Following this is a more in-depth discussion on individual disabilities, with a primary focus on those with hearing, sight, and physical ailments.

### 2.2 Digital Reality

As stated previously, Digital Reality refers to the various platforms that simulate our reality or entirely new realities in various ways. To fully understand this however, we need to understand what we mean by the term ‘Reality’. Reality is defined as the state of things as they actually exist, as opposed to an idealistic or notional idea of them. It requires an understanding of what is desirable, and what is feasible, and the notion that digital reality already has distinct definitions and different concepts for Virtual, Mixed and Physical Reality shows that they have yet to truly integrated into our society (Baranyi, Csapó, Budai, & Wersényi, 2021). However, while they have yet to seamlessly link with our own reality, they are still very much a key part of it moving forward and the various terms serve as a good way to differentiate them without confusing the users.

According to Katharina Finger & Elvedin Mesic (2019), Digital Reality immerses you in the content, providing a unique perspective on media that could have been previously unattainable through standard means, in addition to providing users with more control. Three primary platforms are easily identifiable as separate forms of Digital Reality, as they provide users with these defined parameters. Virtual Reality, which aims to fully immerse the user in a digital rendition of a world, either based on our own or wholly original. Augmented Reality, which aims to use the real world as a canvas and allow people to interact with things that aren’t there in reality through the use of headsets, mobile devices and cameras. And Mixed Reality, which aims to combine the physical and digital world, and is not tied to any particular technology (Young, Sharlin, & Igarashi, 2011).

#### 2.2.1 Virtual Reality

Virtual Reality is an advanced, human-computer interface that gives the user full control and movement in a virtual world, being able to interact and reshape it as they see fit, according to Zheng, Chan, & Gibson (1998). While historically seen as something incredibly futuristic and far off, the technological boom has seen VR expand exponentially in the past decade. While originally only available in places like arcades and specific VR experiences, VR has become a more common household name with the introduction of cheaper headsets such as the Oculus Quest, or Playstation VR.

VR allows users to experience locations, entertainment or social events in ways that could have never been imagined before. Being able to completely immerse yourself in a virtual world with full motor control is an unforgettable phenomenon. Most VR headsets come with a plethora of extra peripherals that can greatly enhance the user's experience such as additional trackers for full body motion tracking, more advanced controllers for individual finger tracking, and even full body haptic feedback suits, to allow users to truly feel the worlds they immerse themselves in.

VR applications are used in many domains, such as medicine, education, or entertainment (Burdea & Coiffet, 2003). VR allows people to explore real life locations created by owners of museums, art exhibits, and even allows educational institutions to perform classes completely remotely while still retaining some semblance of togetherness. Medical institutions and flight training courses utilize VR as training, as it allows those learning to operate in a safe, controlled environment with no real-world ramifications.

VR still comes with a lot of setbacks, however. If one wishes to own a VR headset, despite the prices becoming more affordable than they once were, it is still a very expensive hobby. Enough space is required to operate these pieces of hardware as well, and although certain devices (Such as the Oculus Quest) require no wires or trackers, many of the larger brands do. Such setbacks are things that Augmented Reality (AR) and Mixed Reality (MR) often don't have to deal with.

## 2.3 Accessibility

Virtual Reality is incredibly sophisticated; however, it lacks the key fundamentals for accessibility. While this is of course different for each developer on this platform, it is still an issue that needs to be discussed. The United Nations (2007) states "Accessibility is about giving equal access to everyone and without being able to access the facilities and services, persons with disabilities will never be fully included." (As cited in Accessibility for Disabled in Public Transportation Terminal, 2011) and in the following section, I will discuss how Virtual Reality both succeeds and fails in offering accessibility options to those that truly need it.

### 2.3.1 Visually Impaired

According to the World Health Organization (WHO), at least 2.2 billion people have some form of visual impairment. As such, many applications involving Digital Realities have a plethora of features designed to aid users with such impairments. Most modern applications come with many options for colorblindness, as well as the ability to increase or decrease the size of text that may appear on screens. Some even offer the option for on screen reading of text, and descriptions of things that are occurring if needs be. For MR and AR applications, this amount of accessibility options covers the vast majority of afflicted users and is generally enough to help them. However, VR as a whole makes it much harder for those with visual impairments to fully enjoy the experience.

As it stands, VR Headsets will always be shipped with the same lenses, and the overall design of the interior is always the same. This is a problem for those who require glasses to see properly. Many users report how uncomfortable VR Headsets are to wear while also having to use glasses. They aren't designed with them in mind, and most attachments you will find to aid with this issue are made and distributed by third parties, not the actual developers themselves. The simple solution to this is to replace the lenses that the headset comes with, ones with the correct prescription to allow the user to enjoy VR without the need of glasses. Third party companies are already doing this, with the likes of VR Optician or VR Lens Lab, but the fact that VR Developers don't give users any sort of option when it comes to prescription lenses is the core problem which needs to be addressed.

### 2.3.2 Hearing Impaired

WHO states that over 5% of the world's population require some form of rehabilitation to assist with their hearing loss. All forms of Digital Reality address their Hearing-Impaired users in the same manner, as not much can be done to truly assist. Regardless of the platform, every single modern application or form of media (Video Games, Music, Television etc.) is released with the option of subtitles. Some applications are far more in depth with their options, allowing users to modify the subtitles fonts, sizes or even location on the screen. Some applications even have an option for full audio transcriptions, allowing subtitles to be displayed which describe every sound that can be heard.

In VR, there is a limited amount of hand movement available, so while signing in VR is far more difficult, it remains a possibility. Certain online platforms have entire groups dedicated to learning and creating forms of sign language in VR games such as VRChat (The group known as Helping Hands), using the limited amount of hand motion to its full capability.

VR is limited in how it can be more accessible to people who are hard of hearing, however with the creation of more advanced controllers, which allow for more use of a user's hands in a virtual space as well as the ingenuity of communities within those platforms, it is becoming more and more accessible and is simply limited by technology.

### 2.3.3 Physically Impaired

Physical Disability is a term that covers any and all impairments that affect a person's ability to use any part of their body such as muscles, limbs, head etc (Arc Speech). Physically impaired users often have the hardest time navigating and utilizing forms of Digital Realities. However, efforts are made to accommodate people with various ailments.

VR applications can allow those with severe physical disabilities to participate in activities otherwise impossible due to their conditions. Not only are they mentally invigorating, but some VR experiences are also specifically designed for rehabilitation, and aid in the development of social and communication skills (Mileva, 2022).

The vast majority of VR headsets and games include a seated mode, meaning that you can fully enjoy VR experiences without having to stand up or move around. Physically Impaired individuals who no longer have any motor control of their limbs can enjoy voice controls for various applications, and those with limited motor controls often have custom made interfaces to allow them to enjoy the application as any other user would.

Unfortunately, they are still severely limited with how much they can experience these new forms of media. Many VR controllers simply aren't designed with this demographic in mind, and custom-made ones often cost incredibly inflated prices compared to the standard controllers. Some VR

experiences are entirely possible with the use of a single limb, but it is almost never intentionally done for those who may need it the most.

## 2.4 Conclusion

While there are certainly many options made available for those with disabilities to enjoy and experience Virtual Reality at the same level as those without, there are still many things that must be addressed before it is truly accessible to all who wish to experience it.

VR is a platform that requires a lot more effort to make it more accessible for those who need it. While there are options available, having them as third party's instead of the original developers is an issue that needs to be addressed, as many people wouldn't think to look outside of the original creators' stores.

Digital Realities are an essential part in modern society, being utilized by many people from a wide range of backgrounds and communities, and as such it is essential that they become more widely available to anyone who wants to experience them.

## 3 Requirements

### 3.1 Introduction

This section presents and analyses various solutions to User Interface (UI) design in VR. Their respective advantages and disadvantages are outlined.

A survey was conducted with a small number of respondents. The responses will be discussed and analysed. Interviews were performed in which I asked users what they felt were important features for games and UI features. Following this, a brief explanation of the results of a survey created and sent to multiple groups of people regarding the necessary requirements for the application.

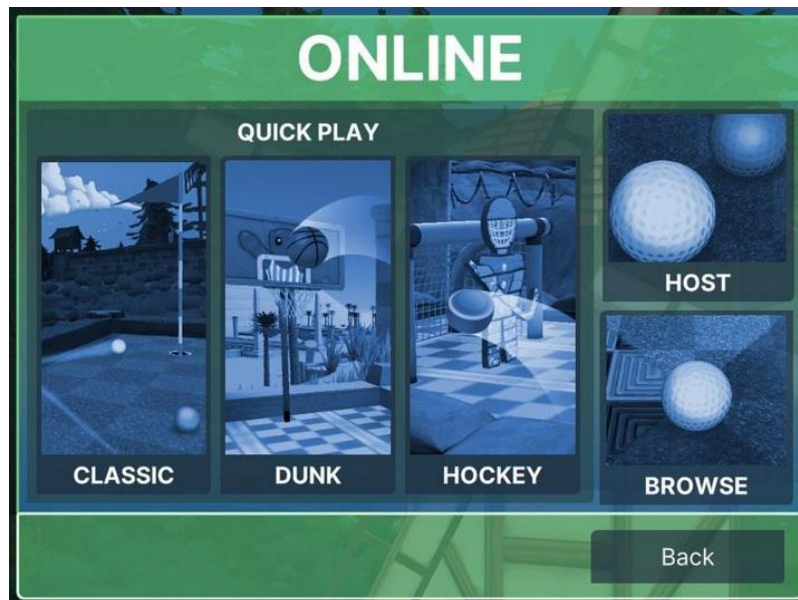
Using these responses, a short list of features was created that the application will have implemented for release, followed by another list of features that aren't necessary for completion, but will reduce the quality of the final product.

Finally, the feasibility of my application is discussed. The technologies used during development will be briefly described and the problems that may arise due to compatibility between them will be discussed, as well as solutions to those problems.

### 3.2 Requirements gathering

#### 3.2.1 Similar applications

In my research I found a few other games and used their respective multiplayer elements and VR design choices as inspiration. The first was a multiplayer golf game called "Golf With Your Friends" (Blacknight Interactive, 2020). The server browser and hosting features are very similar to what I want to create for my own game, as visible in the screenshots below.



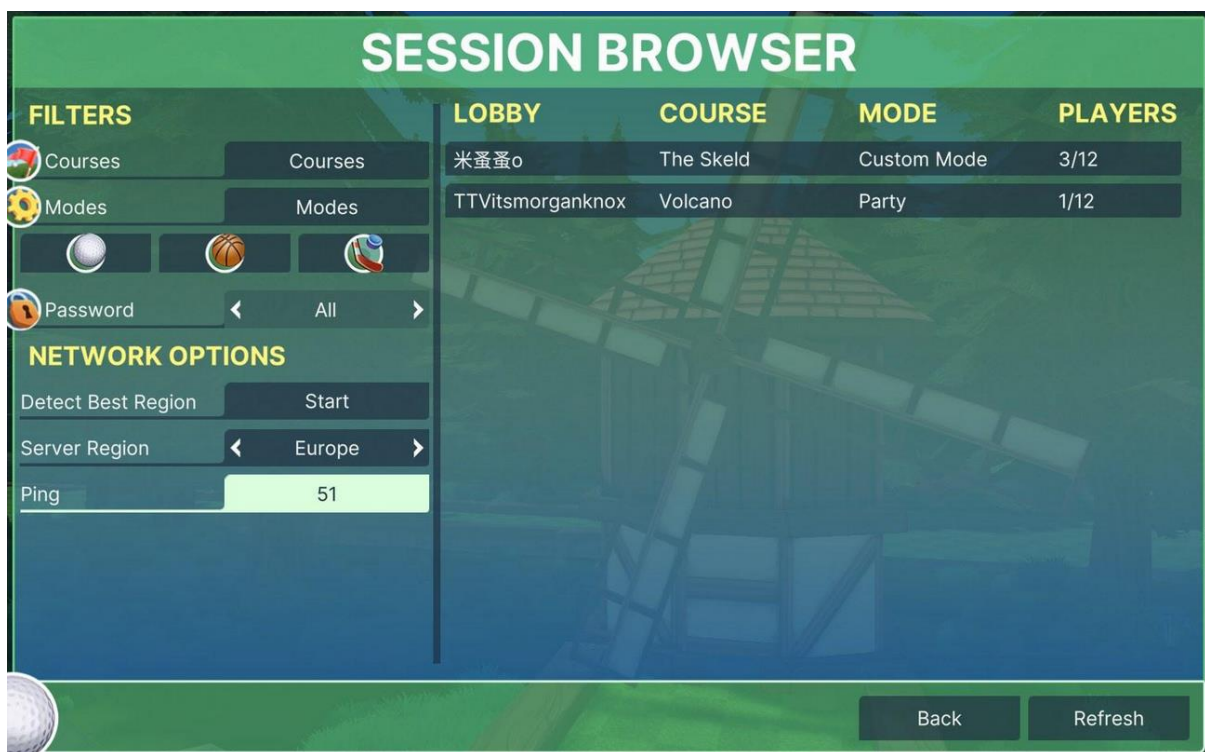
1 Golf with your Friends Multiplayer Menu

Upon selecting Multiplayer from the main menu, this comes up in screen. The 'Host' option allows the user to set up a server (publicly or privately), selecting the game mode they will play. If the lobby type is set to private, only friends of the user may join (The game is run through the popular online game store Steam, and so it uses Steam's friends list and server features).



2 Creating a Room

Similarly, the 'Browse' option opens up a server browser. This menu has a myriad of features, most prominently the ability to pick and choose any publicly hosted session and connect to it.



3 Server Browser

It also allows users to filter the available game modes, as well as select the region they wish to connect to. There is useful information on display for the sessions available such as the lobby name, course they will be playing on, game mode selected, and the number of players currently connected to the lobby.

While remaining relatively clear and simple as to what button does what function, I personally found the number of options on the first screen to be too many for my taste. While possible to gain an



understanding of what they may be, it can be confusing for some users who may have never experienced this type of game before. The multiple game modes being on display immediately with no explanation of the rules or goals could be counter intuitive, and while my game may have more than one game mode, I will be excluding this from the initial 'Play' menu.

Another game I investigated was the popular VR game "Half Life Alyx" (Valve, 2020). Specifically, I focused on the UI the game uses whilst not in any menus. The main character has gloves on each hand, and both have useful information for the player. The left hand displays the players health and the in-game resource known as Resin, while the right-hand displays nothing unless the player is holding a gun, in which case it will display the weapons remaining ammo count.



*4 Half-Life Alyx Hand Models*

These gloves also allow the player character Alyx to interact with the world in unique ways. As they are called "Gravity Gloves", the player can use them to grab objects from a distance with a helpful line indicator showing what it is they will be picking up.

This approach to an immersive UI system is very intriguing, as it allows the player to feel much more immersed in the game world due to the lack of floating text objects a normal UI system would include. The issue, however, is that the player may constantly be looking at their own hands instead of paying attention to the game around them. After a long period of time, this can get distracting and frustrating.

The third and final game I looked at for my project was the popular VR game known as "VRChat" (VRChat Inc., 2017). VRChat is an online hangout game, and has a multitude of settings and options, and different UI systems available. The most common UI form is the Quick Menu as seen below.





5 VRChat Quick Menu

This menu pops up whenever the user hits the default menu button. The menu orients itself so that it is always facing the player and positions itself just above the controller that the player selected the menu button with. The menu then follows the players hand movement, allowing them to move it wherever they deem comfortable, and interact with it using the opposite hand with a line pointer.

From this menu the player can access more in-depth features such as the games options menu, which has a myriad of audio and video options, as well as accessibility options. The world browser is accessible through the menu as well, alongside the friends list and customisation options.

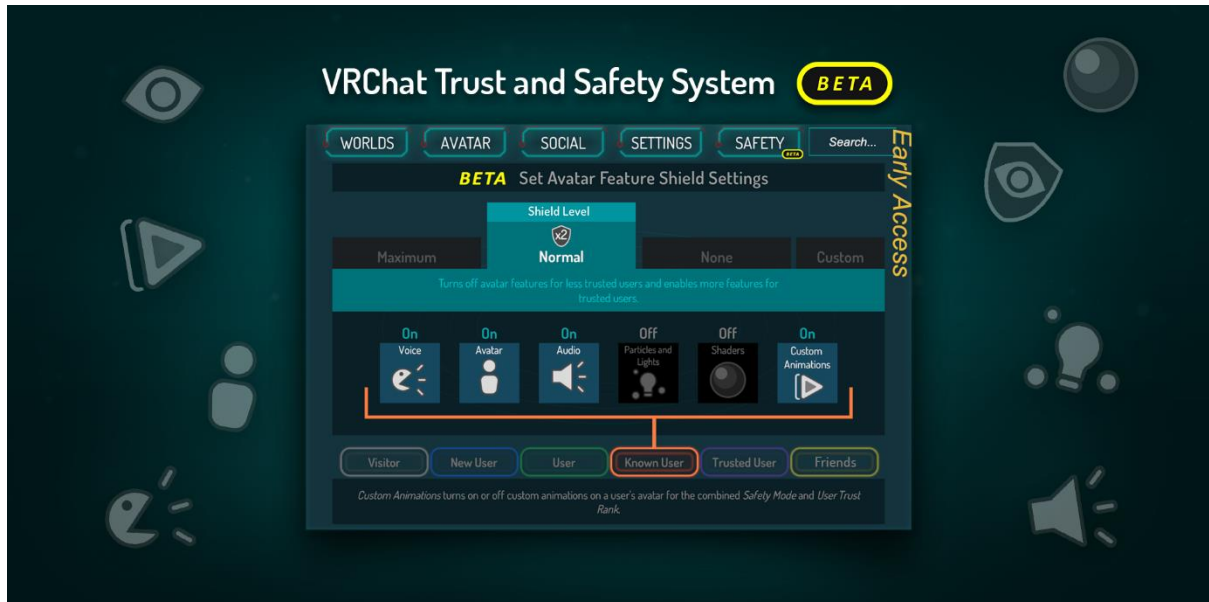
The player also has access to a smaller, more compact menu by holding down the menu button instead of simply pressing it.

6 VRChat Radial Menu



This gives players a radial menu to allow them to make changes specific to the Avatar they are wearing, as well as a smaller configuration menu for disabling other models or muting players in the lobby. It is controlled entirely by the control stick relative to the controller used to access the menu.

VRChat also contains a huge amount of accessibility options for being playing using VR equipment and those playing on a regular computer. VR users could choose between two separate movement modes: continuous or teleportation. They can also change their turning style between continuous or snap turning and can even change the degree and/or speed of which they turn at.



7 VRChat Shield System

They even include what they call the 'Shield' system, which allows players to toggle on and off every part of people's avatars, purely for safety purposes.

VRChat being a social platform has many options available to the players. Each one has been designed to ensure player safety and satisfaction, without impeding anyone's experience with the game. The different options for locomotion and the huge array of options purely for player comfort make it accessible to almost anyone who decides to try it out. However, the sheer number of menus and options available are also a big reason many people are pushed away from it. To a new player, it can be daunting learning how to navigate so many menus, and finding the options that are just right may take some time. Additionally, as VRChat consists of primarily player made content, every world or person you come across can function entirely differently, and some settings put in place may fail to function as intended.

Looking at these applications shows clearly that no single accepted UI standard has been bedded down for VR interactions. Both VRChat and Half-Life Alyx tackle the menu system and UI in vastly different ways, but neither one is inherently better than the other. Both games were developed with the UI in mind and subsequently were designed around it. Half-Life Alyx's UI works so well because the user's immersion in the game world is what the developers feel is always most important, only occasionally breaking that immersion for key moments. VRChats UI is the opposite, consisting of large pop-up menu's that never fit in the game world itself. This is not a negative however, as it would be impossible for VRChat's UI to match every single world on the platform, so keeping it consistent across everything is the smartest design choice.

However, VRChat's menu system suffers from having too many choices. Much like the first game discussed (Golf With your Friends), the sheer number of options within the UI can be overwhelming, and it's important to make sure players don't feel that way while playing. Finding a balance between a UI that fits into the game world and doesn't overwhelm the user while simultaneously providing enough functionality is something that will need to be focused on.

### 3.2.2 Interviews

Three users were interviewed prior to development on the application. These users were chosen due to their history with Virtual Reality devices and applications and were asked seven questions relating to their experience with VR.

#### **Q1: When discussing virtual reality, what would you consider to be a fundamental feature in any VR game?**

Each interviewee gave a different response for this question. The first believed that player interaction with the world is the number one priority, while the second believed that accessibility while staying immersive should be a key focus. The third answered with graphics, believing a better-looking experience to be more engaging.

#### **Q2: Do any games come to mind that implement that feature properly?**

The participants answers varied per person, but VRChat was one that came up for two of the three, as did the game No Man's Sky. Other games included Half-Life Alyx and Blade and Sorcery.

#### **Q3: On the contrary, do any games come to mind that don't?**

Each participant answered differently, with one completely abstaining from a response. The games given were Tabletop Simulator and Nostos.

#### **Q4: Are there any accessibility features that you utilize in VR Games?**

While one interviewee didn't use any, the two that have utilised snap turning as opposed to smooth turning, and the vignette feature to reduce motion sickness.

#### **Q5: What VR game, if any, have you spent the most time with?**

Two users responded with VRChat, while one chose Phasmophobia.

#### **Q6: Where does that game fail to meet your expectations?**

The similar response among all participants to this question was lack of optimisation. VRChat has little to no world optimisation requirements, and the UI can be too confusing. Phasmophobia was also remarked as being graphically underwhelming.

#### **Q7: Where does it succeed in meeting your expectations?**

While VRChat suffers with optimisation, both participants felt that the community aspect of VRChat is unrivalled, and the huge amount of content allows for a near endless number of activities. Phasmophobia was credited with being incredibly immersive, despite the poor graphics.

The results of the interview will be used in conjunction with the survey responses in order to plan what is and isn't need for the game during development. A Full transcript of the Interviews can be found at the end of the document.

### 3.2.3 Survey

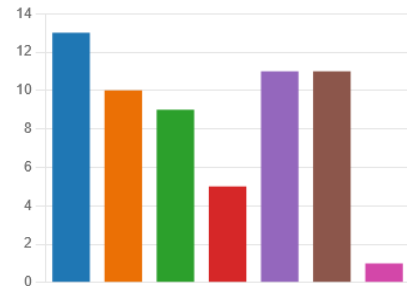
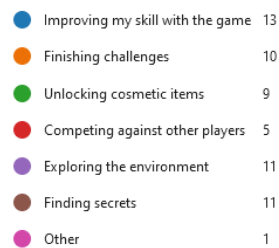
In the process of initial research for this project a limited survey with a small number of respondents was conducted in order to gain a better understanding what users would want to experience while playing in VR and what they would consider a priority.

A large majority of users are between the ages 18 to 34, with most being within the 18 to 24 range.

Following this, I asked users what they personally enjoy doing in games from a set of options. Users could choose any number they liked, and this paints a clear picture of what needs to be prioritised.

4. Out of the options below, what do you prefer to do when playing? Pick any that apply to you.

[More Details](#)



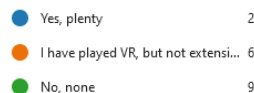
8 Survey Results 1

The most popular response was that users wanted to improve their skills over time, with environmental exploration and secret discovery coming in close second. Conclusively, the top priority for this game will be overall game feel (which upon further research has no official definition but can be loosely defined as the sensation experienced when playing video games). The simple act of firing the bow and arrow must be satisfying in order to keep players engaged. I then asked a few questions about VR and motion sickness.

I first needed to find out how many survey responders had ever used VR before in order to understand how important it would be to explain and teach them how to control various aspects of the game. Following this question, participants were asked what hardware they had experienced VR with, and the most popular response

5. Do you have previous experience using Virtual Reality?

[More Details](#)



9 Survey Results 2

given was the Quest headsets, along with HTC Vive. As the Quest is what I'm developing with primarily, it is good to know that it will be the most familiar for them. I then had an optional question asking people to list the games they had played in VR. I got many responses such as VRChat, Subnautica, Among Us and the most popular response was Beat Saber. These responses help me dictate how best to implement certain features, using the games given as inspiration.

Question eight asked if they suffer from motion sickness. While the most popular response was 'No', it was closely followed by 'Sometimes', and 'Yes' was only 2 votes behind that. While the majority suffer no ailments, combining the other two results shows a different story entirely. Some attempt to combat motion sickness must be

8. Do you suffer from motion sickness?

[More Details](#)

[Insights](#)



10 Survey Results 3

implemented in order to allow all users to experience the application without issue.

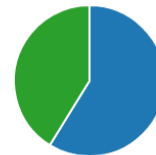
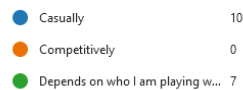
The questions following this were targeted primarily at the gameplay aspects of my application. When asked what players found most important in a video game, 10 of the 17 responses claimed that both multiplayer and single player were equally as important as each other. As such, an equal amount of emphasis will be focused on both aspects of my game. The next question asked users if they played games more competitively or casually. While the most popular response was casually it was closely followed by the third option, which declares that it depends on who they're playing with.

When asked to elaborate, 50% of the users explained that playing competitively greatly depends on the people they are playing with or against. Some claimed that they are more competitive when playing with friends or a full team of people they can communicate

10. In a multiplayer environment, do you play more casually or competitively (Do you play for enjoyment, or to win and improve?)

[More Details](#)

[Insights](#)



#### 11 Survey Results 4

with. Others play more competitively purely when competing against complete strangers. There were other factors as well, such as what game or game mode they are playing at the time, and some simply don't enjoy playing competitive games at all.

The last two questions were centred around finding out what players felt were necessary in multiplayer games. Question 12 received many responses from those that answered the question. Options and communication are seen as very important features for a multiplayer game, with players stating that multiple movement options, communication systems such as a "ping" system, or in game voice chat, and safety features such as removing players from an instance or muting their voice are required. Other responders put more emphasis on customisation, requesting the ability to customise their in-game characters or perhaps having a customisable icon that represents them. Multiple game modes were another popular response, along with the ability to see and invite friends specifically.

The final question of the survey asked for any games that the individual considered games with good multiplayer features. I received a plethora of suggestions, with the game Minecraft easily being the most common response. Alongside this were games like Mario Kart, Team Fortress 2, Terraria and Among Us. These responses will help me understand what multiplayer features are truly important when developing a multiplayer game and will aid in my development process.

## 3.3 Requirements modelling

### 3.3.1 Personas


After categorising and studying the survey results, a few user personas were developed from the answers given in order to get a better understanding of the people who could potentially play the game.

### Justin McCormack

<p><b>Gender</b> Male</p> <p><b>Age</b> 17</p> <p><b>Occupation</b> Student</p> <p><b>Education</b> Secondary Level</p> <p><b>Marital Status</b> Single</p> <p><b>Location</b> Dublin, Ireland</p>	<p><b>Motivations</b></p> <p>Justin plays video games extensively after school and on the weekends. He sees it as the best way to spend time with his friends, but understands that his education is important as well. As such, he never lets it overshadow his school work.</p> <p>Justin is fairly competitive when he plays, striving to improve his skills all the time, but is able to unwind and enjoy himself when he's playing with his friends. That doesn't stop him from proving to them just how good he is, however.</p>
	<p><b>Priorities When Playing</b></p> <ul style="list-style-type: none"><li>• Constant Improvement</li><li>• Competing against other players</li><li>• Finishing Challenges</li></ul> <p><b>Frustrations when Playing Games</b></p> <ul style="list-style-type: none"><li>• Poor multiplayer functions</li><li>• Lack of clear skill improvement</li><li>• Limited time challenges or modes</li></ul> <p><b>Personal Favorite Games</b></p> <ul style="list-style-type: none"><li>• Apex Legends</li><li>• Call of Duty</li><li>• Deep Rock Galactic</li></ul>

12 Persona One

### Frankie Simmons

<p><b>Gender</b> Non-Binary</p> <p><b>Age</b> 28</p> <p><b>Occupation</b> Business Associate</p> <p><b>Education</b> Masters</p> <p><b>Marital Status</b> Single</p> <p><b>Location</b> Newcastle, UK</p>	<p><b>Motivations</b></p> <p>Frankie uses video games as a method to de stress after their stressful work days. They have no interest in being competitive and prefer to play games that let them unwind and socialize.</p> <p>Their job allows them to afford any consoles and video games that interest them and as such they have no issues with experimenting. Preferring to play alone, Frankie tries their best to escape and enjoy their time when they sit down to play. They still do play competitive games with their friends when asked however.</p>
	<p><b>Priorities When Playing</b></p> <ul style="list-style-type: none"><li>• Relaxation</li><li>• Solo Experience</li><li>• Customization and self Expression</li></ul> <p><b>Frustrations when Playing Games</b></p> <ul style="list-style-type: none"><li>• Forced online connection</li><li>• Lack of cosmetic options</li><li>• Content locked behind difficulty curves</li></ul> <p><b>Personal Favorite Games</b></p> <ul style="list-style-type: none"><li>• Animal Crossing</li><li>• Dinkum</li><li>• Valorant</li></ul>

13 Persona Two

The first is a young student named Justin McCormack. Justin has an even balance between enjoyment and competitiveness when he is playing video games. He often will enjoy having fun overall, especially when playing with friends, but still strives for improvement. As he is still in school, it's important that he doesn't disregard his education, and as such doesn't like limited time events or challenges due to his own time constraints.

Next is 28-year-old Frankie Simmons. Frankie plays video games much more casually, opting for more relaxing titles such as Animal Crossing or Stardew Valley. Their work life is stressful, and so they prefer to use their gaming downtime as a means of unwinding, and prefer not to worry about points or competition, and just have fun. However, they do occasionally play competitive games at the request of their friends but are never one to take it too seriously.



## Emma Conroy

**Gender** Female

**Age** 22

**Occupation** Retail Worker

**Education** Bachelors Degree

**Marital Status** Married

**Location** Chicago, USA



### Motivations

Emma spends most of her free time playing video games competitively. She plays on a team with 4 of her other friends, and when they aren't playing together, most would be practicing by themselves by playing games alone against other people.

She takes her hobby very seriously, hoping to turn it into her career, and would rather not play a game if some competitive feature isn't available. The few times she doesn't take her hobby seriously is when playing with her partner, or indeed when she very rarely plays a more relaxing game.

### Priorities When Playing

- Improving her skill
- Achieving higher ranks
- Winning

### Frustrations when Playing Games

- Lack of a proper competitive mode
- Poor communication options
- Inability to play with friends

### Personal Favorite Games

- Call of Duty
- Counter Strike
- League of Legends

Finally, we have Emma Conroy. Emma is fresh out of college and yearns for competition. Most if not all her free time is spent playing competitive games such as Counter Strike, League of Legends or Call of Duty. Emma plays to such an extent because she is a part of a team and dreams of becoming a professional in the video game industry through Esports. When able she will practice with her team, but even when not available she is playing by

### 14 Persona Three

herself and rising in the ranks as best as she can. The only downtime with she has with her hobby is when she's playing with her partner, and will often play more relaxed, cooperative games to have fun.

These personas were developed to create three extremes of players from the results of the survey, and act as a guide for the development process for features in the game.

### 3.3.2 Functional requirements

Functional Requirements are features and functions that must be implemented for the end product to work as intended and allow users and testers to use the product adequately.

The application should have the following features:

- Multiplayer Server browser and Lobby system
- Fully functional bow and arrows
- Competitive scoring and high score system
- Engaging, non-intrusive UI
- Single player and multiplayer content
- Full movement in VR
- Small but fully explorable environment
- Local and online leader board
- Achievement system/unlockable content
- Player models with slight customisation options
- Voice Chat

### 3.3.3 Non-functional requirements

Non-functional Requirements define how the system should perform according to Altexsoft (2021), for example loading times, multi-user instances without stuttering etc.

The application will have the following non-functional requirements:

- Accessibility options

A selection of accessibility features will be available in the options menu to allow users to customise their experience with the game.

- Performance

The game will be developed using a low poly art style and each scene will take place in a small, enclosed arena, ensuring high performance.

- Platform

The game must be developed to run on the Oculus Quest 2 VR headset. It must run smoothly and be fully operational with the hardware.

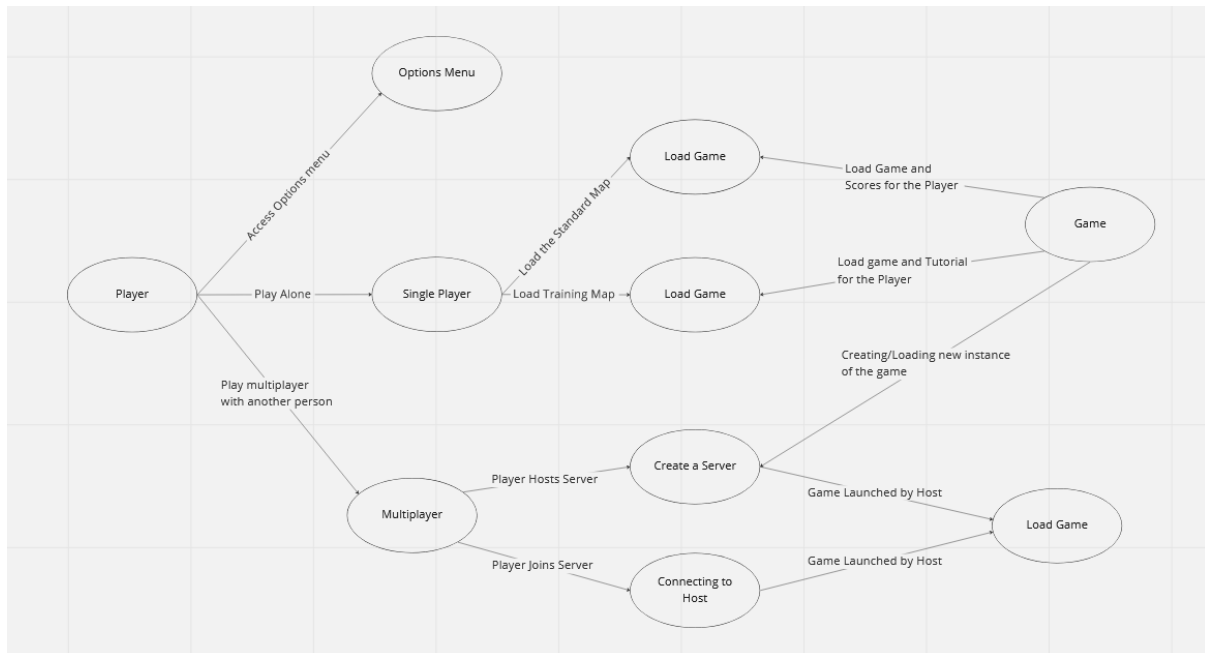
- Usability

The game must clearly demonstrate how to navigate the scenes and use the various equipment available in the game world.



### 3.3.4 Use Case Diagrams

Below is a Use Case Diagram of the archery game being developed.



15 Use Case Diagram

Players will first choose to play the game or navigate to the options menu to tweak the settings to their liking. Upon choosing to play the game, they will then decide if they will be playing alone or with multiple users. If Single Player is selected, they will be prompted to play either the standard mode or the training map. In both cases, the default map will be loaded with slightly different parameters. From the user Case Diagram, the game will load the game for the player but in the standard option, the saved scores will be loaded alongside it. If the practice mode is selected, a tutorial will be loaded alongside the map for the player.

Alternatively, if multiplayer is selected, the player will then be able to either join an already existing server or host a new server of their own. If choosing to host, the game will then create a new instance of the game for the player to oversee. The host will then wait for another user to join their room and then launch the game, loading the game world. If the players choose to join a room, they will first connect to the room being hosted, and simply wait for the host of the room to launch the game instance.

## 3.4 Feasibility

The project is going to be developed using Unity as it's game engine. As I have some previous experience developing with Unity, it felt like the optimal choice. Unity is a free game engine that can be downloaded on any computer system and lets anyone to start developing and coding in a virtual environment that allows the developer to actively test what they are working on in real time. I will be referencing the Unity Documentation as well as any number of tutorials online created by other Unity developers to aid in the creation of the project.

Within Unity there is a VR framework known as the XR Interaction Toolkit. This toolkit installs several premade scripts and functions into Unity, along with a large number of prefabs and controllers. It

allows developers to create a VR project from scratch, and all they need to do is import the XR Interaction Manager into their scene. As this is added from within Unity, there will be no compatibility issues.

Finally, I will be using Normcore to develop the multiplayer for the game. Normcore is a multiplayer framework built specifically for Unity. It is designed with multiplayer needs in mind, such as Voice Chat (VOIP), networked physics, persistent spaces, and most importantly support for XR, the VR toolkit that is being used for this project. As such, this project should not have any compatibility issues considering each of the technologies are designed with each other in mind.

The only concern is that the version of the XR Interaction Toolkit I may use is an older version, and Normcore may be designed with the latest version in mind. If that's the case, I simply must rebuild the project with the latest version of the XR Interaction Toolkit. This will take some time but is the simplest way to solve the problem.

### 3.5 Conclusion

This initial chapter has discussed the various actions that have been taken to properly prepare for the development cycle of the application. The application will be a VR Archery game developed for the Oculus Quest 2, utilising the Unity Game Engine, the XR Interaction Toolkit and Normcore as it's multiplayer framework.

The key area this project will explore is VR Game Development and Multiplayer Networking, and the previously mentioned technologies will be used to develop these systems.

The requirements for the game have been clearly stated, and various diagrams have been modelled as a visual aid. Additionally, the creation of Personas will further assist in production, as they will be used as a baseline for what users want when playing this game. The functional requirements provide clear goals for the project and will be referenced often during development.

## 4 Design

### 4.1 Introduction

The application for this project is an APK to be launched from an Oculus Quest. The APK will be created and built with the program Unity, using the XR interaction toolkit and Normcore Framework. It will be developed using the Oculus Quest 2 for playtesting and debugging purposes.

The application will feature both single player and multiplayer modes, an internal high score system, a challenge and rewards systems, voice chat and a competitive multiplayer system.

The Program Design portion will discuss the various applications being used during development, such as the technologies utilised, and an explanation of the main game engine used. A breakdown of the application architecture is included, clarifying how the various systems communicate with each other.

Secondly, the UI portion will first discuss the core thought process behind designing the UI and will then transition into more in-depth topics including a Wireframe prototype and a hand drawn Storyboard for a more coherent vision, a User Flow Diagram demonstrating how users will interact with the program, and a section dedicated to the overall style of the program where colour schemes and level design are explained.

### 4.2 Program Design

The program will be split into three sections for ease of work during development.

First, the core VR mechanics will be implemented and tested thoroughly. This includes a basic scene with a working VR player controller with both movement and interactions. Additionally, the bow and arrow objects will be implemented and functioning properly before further development. Finally, some basic scenes and scene loading scripts will be created to allow transitioning between different levels.

Secondly, the multiplayer aspect of the game will be tackled. As this is one of the core functions, it is best to ensure that multiplayer works properly as soon as possible. Having never worked on Networking previously, tackling the most complicated part first and foremost ensures an easier development cycle further down the line. This includes the implementation of the Normcore Framework, along with synchronising objects that the players can interact with such as the bow and arrow, and the score system.

Finally, single player will be worked on once the multiplayer is functioning properly. As this requires no object synchronisation or networking, the previously created assets will have no issues working in an offline environment. The score system and timer will be modified to work offline, along with the bow and arrow objects.

Once these sections have been developed to a playable standard, work will begin on the overall games design. This includes both environmental and UI elements. Scenes will be developed with Assets found on the Unity store or other online marketplaces and will be tested afterwards to ensure they interact properly with the games objects and systems.

#### 4.2.1 Technologies

The application required the following technologies:

- Unity (Unity Technologies, n.d.)
- Normcore (Normcore, n.d.)
- XR Interaction Toolkit (Unity Technologies, 2023)

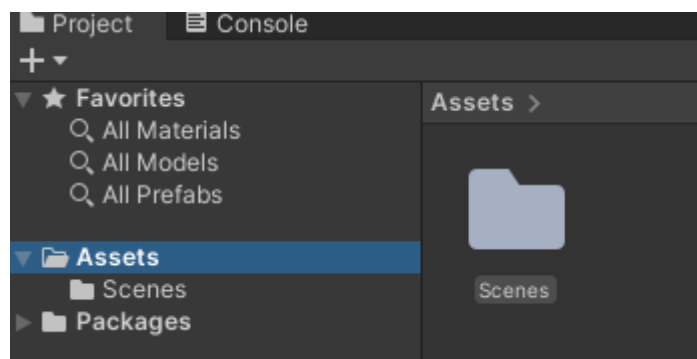
These technologies were chosen because of the familiarity and extensive documentation written for each one. Similarly, these technologies were specifically designed to interact with each other, and as such the development process is significantly simplified as a direct result. There are no errors present or work arounds needed to get them to work with each other. Unity is an incredibly robust and modifiable game engine and is perfect for entry level game development projects. Normcore allows the creation of multiplayer games and synchronisation of objects within those games simply by adding it into the projects and modifying assets to be stored in the server, and the XR Interaction Toolkit is the core VR Framework for Unity, with everything that a VR game needs.

Before landing on these technologies, others were considered. For the game engine, the second choice was the Unreal Engine, developed by Epic Games. This was considered, but ultimately decided against specifically due to familiarity with the Unity Engine, having developed on it beforehand. Additionally, the Unreal Engine puts a lot more emphasis on the graphics of video games. While graphics can be important, sacrificing the flexibility of Unity for a nicer looking game was the incorrect decision, and so Unreal was left to the side.

On top of this, deciding on a Multiplayer Framework to use within Unity was the next big decision. Unity has many multiplayer frameworks available to use, varying in complexity and usability. Before settling on Normcore, Photon was considered for the project. Photon is designed specifically for multiplayer and networking; however, it appears lacking in some parts. After some research, it was clear that Photon was not designed for VR games and would be a struggle to implement. Normcore on the other hand was designed with the XR Interaction Toolkit (VR) in mind and focuses on that largely in its documentation and website. Additionally, Normcore's implementation is a lot simpler than Photons, coming with many premade scripts that make syncing users easy.

#### 4.2.2 Structure of Unity

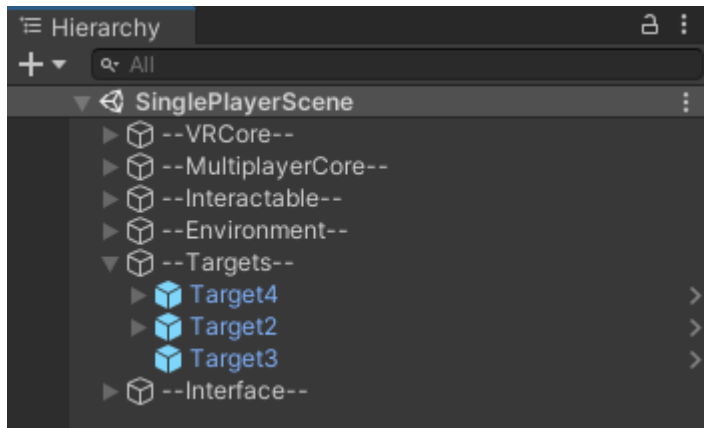
When a new project is created within Unity, developers are met with many options available to them, displayed in the Unity Editor. First, let's look at the Project window on the bottom of the Editor. This section has an open Assets folder with a Scenes folder within it, and a second Folder called Packages. The assets folder is where every Game Object, Script, Prefab, Material/Texture etc are stored. The Packages folder below this is where Unity stores dependencies for the objects in your scene, and objects installed through Unity's package manager.



16 Unity File Explorer

In Unity, a scene is something that contains the objects within a game. Games created in Unity will generally have multiple scenes, such as a main menu, various levels or even loading screens. The objects stored within these scenes are known as Game Objects. A Game Object can be any item

present and visible within that scene. For example, in a game set in a village, all the trees, foliage, buildings and even characters walking around are labelled Game Objects. Game objects themselves consist of various components such as scripts or models, and these then have properties assigned to them such as position, rotation and scale.



18 Unity Hierarchy

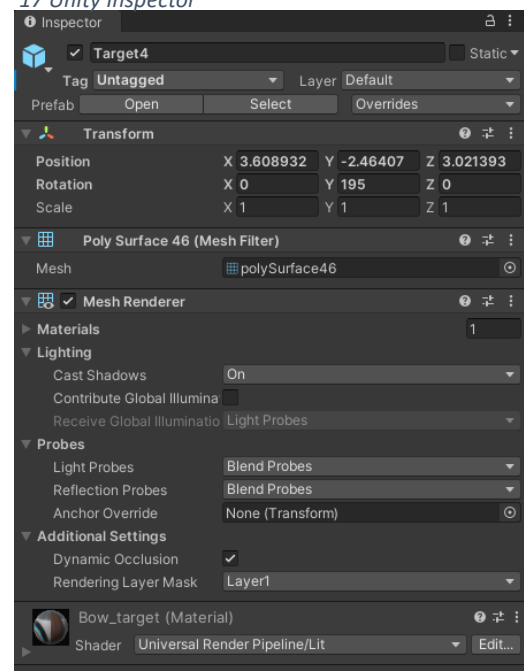
you are currently working on, and what objects belong to it. Objects in the hierarchy are stored in the game's primary memory and are loaded every time that the scene is loaded. Objects in the assets folder are stored in Unity's secondary memory, which is generally the Hard Disk on a computer.

Upon selecting a game object in the hierarchy, a window on the far right will open containing all the components within that game object. The inspect window will contain all the information related to any object within your Unity project upon selection. This also lets developers edit the properties of those objects, as well as apply new components to them if necessary. In this image, the 'Target4' object was selected from the hierarchy as evident by the name at the top of the inspector. The object can be transformed and has been given a mesh renderer as well as a material, which is what displays the object in the scene as a physical item. These are the object's components. More examples include a 'Rigidbody', which gives the object collision detection and physics, or a 'Collider', which allows the object to interact with the environment through script events.

A script is a piece of written code that allows developers to give objects specific properties, such as speed or jump height. A script can use other components attached to the object it is attached to as reference points to be used within itself. For example, the Bow object in this scene has a Bow Request script attached to it. The bow has been assigned the ID of -1, and when it is grabbed, that value updates to match the ID of the player who grabbed it in the scene, all of which is done due to the code in the Bow Request script.

The objects present in a scene can be easily found by looking at the scene's Hierarchy. The Hierarchy can be found in the top left corner of the Unity Editor, and within it is every single game object that is in the currently loaded scene. As described above, each of these Game Objects has scripts, models, and things like colliders or tags assigned to them. At the very top of the hierarchy, the name of the scene is displayed, so you know which scene

17 Unity Inspector



19 Ownership ID Code Example

```
// Update is called once per frame
void Update()
{
    if (Bow.isSelected)
    {
        RealtimeTransform.RequestOwnership();
        ownership = RealtimeTransform.ownerID;
        //Debug.Log(ownership);
    }
}
```

### 4.2.3 Structure of Normcore

Since Normcore functions as a Unity plugin, a new VR project must first be created before it can be implemented. Upon creation, Normcore can be installed by downloading it from the Unity Asset Store, and installed into the Unity Scene by navigating to the Package Manager and installing it from the My Assets tab.

Upon installation, a new prefab is added to the Hierarchy called “Realtime + VR Player” and a number of scripts are added to the project. In the inspector, the Realtime and Realtime Avatar Manager scripts are attached to the prefab. The Realtime script contains all the information required for hosting the scene on Normcores services.

An App Key is generated from Normcores website and functions as the password to access the server and it’s functions. The Room name allows scene loading scripts to reference different scenes and load into the correct one when prompted.

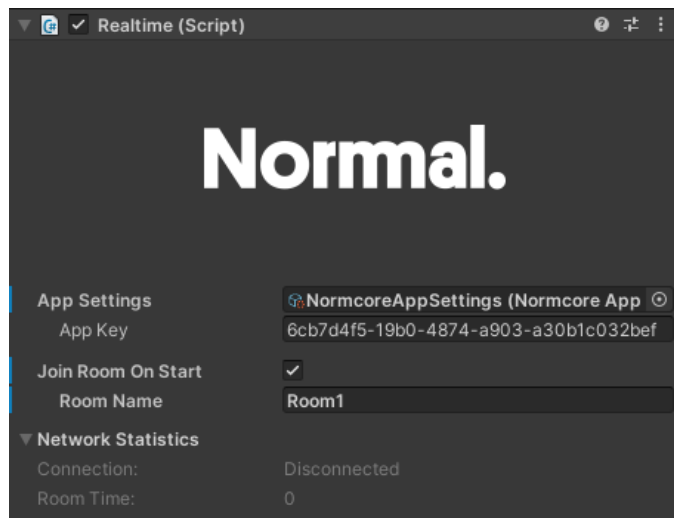
Below this is the Realtime Avatar Manager script, which is what syncs the local player movement up with the local avatar and pushes that information to Normcore to sync it up between players.

As stated previously, Normcore comes with several scripts that are used to sync up various objects between players. The most frequently used Scripts are the Realtime View and Realtime Transform scripts. These scripts are added to every object in a scene that move either independently or from player interaction. In this case, they were added to the Bow and Arrow objects in the multiplayer scenes so that players could see them being fired at targets.

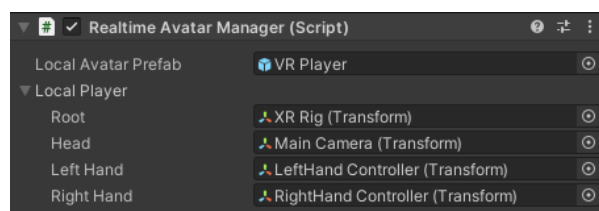
Normcore also adds Models, which are standalone scripts not attached to any object, but referenced by other scripts in the scene. Models contain pre-set data as written by the developer such as player score or scoreboard information. The models take data from local sources, such as a player’s score, and upload it to Normcores servers. This information is then relayed to any other user in the instance and updated as necessary. Models are compiled automatically by Normcore after the first few lines of code dictating what it stores is written.

```
[RealtimeModel]
public partial class ScoreboardModel
{
    [RealtimeProperty(1, true, true)]
    private string _scoreBoardText;
}
```

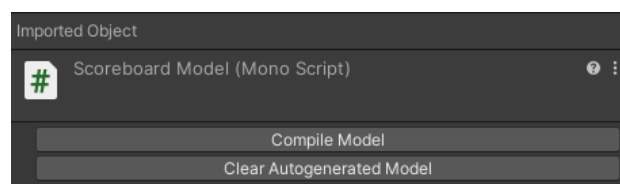
22 Initial Model Code with Scoreboard Text Parameter



20 Realtime Script



21 Realtime Avatar Manager Script



#### 4.2.4 Structure of XR Interaction Toolkit

The XR Interaction Toolkit is a package available within the Unity Game Engine. When a new Unity project is created, the XR Interaction Toolkit can be installed from the Package Manager under the Unity Registry tab.

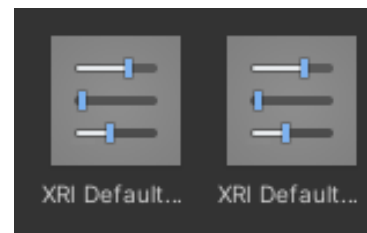
Upon installation, right clicking on the hierarchy will show a new option called XR. Within this new tab are a number of VR components, and selecting the Stationary XR Rig will replace all the basic scene components with the starter VR Components.



23 Stationary XR Rig

The main camera is responsible for the players head tracking, and the left and right controllers are responsible for the players left and right-hand tracking.

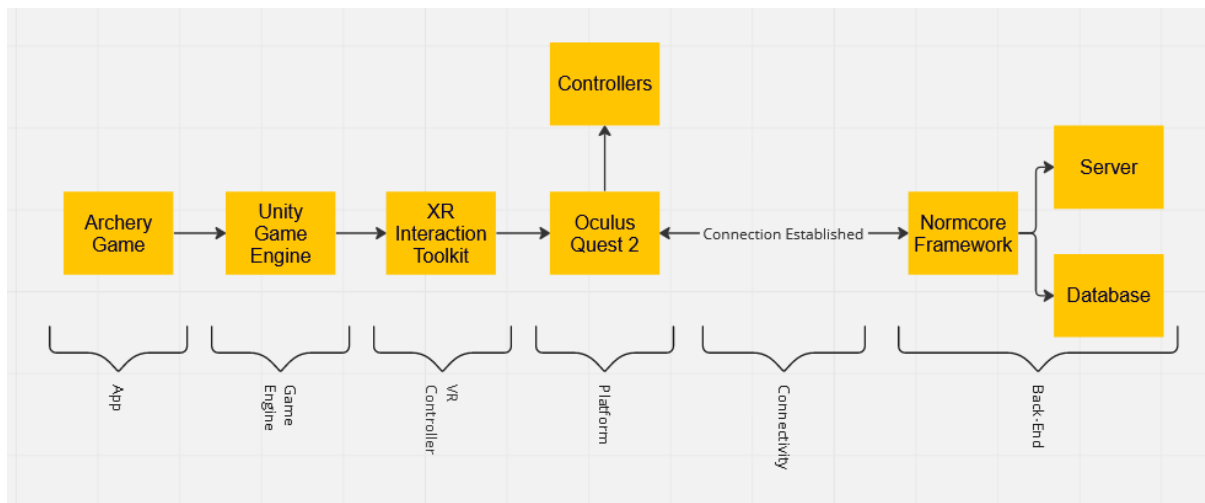
The XR Interaction Toolkit also comes with pre-made controller layouts, which can be assigned to the individual controllers present in the scene.



24 XR Default Button Inputs

Additionally, the XR Interaction toolkit provides developers with new UI elements that are compatible with VR controllers. This allows for the creation of a UI Canvas, which can be placed anywhere in the scene and functions identically to a Normal UI.

#### 4.2.5 Application architecture



25 Application Architecture for Project

Above is a diagram depicting how each part of the application interacts with each other. Within Unity there is both a sound and graphic engine that controls the respective components by default, but the XR Interaction Toolkit is a separate plugin that needs to be enabled by the developer within Unity to allow for VR control and development. The platform selected for the application is the Oculus Quest 2, but this could also be developed for other VR devices such as the Valve Index or the HTC Vive. These platforms are responsible for controlling both the camera and arm movement within the game. The connection and back end are all handled remotely by Normcore.

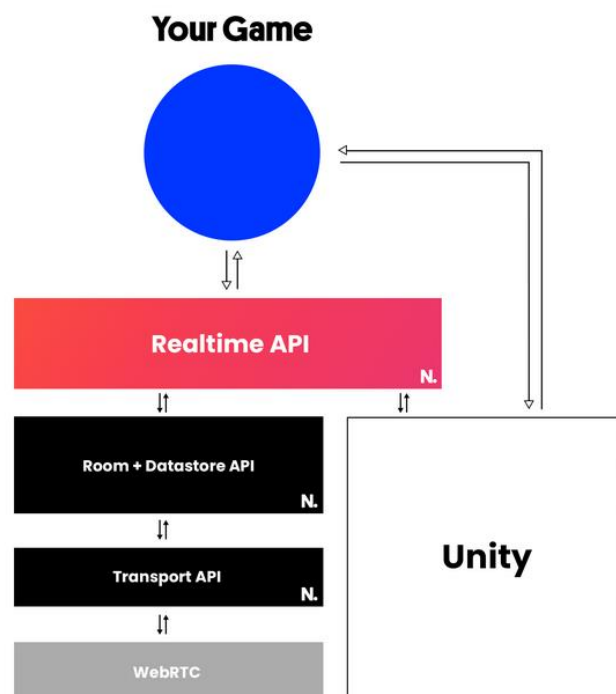
Normcores architecture consists of multiple layers which all serve a different purpose.

The Realtime API is responsible for connecting the application to Normcores datastore. This is also responsible for the synchronisation of all objects in a scene to the Normcore Datastore.

The Room + Datastore API is responsible for the connection to the room server and the datastore. It stores object information such as position or scale and will ensure that those stay consistent across all clients.

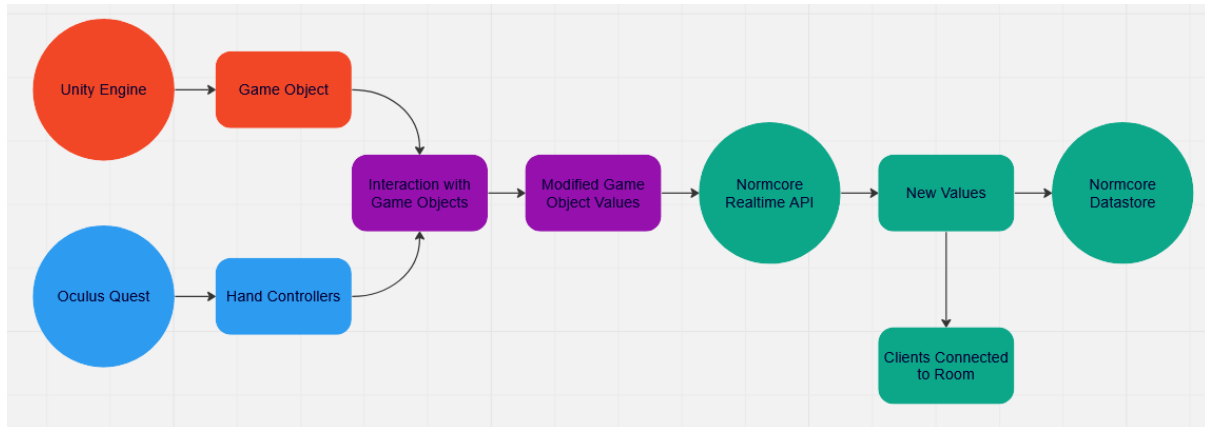
The Transport layer is responsible for communication and gets things from one point to another, working in tandem with WebRTC.

WebRTC handles real time communications such as Voice Chat and instant messaging.



26 Application Architecture provided by Normcore Documentation





27 Figure displaying Interactions between Application objects and Frameworks.

This final schematic shows how the Game Engine, Normcore and the Platform all interact with each other. From left to right, we first have the Unity Engine connected to a game object and the Oculus Quest connected to its controllers. Both are involved with interacting with objects in game, which subsequently modifies the game objects values (Specifically it's positional data).

The objects new data is then sent onto Normcores Realtime API where it is saved as the object's new values. These new values are pushed to any clients connected to the room so the position for them is updated accordingly, as well as the Normcore Datastore which internally saves the objects position.

### 4.3 User interface design

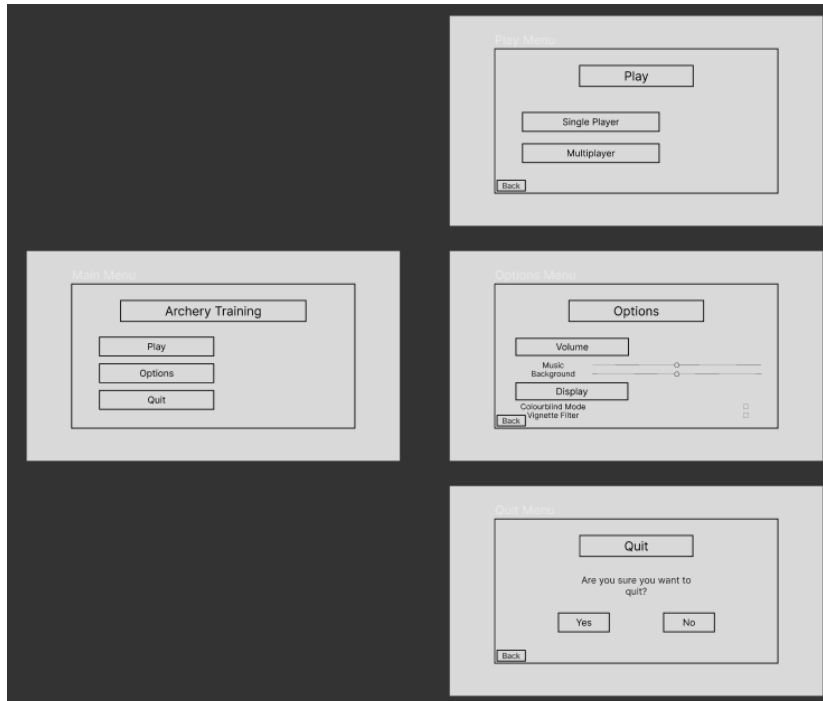
The UI for this application was developed with simplicity and immersion in mind. The menu's such as the options menu or server browser only have the necessary information on display, to prevent the user from being confused or overwhelmed. When in-game, the UI will be built into the game world. Buttons will be attached to fences or walls like they were attached there in the game world, keeping the players view clear of clutter or unimportant information.

For example, the main menu will be a large signpost with buttons the players can interact with using a pointer coming from their hands. The menu will change depending on what selection the player makes. In game, the players menu will be off to the side and will function similarly, allowing the user to access the options or quit to the main menu. Additionally, menu elements will be outlines one a user is hovering over them, for additional clarity.

The only intrusive UI will be a popup in the single player section of the game prompting the player if they would like a short tutorial on the game's controls.

### 4.3.1 Wireframe

A wireframe prototype was developed using Figma to gain a better understanding of what the end goal was for the product.



As the game is in VR, most of the wireframe focus was on the menu system that would be present in-game. The main menu would have 3 separate options for the user.

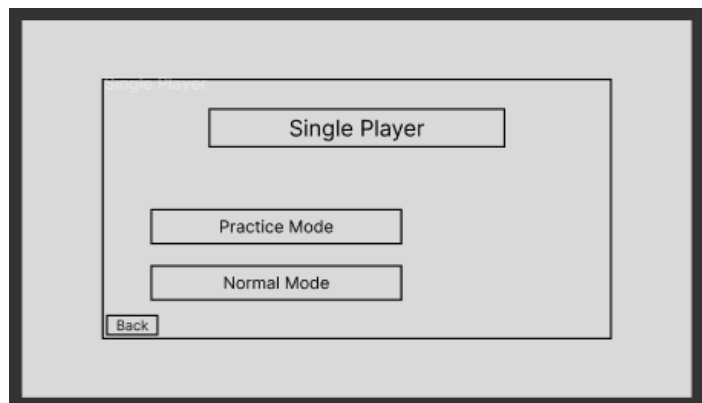
“Play” brings the user to the Play menu, where they then have the choice between single or multiplayer.

“Options” brings the user to the options menu, where they can modify the game audio or display settings to suit their needs, along with activating any accessibility options they require.

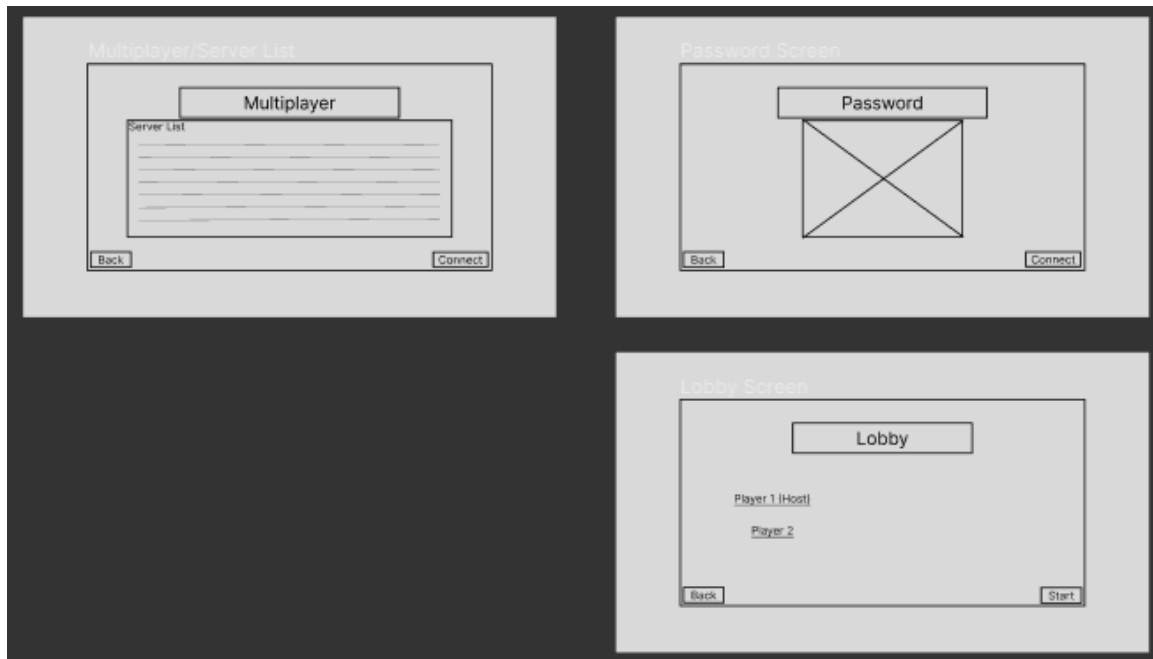
Finally, the “Quit” option simply brings players to a secondary menu where they can confirm they wish to exit the game or cancel and return to the main menu.

Upon selecting the Single Player option, users will then be given a choice to play the game normally, or first go to a separate practice mode. Both options load the player into the game scene after this point.

Alternatively, if Multiplayer is selected the user will be sent to a server list screen. On this screen the user can select a server that they wish to join or create one themselves in order to host a game. The server list will consist of room names and a lock icon next to rooms that are password protected. From this screen, users that attempt to join these rooms will either join immediately without prompt or will be prompted to insert the password required to gain access. Once the user has connected to the room, both player names will be



displayed on the left side of the screen, and the host of the room will be able to launch the game, loading both players into the game world.

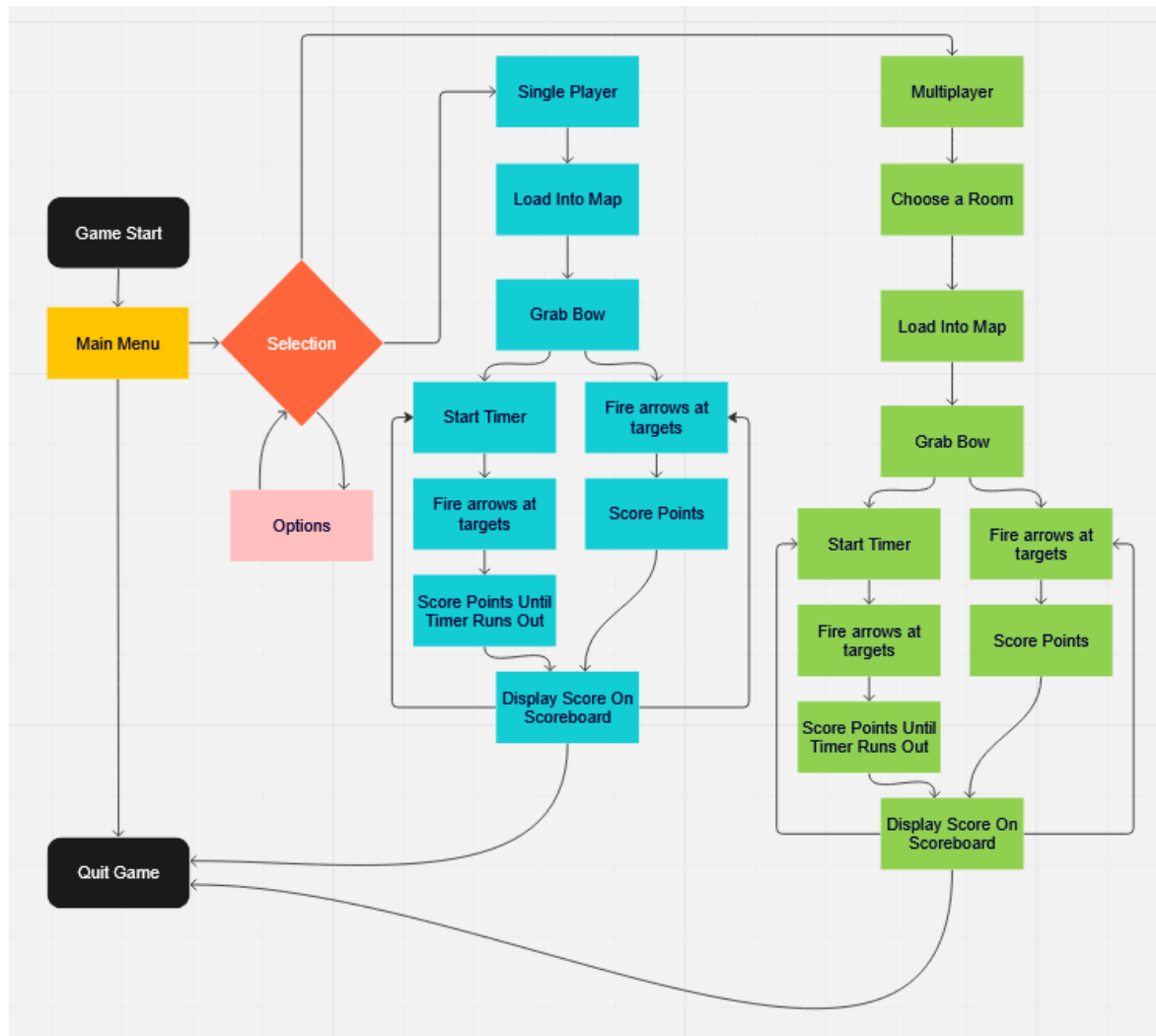


A mock scene was drawn and imported to Figma to display what the VR view might look like in-game as displayed below.



In this multiplayer view sketch, the player can see down the archery range at their targets with the options to look left and right. On the right, the other player is visible alongside them separated only by another fence. To the left, the bow is displayed on the wall along with an option and quit button in either corner. The quit button takes the player back to the multiplayer server menu, and the options button opens the same options menu available on the main menu of the game.

#### 4.3.2 User Flow Diagram



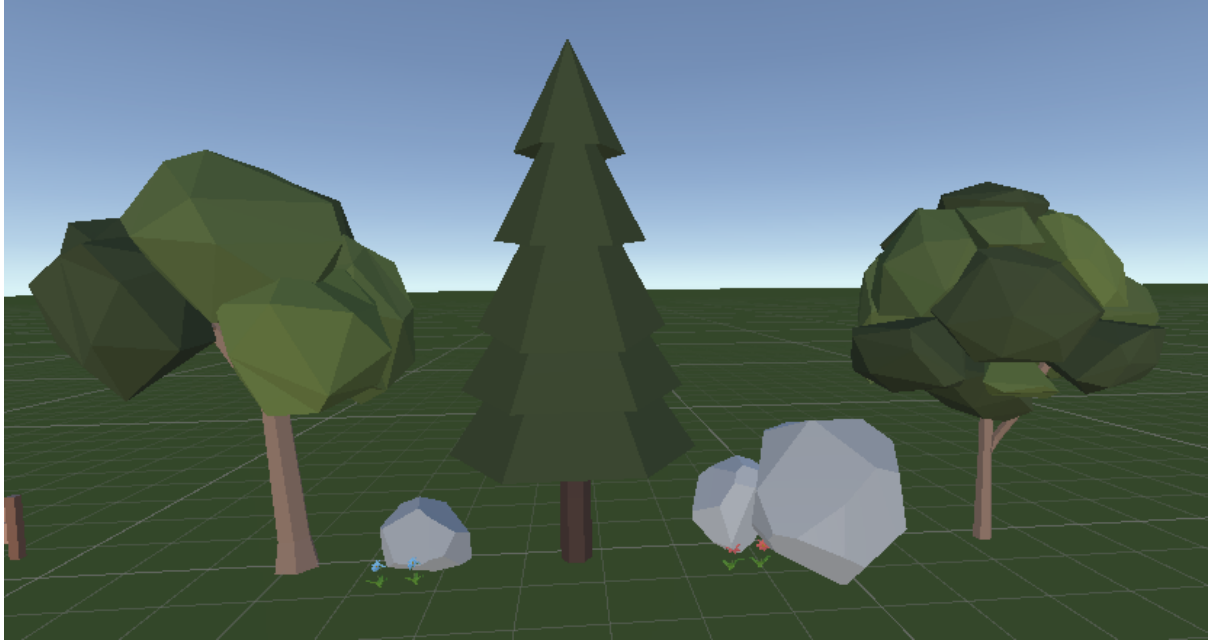
28 User Flow Diagram for the Application

The diagram above shows how users will interact with various parts of the application. From the main menu, users will have the option to quit the game entirely, open and modify the game options or choose to play in either single or multiplayer.

Depending on the chosen game mode, players will either immediately load into the single player scene or will instead be prompted to choose a room to connect to. After choosing a room, the core gameplay of both the single and multiplayer sections are functionally identical. Players will use the nearby bow to score points by firing it at nearby targets. In the multiplayer version, the coloured bow dictates which targets the player needs to fire at in order to score points for themselves. A timer can be started to initiate a timed mode, where players score as many points as they can before the timer reaches zero. Once the timer reaches zero, players can choose to start the timer again which resets the number on the scoreboard, or they can choose to quit the game entirely.

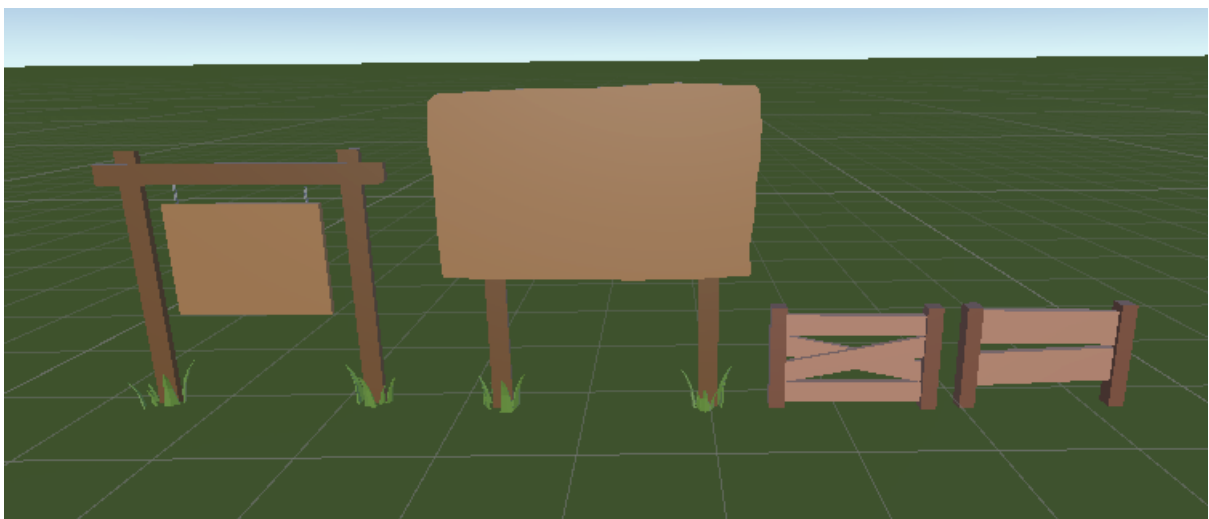
#### 4.3.3 Style guide

The game's environment will consist of natural enclosed forested areas, and the colour scheme will reflect that. It will utilise low-poly models with a cartoon shader. The man-made objects will be made of wood or straw, and as the game is in VR the colours will be softer and less intrusive for the player, as to not be hard on the eyes. There will be various shades of green and brown for the foliage, with flowers occasionally breaking the monotony with others such as red or blue, with grey rocks scattered about the various scenes.



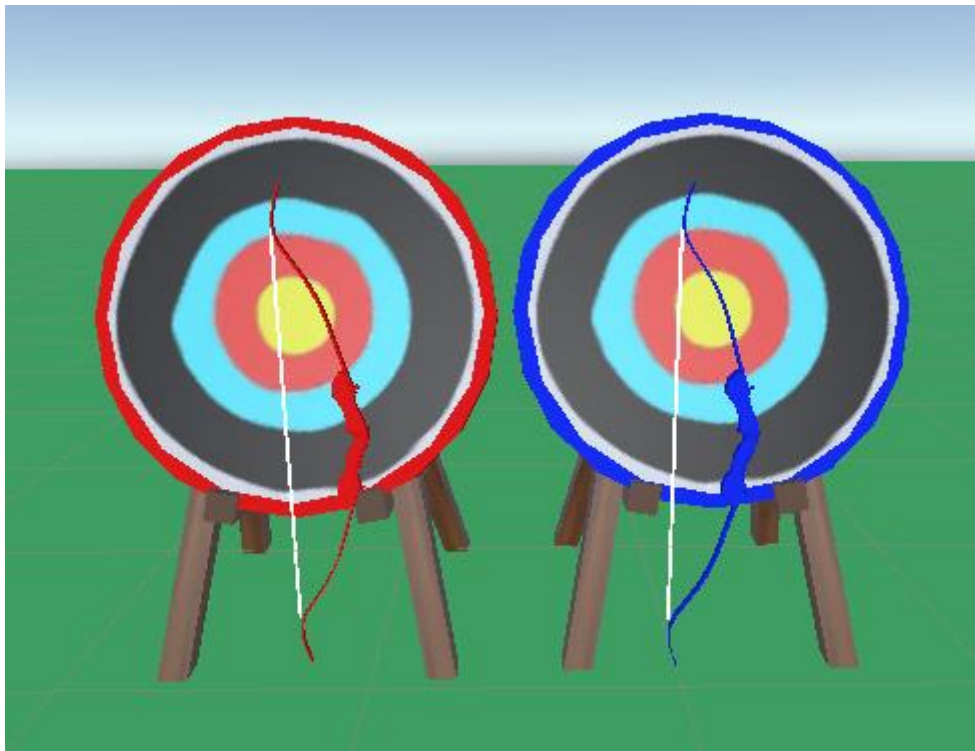
*29 Various Environmental Objects*

Fences and signs will all be a lighter shade of brown to differentiate themselves from the large amount of brown already in the environment and to give themselves a more processed look, along with the player bow and arrows fired.



*30 Signs and Fences Chosen*

In multiplayer, the players bows will be two different colours matching the targets down range. This is to differentiate them and make it clear that the bows will only work when the arrows fired hit the targets matching their colours.



*31 Recoloured Targets and Bows for Multiplayer*

Finally, the font chosen will be Liberation Sans. Visually it is incredibly clear especially when paired against the wooden sign backdrop, and more complicated fonts may be harder for players to read while in VR. All text will include a black outline for further visual clarity throughout the application.



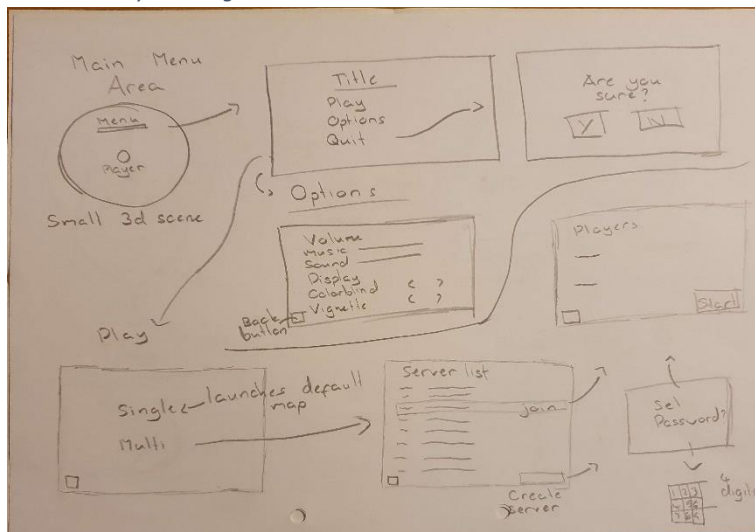
*32 Updated UI Design*

The colours are presented in a matte shade to ensure player comfort while in VR, as highly saturated colours may irritate players or cause unwanted side effects such as headaches or eye strain.

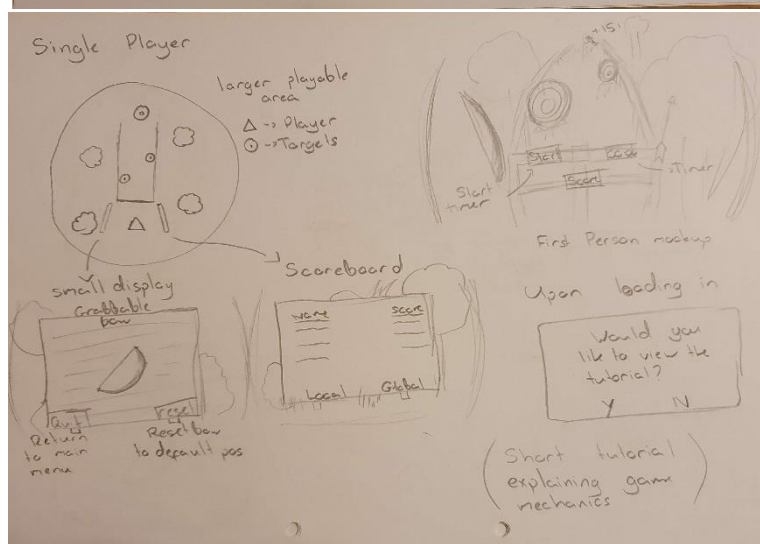
#### 4.3.4 Storyboard

As the game is multiplayer focused, the Storyboard that was developed is relatively short. It primarily covers the menu and UI systems, with an eagle eye view of the map and a rough sketch of the first-person VR View.

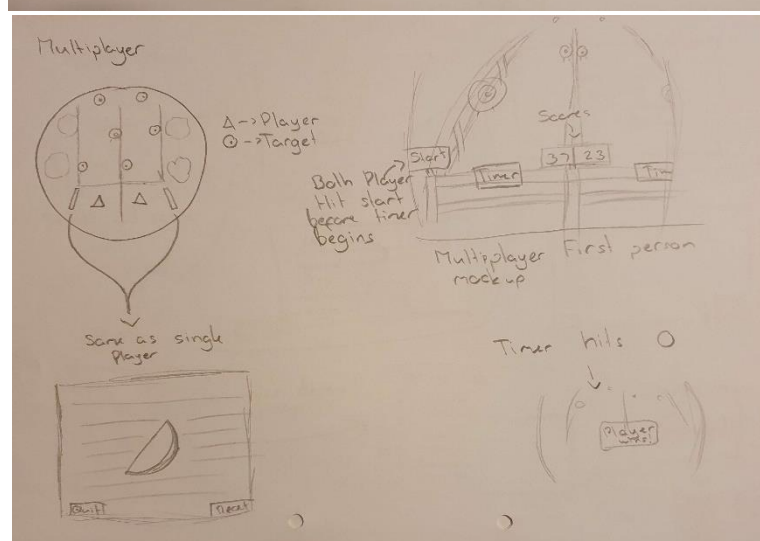
##### 33 Initial Storyboarding



This first page goes over how to navigate the menu systems, such as joining a multiplayer server or starting a single player game. The player can also navigate to an options menu to access accessibility features or change the display settings. An eagle eye view of the main menu is displayed as well, as this will take place in a very small scene.



The second page goes over the single player view. An eagle eye view of the map is displayed as well as a rough sketch of the first-person perspective in VR. A scoreboard is present to the right of the player, and a display is to the left with the bow used in game as well as a button for the options and quitting. Along this is a small tutorial option that will pop up when the scene is first loaded.



The third page goes over the multiplayer view. Another eagle eye view of the map is displayed, along with a rough sketch of the first-person view in VR. The board with the bow is on the left or right of the player, depending on what side they spawn in on. On the fence, a timer and start button are visible along with the score for each player.

#### 4.3.5 Level Design



As the game does not have any sort of level or progression system, each stage is designed as an enclosed space with a central firing range. Both the single and multiplayer stages will be designed with a linear archery range as the central piece, with targets placed randomly down the range. Each target will be worth a different number of points, depending on how difficult they are to hit with the arrows fired from the player. The game will have individual spaces the player will load into, but they do not act as levels in the traditional sense as they do not have a beginning or an end. They simply act as scenes with which the player can interact with the gameplay mechanics.

There are a total of three individual scenes the player will be in. The first is the main menu. As the game is in VR, the main menu is a physical space that the player can look around. The player is not able to move in this scene, but they can use the pointers protruding from their hands to interact with the menu in front of them and access the rest of the game.

The second is the Single Player archery range. The player will load into a new scene in front of a single archery range indicated by fences. The Range will have several targets and hitting the targets with arrows fired from a nearby bow will update a scoreboard visible next to the range. To the left is a taller fence wall where the bow will be mounted, along with a button to return to the main menu and one to open the options.

The final scene is the Multiplayer archery range. Much like the single player range, the player will load into the scene in front of an archery range, this one twice as large as the original. There will be two firing points separated by fences that the players will move towards, and each side of the range will have colour coded bows and targets. To the side of each player will be a menu much like the single player scene, where players can quit to the main menu or open the options.

#### 4.3.6 Environment

As the game is based entirely around archery, the environment will be that of a traditional firing range out in the forest. Each scene will be populated with foliage such as trees, flowers, and bushes, as well as small amounts of dirt paths and rocks. Additionally, there will be a small number of man-made objects as well such as fences, targets, signposts and in some instances tents and campfires.

The main menu will be a small camp that the player is positioned slightly away from. In front of them will be the main menu positioned on a sign built into the world. The camp will consist of multiple targets, a small tent and campfire, along with arrows sticking out of various surfaces. The player will be immediately surrounded by cliff faces (This will remain consistent throughout each level) with their camp nestled in a clearing, surrounded by trees and foliage.

The single player and multiplayer scenes will both look very similar. The main aspect of each will be the central archery range that the player will interact with. This will be built into the environment, with foliage growing over and around the fences surrounding the range. The targets down range will be either free standing or placed hanging from tree branches or fences. In the multiplayer version, fences will be separating each players individual firing range and the targets will be colour coded to match their bows. The environment will be decorated with foliage and signs, along with man-made structures, and allow a small amount of exploration for the player.



## 4.4 Conclusion

This chapter has discussed the steps taken to ensure that the application is visually appealing to any users that play it.

Player navigation (in the game world and the UI) was explained with both wireframe and paper prototypes, to solidify how navigation would work. Alongside these were multiple diagrams, visualising exactly how each mechanic in the game would lead to the next.

Level and environment design were also discussed, and the overall theme of the game was finalised. Focusing on nature, the colour scheme would be very natural but not too hard on the eyes. To this end, pastel colours were decided upon for what would normally be harsh colours, while the more muted colours could remain unchanged.

Finally, the UI design was also concluded, and would be built into the game world using signposts positioned around the various environments.

## 5 Implementation

### 5.1 Introduction

The implementation section will review the process of implementing the various features of the game over the development period.

The development environment will be discussed and explained first, giving an in-depth discussion about its functionality and reliability over any other options, especially when working in conjunction with the chosen game engine.

The Sprint section will discuss the specific activities done during the development period, split up into two-week sprints over the 4-month development cycle. Each sprint will focus on the main feature added during that time and will comment on any issues that occurred during development.

### 5.2 Technologies

The application has been developed using the following technologies:

- Normcore  
“Normcore is a networking plugin and hosting service developed by Normal” (NormalVR, 2021). Developed for the Unity Game engine, it allows developers to easily create and host multiplayer games, synchronising users within the online space and was developed with VR in mind.
- XR Interaction Toolkit  
“The XR Interaction Toolkit package is a high-level, component-based, interaction system for creating VR and AR experiences.” (Unity Technologies, 2023). This is a framework created and developed by the Unity company provided free of charge in every copy of Unity. It automatically adds necessary components into your Unity project, allowing for the development of VR applications.
- Unity Engine  
The Unity Engine is a free game engine developed by Unity Technologies. It’s an incredibly small yet powerful application with a plethora of both official and community created plugins and extensions. Its simplicity allows for any level of developer to pick it up and use it, and the extensive community provides near limitless educational content.

### 5.3 Development environment

The integrated development environment (IDE) being used for this project is Visual Studio (VS) developed by Microsoft. Within VS is Visual Studio Code (VSC), the primary coding application being used for this project. VSC includes a built-in command-line interface, extensive community support and a large variety of addons developed by members of both the community and Microsoft themselves. It includes a built-in debugger, intelligent code completion and Git.

VSC is the primary coding software that Unity comes pre-packaged with along with C# compliant additions to aid with creating scripts. Within Unity, when a new script was created it would automatically open in VSC along with a default Start and End statement to allow immediate work to begin. Additionally, it imported the default unity libraries so that developers don't need to worry about missing components.

Alongside VS, Github was used extensively to upload the project files online. Github desktop significantly simplified the process of updating the master branch on the web, and updates were pushed once major changes were made to the application after being tested thoroughly. Having an online repository meant that there was always a backup available and was an indisputably important safety net when the application stopped functioning at times.

## 5.4 Sprint 1

### 5.4.1 Goal

For the initial sprint, the goal was to further develop and finalise the tools and frameworks being used to develop the game.

Stated previously in the design document, multiple game engines and multiplayer frameworks were researched before ultimately landing on the Unity Engine for the game engine, and Normcore for the multiplayer framework. The VR framework chosen was the XR Interaction Toolkit as that comes pre-packaged with Unity and Normcore was developed with the XR Toolkit in mind.

Additionally, during this time paper prototypes of the game's environment and UI were drawn up to get a better idea of the end goal. A storyboard was also developed to show how users would navigate the game world properly in VR.

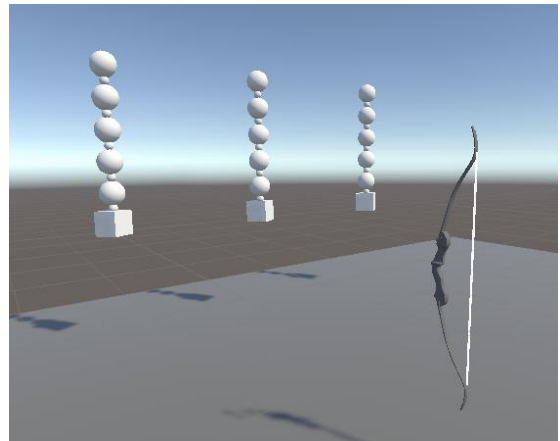
## 5.5 Sprint 2

### 5.5.1 Goal

Once the game engine and frameworks were finalised, I began work on an initial prototype to set the groundwork for future iterations. This prototype would not have any multiplayer functionality but would instead focus on having a working VR player controller, and a functional Bow and Arrow system.

### 5.5.2 Bow and Arrow Prototype

A prototype application was developed as a proof-of-concept. This prototype included a single scene with a VR player controller, an interactable bow and a small selection of targets the player could shoot at. During this cycle, multiplayer was yet to be implemented and the purpose was simply to set the groundwork for future iterations. This prototype used the XR Interaction Toolkit to create the working VR player controller and utilised the Bow and Arrow prefabs and scripts developed by Andrew, a Unity course creator, and Youtube Content Creator with several VR tutorials (Andrew, n.d.).



34 Initial Bow + Arrow Prototype

The Bow and Arrow package comes with a number of pre-defined scripts, and once imported into my scene I began implementing them properly following the guides that Andrew had provided users with.

The Notch Script is responsible for allowing the Bow and Arrow scripts to interact with each other. This script creates a `releaseThreshold` which determines the minimum distance the string needs to be pulled for an arrow to be released. This in turn calls the `PullMeasurer` script, which then calls the `ReleaseArrow` Function.

The `PullMeasurer` script calculates the distance that the arrow projectile will travel once the string is released. Using the 3 Transform points present on the String prefab, it limits how far the pullback animation goes, and calculates the force applied to the arrow object.

Once this information is calculated, the Arrow script calls the `Launch` function within itself. This enables the `SetLaunch` function, and the values calculated from the `PullMeasurer` script are sent to the `ApplyForce` function. This value is added to the arrow object as force, causing it to fly forward.

```
protected override void OnEnable()
{
    base.OnEnable();

    // Arrow is released once the puller is released
    PullMeasurer.selectExited.AddListener(ReleaseArrow);

    // Move the point where the arrow is attached
    PullMeasurer.Pulled.AddListener(MoveAttach);
}
```

35 Releasing Arrow + Resetting String

```
private float CalculatePull(Vector3 pullPosition)
{
    // Direction, and length
    Vector3 pullDirection = pullPosition - start.position;
    Vector3 targetDirection = end.position - start.position;

    // Figure out the pull direction
    float maxLength = targetDirection.magnitude;
    targetDirection.Normalize();

    // What's the actual distance?
    float pullValue = Vector3.Dot(pullDirection, targetDirection) / maxLength;
    pullValue = Mathf.Clamp(pullValue, 0.0f, 1.0f);

    return pullValue;
}
```

36 Calculating Trajectory of Arrow

## 5.6 Sprint 3

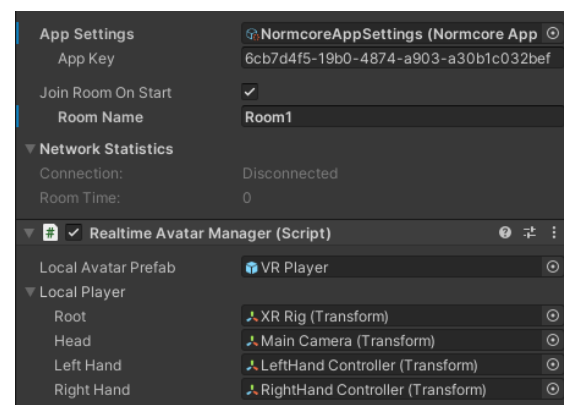
### 5.6.1 Goal

Following the successful initial prototype, I immediately began working on a multiplayer version of the basic scene I had created. This involved the installation of the Normcore framework and required sections of code to be edited to work properly with Normcores multiplayer system such as the Bow and Arrow scripts, as well as several objects (Normcore, n.d.).

### 5.6.2 Multiplayer Framework Implementation

Implementing Normcore into the project was incredibly straight-forward. Normcore is a premade Unity package that can be installed from Unity's asset manager (Normcore, n.d.). After installing Normcore and adding it to the scene hierarchy, I opened the prefab in the inspector and added the required transform components to the Realtime Avatar Script Manager. This ensures that the player avatar is updated in real-time on the multiplayer server.

For the room to be properly hosted, I went to Normcores website and created an application. This application handles all the server functionality required, and after its creation I was given an application key. This key allowed the unity project to connect to the server by inputting it into the App Key section in the inspector, and as a result the game was successfully made multiplayer. Upon loading the scene, the application connected to the Normcore server, and loaded the player into an online instance of the scene. Any other players with the same application would be visible.



37 Normcore Initialisation

### 5.6.3 Multiplayer Object Synchronisation

Once multiplayer was functional, I began work on synchronising objects within the scene. Normcore comes with several scripts that can be added onto objects to synchronise them online. Most notably, any object that could be interacted with or moved required the Realtime View, and Realtime Transform script. These two scripts sent updates to the server anytime an object was moved in the scene, displaying that movement to everyone connected to the room.

These scripts were added to the Bow and Arrow game objects and would later be added to the score and points systems.

Certain objects required more work, however. The arrow objects were originally refusing to be synchronised with the Realtime scripts on them, and it was discovered that synchronisation was failing due to them not being present in the scene until manually spawned in by a player. The solution to this was in the Quiver script. As the

Quiver script is responsible for spawning the arrows in, it needed to be modified. A new package was loaded called Normal.Realtime, and in the CreateArrow function,

"Realtime.Instantiate" replaced "Instantiate". This ensured that every arrow created

```
private Arrow CreateArrow(Transform orientation)
{
    // Create arrow, and get arrow component
    GameObject arrowObject = Realtime.Instantiate
    return arrowObject.GetComponent<Arrow>();
}
```

38 Fixed Arrow Code

would be instantiated and synchronised with Normcore, allowing players to see updated positions without error.

## 5.7 Sprint 4

### 5.7.1 Goal

After successfully implementing multiplayer and synchronising actions between multiple users, work began on creating a multiplayer scoring system. This involved the creation of a scoreboard and score sync model, a feature that Normcore created to upload data from the game to the server and ensure each player receives it, as well as two separate scoring scripts to give players points. An online video tutorial was utilised during this period by Mike McCready on Youtube (McCready, 2020).

### 5.7.2 Multiplayer Scoreboard

First, the model script was created. This is simply a C# script that Normcore compiles for you once you have put in the correct code and necessary properties.

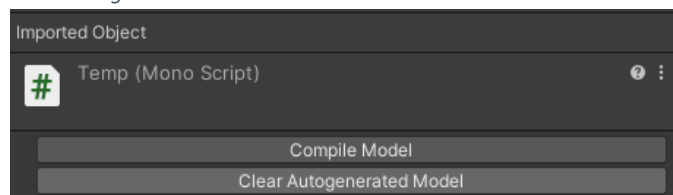
This code is first defined as a Realtime Model and as nothing needs to be inherited, the default `MonoBehaviour` section is removed. After, the class is completely cleared, and the necessary properties are added for compiling. As this is a parameter within the Realtime system, `RealtimeProperty` must be added. The 3 parameters are for the items ID, selecting whether it is reliable or not, and dictating whether the object can interact with or change parameters within the event system.

In the Unity inspector, the script now has the option to compile the model, meaning that Normcore will automatically generate all the necessary code needed to begin synchronising the values in the script between users.

Once the two models were created, work began on the Scoreboard script first. After importing the necessary Normcore and Unity libraries, the script is changed so that it inherits from the `RealtimeComponent`. The scoreboard model is referenced along with Normcores avatar manager to allow the script to access data stored on player avatars, as well as update the display text whenever a user joins or leaves the room. Additionally, this script registers every user that joins and gives them a Unique ID. When the scoreboard updates with a new user, they are displayed as Player 1, Player 2 etc. instead of repeatedly displaying the same name.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Normal.Realtime.Serialization;
5
6 [RealtimeModel]
7 public class Temp
8 {
9     [RealtimeProperty(1, true, true)]
10     private string _scoreBoardText;
11 }
```

40 Creating a Realtime Model



39 Compiling New Model

```
//Loops through each Avatar in the scene and grabs its score
private void SetScoreBoardText()
{
    //Applies unique ID to every user connected to a room
    int playerID = 0;
    _model.scoreBoardText = "";

    foreach (var item in _avatarManager.avatars)
    {
        //Updates scoreboard to say Player 1, 2 etc when new players join
        playerID = item.Key + 1;
        //Access the public score value on the model
        _model.scoreBoardText += "Player " + playerID + ": " + _avatarManager
    }
}
```

41 Updating Scoreboard when new Player Joins the Room

The scoreboard script is responsible as well for constantly updating the scores that are on display, and it does so by checking the players internally stored score whenever an update is made to the text object attached to the avatars. This loop also ensures that if a player leaves, it treats it as an “avatar destroyed” event, removing it from the scoreboard. This scoreboard script was then added to the Scoreboard UI element within the scene and edited appropriately.

### 5.7.3 Multiplayer Score Syncing

The `ScoreSync` model and script were both written in parallel with the Scoreboard model and script. This script first references the `scoreSync Model`, and within its `Awake` function it immediately gets the reference to the Score Text Game Object. Following this, the script accesses the `scoreSync Model` in order to push the client-side score values to the Normcore server. This allows every user to obtain all players scores and display the correct value on the Scoreboard.

Following this, the `scoreSync` script waits for any number of events to trigger. Whenever the score is updated, it pushes the new value to the client-side scoreboard and updates the display appropriately. This value is then pushed to the `ScoreSync Model`, which uploads the value to Normcores servers. This value is what the Scoreboard Model and Script pull whenever an update to the player score is detected.

All this information is pulled from the `playerScore` value stored internally in the avatars.

### 5.7.4 Multiplayer Point Acclamation

Finally, with the scoreboard and score sync system functioning properly a script was developed to provide the player with points once targets were successfully hit. This simple script referenced both bows available in the scene, as well as the Scoreboard. It also provided the bows with unique owner ID's and allows the target scores to be modified easily.

Once the scene loads, the script applied to the targets wait for a collider trigger to fire. When this occurs, it checks that the object that triggered the event is tagged “arrow” and that the bow is currently being held by a player. If all the prerequisites are met, the script gets the owner ID by requesting the owner of the bow, and cross references that with the ID for the score value, updating the displayed score that matches the holder of the bows ID. It then resets the bows ID value and waits for another trigger event to fire.

```
private ScoreSyncModel model
{
    set
    {
        if (_model != null)
        {
            // Unregister from events
            _model.playerScoreDidChange -= ScoreDidChange;
        }
        // store the model
        _model = value;

        if (_model != null)
        {
            // Update the score the match the new value
            UpdateDisplayScore();

            // Register for events so we'll know if the score changes later
            _model.playerScoreDidChange += ScoreDidChange;
        }
    }
}
```

#### 42 Code Pulling Score from User

```
//Updates score when change is detected
private void ScoreDidChange(ScoreSyncModel model, int value)
{
    UpdateDisplayScore();
}

//Updates score by pulling the data from the model
private void UpdateDisplayScore()
{
    _scoreText.text = "Score: " + _model.playerScore;
}

//Gets player score attached to Avatars in game
public int GetScore()
{
    return _model.playerScore;
}
```

#### 43 Code updating Scoreboard with Users Scores

```
private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag("arrow"))
    {
        if (Bow.isSelected)
        {
            ownerID = GameObject.Find("Bow").GetComponent<BowRequest>().ownership;
            _scoreBoard.SetScoreForPlayer(ownerID, score);
            GameObject.Find("Bow").GetComponent<BowRequest>().ownership = -1;
        }
    }
}
```

#### 44 Retrieving Bow Owner ID when Points Scored

This script was duplicated and modified to work with the second bow in the scene, as both act independently of each other and so needed separate scoring scripts.

The bows Owner ID was obtained from the `BowRequest` script added to the Bow object in game. This script is responsible for updating the position of the Bow for each connected client and provides the ID of the user who is holding it through the `RequestOwnership` method.

During this time, I attempted to create and synchronise a timer for the multiplayer environment, in order to properly incentivise competitive gameplay. Using a step-by-step guide on Normcores documentation, a timer was successfully created and synced between users. Unfortunately, it would only count from zero, and despite numerous attempts, I was unable to force it to stop after a certain length of time had passed. Without the ability to stop the timer, there would be no way to time when the score system should disable itself and tally up the final scores while keeping everything synchronised between players which ultimately resulted in the online timer being removed.

## 5.8 Sprint 5

### 5.8.1 Goal

With the online scoring system functional, work began on the single player aspect of the game. Many scripts and objects had to be modified in order to work properly in a multiplayer environment, and so creating the single player scene required more effort than initially thought.

### 5.8.2 Single Player Bow and Arrow

First, the bow and arrow scripts and objects had to be duplicated and reverted to their original code. As Normcore requires constant connection, the objects attempting to connect to the server when the scene was not an online instance would refuse to transform or update properly.

As a result, the modified scripts were copied and renamed to single player variations along with the objects they interacted with. Some changes were minor, only requiring one or two new lines of code to function normally in an offline environment.

### 5.8.3 Single Player Score System

Secondly, an entirely new scoring system was created to act independently of the multiplayer system. While functionally identical, just like the previous objects the multiplayer system required constant connection and the scoreboard would be completely blank as the single player avatars do not have an internally stored score value.

The new score system was created by adding a new script to the single player arrows called `ArrowController`. This script simply uses the same `Collider Trigger` as the multiplayer `scoreSync` script to detect when an arrow hits a target, and the score is updated appropriately depending on the value provided in Unity's Inspector. The updated values are sent to the `ScoreController` script.

This script is attached to the scene Event System and contains the text object that the scoreboard

```
private void OnTriggerEnter(Collider other)
{
    if(other.gameObject.CompareTag("targetEasy"))
    {
        ScoreController.score += scoreEasy;
    }

    if(other.gameObject.CompareTag("targetMedium"))
    {
        ScoreController.score += scoreMedium;
    }

    if(other.gameObject.CompareTag("targetHard"))
    {
        ScoreController.score += scoreHard;
    }
}
```

45 Score System Code



references when displaying points. The value updates whenever the player scores by hitting the targets.

## 5.9 Sprint 6

### 5.9.1 Goal

After successfully developing both a single and multiplayer experience, I began work on the game's main menu screen, and implemented methods to load between the various scenes in game. This included the development of additional multiplayer rooms, and the games starting scene. Multiplayer Scene loading tutorial was provided by The Weasly Wizard on Youtube (Wizard, 2022).

### 5.9.2 Creation of the Main Menu

Within Unity, I created a new scene with the plan of turning it into the games start screen. I created a new UI Canvas and placed it in the game world, adding buttons for both Single and Multiplayer instances. For the buttons to function properly, two new scripts were created called `SinglePlayerLoader` and `AsyncSceneLoader` respectively.

The Single Player Loader script only had two lines of code within the `LoadScene` method. The first executed the command to force Unity to load the given scene by inputting its exact name. In this, case the single player scene. The second line ensured that whenever the scene is loaded, the score displayed is set to zero. This was the fix to an issue that would occur when players scored points in the multiplayer rooms, and then loaded into the single player instance only for the score to display what they had earned in the multiplayer instances.

```
public void LoadScene()
{
    SceneManager.LoadScene("SinglePlayerScene");
    ScoreController.score = 0;
}
```

*46 Single Player Scene Loading + Score Reset*

The multiplayer scene loader was created using Normcores Async loader. Async loading means that the scene is being prepared in the background when the method is called and will only load once all the necessary components are loaded and ready to be displayed.

The script first requires three references. Realtime, to allow users to connect and disconnect from the room. A reference for the name of the room, and lastly the index number of the scene we wish to load. A method is then called to load the scene, and a coroutine is started and called to load players into the new room.

Within the coroutine, Realtime disconnects the player from any room they are currently connected to, and then loads the scene using Async based on the scene Index number in the inspector.

The while statement within the code is checking when the scene will finish loading, and once the loading is finished the user will load into the selected room.

```
IEnumerator LoadSceneAsync()
{
    realtime.Disconnect();
    realtime = null;

    var loadAsync = SceneManager.LoadSceneAsync(sceneIndex);

    while (!loadAsync.isDone) yield return null;

    realtime = FindObjectOfType<Realtime>();
    realtime.Connect(roomName);

    isLoading = false;
}
```

*47 Code Demonstrating Multiplayer Scene Loading*

Once finished, a button was added to each scene so the player could return to the main menu at will. Additionally, the multiplayer room was duplicated, and three more instances were created giving the player four options when selecting the multiplayer mode.

## 5.10 Sprint 7

### 5.10.1 Goal

With the game functioning properly, I swapped my focus towards level design. At the time, I had all the core mechanics working, and so the next step was to design the environment that the player will be standing in.

### 5.10.2 Initial Level/Environment Design

My initial work was on the multiplayer scene. Using assets I obtained from the Unity store, I quickly settled on a low poly style for the game. I used Tree, Fence and Rock assets from the user BrokenVector (Vector, n.d.) on the asset store to design a simple archery range with two ranges to fire down for the multiplayer rooms. The targets were acquired from sketch fab from the user Bram Van Hoof (Hoof, n.d.) and matched the overall aesthetic perfectly.



48 Multiplayer Environment Prototype

Within the scene, I recoloured both the Targets and the Bows respective to the side that they're on. This was to act as a visual indicator for the player to know which targets to shoot at when competing against others.

Once confident with the overall style and colours, the focus changed from the multiplayer scene to the start screen. Using the assets previously mentioned, a secluded camp was designed to act as the title screen for the game.



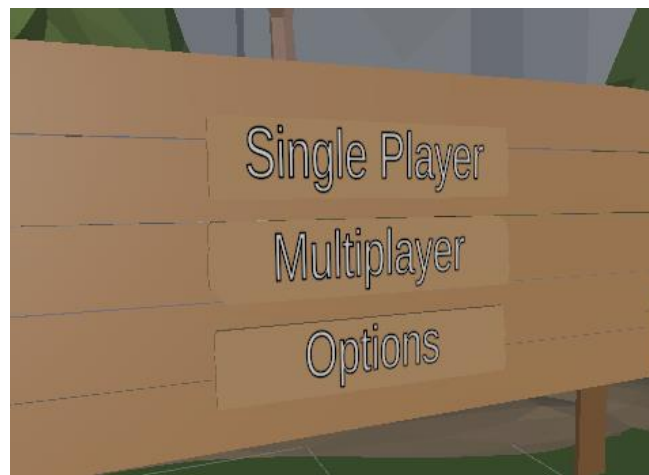
49 Main Menu Environment

The scene was created using the previously mentioned assets from the user Broken Vector, along with his low poly cliff package as well. The low-poly tent was obtained from the user simonaskLDE

on Sketchfab (simonaskLDE, 2021), and the campfire model was created by using already existing wood and stone assets and placing them in the scene manually. Finally, the Wooden log visible behind the campfire was obtained from the user JustCreate (JustCreate, n.d.) on the asset store, from their Low-Poly Simple Nature Pack.

### 5.10.3 Enhanced UI

After the scene was complete, I worked on the UI as I wanted it to blend in with the game and not stand out. To do this, I first imported a collection of signs that could use in the game. These signs were created by the user iPoly2D on CGTrader (iPoly3D, 2020). After adding a large signpost to the scene, using blender I modified an existing signpost and created smaller planks to attach to it. The menu UI was then placed on top of these planks and spaced accordingly, one for each menu the player could navigate to.



50 Main Menu UI

The buttons were then edited to hide or display new objects when certain buttons were pressed using the onClick function attached to them. For example, when multiplayer is selected, the fence updates and hides the original menu options, replacing them with the layout displayed below.



51 Various Menus

An Options menu was also added at this point and includes the ability for the player to change their turning mode from snap turning to smooth turning, whichever they prefer. Additionally, an outline was added to each menu button, so when the player hovers over it in game, a black outline appears for further clarity when selecting menu options. This required adding Mesh colliders and Simple Interactors to each button object, and using a script obtained from the asset store. I then changed

the Raycast type from a line to a sphere, so that they menu object and the button were both being hit when the player hovered over them.

## 5.11 Sprint 8

### 5.11.1 Goal

With the completion of the title screen and implementation of the new UI system, I moved onto the Multiplayer and Single Player levels, purely focusing on level design.

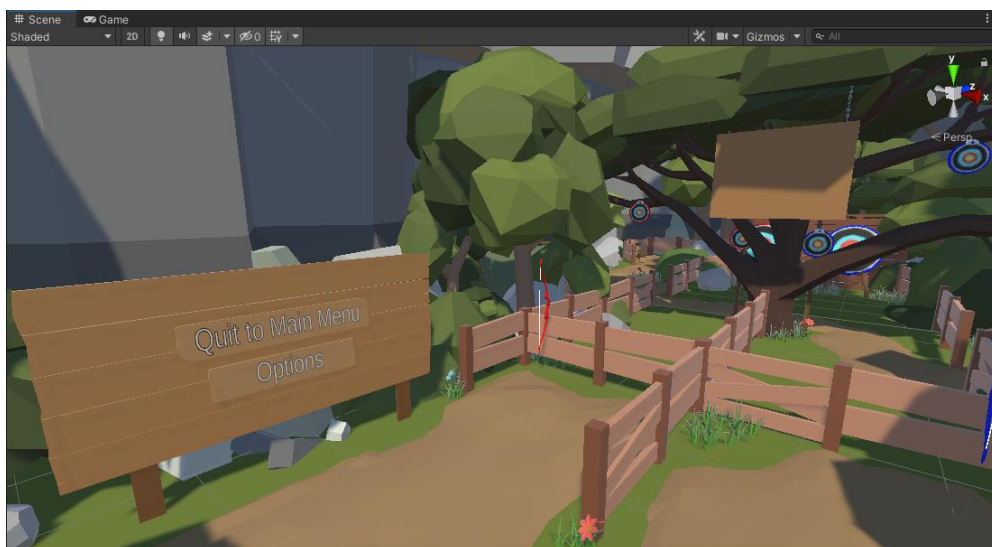
### 5.11.2 Finalising the Multiplayer Environment

I first began work on the Multiplayer scene, using the early design as a basis for the rest of the scene. Much like the title screen, the scene was created using the assets previously discussed (Trees, Rocks, Fences etc.) built into an enclosed forested area surrounded by sheer cliffs.



52 Aerial view of Multiplayer Scene

A few signs were placed around the scene in preparation for the UI elements they would hold, such as the fence hanging above the trees and the sign to the left of the play area, where users could change the options or leave the room.



53 Closer view of Shooting area and UI Signs

The bows and targets were colour coordinated so that players have a clear idea of which target they are meant to shoot, and once the scenery was in place mesh colliders were added to every objects so that collision would work properly with everything around the scene.

The Multiplayer scene is decorated with natural flora as well as manmade objects and includes a small camp area and an alternate training area with target dummies. Effort was made to ensure the arena would be fun to explore, and players can walk around most of the scene. Once finalised, this multiplayer scene was copied and applied to all 4 rooms in the application, modifying the room settings to ensure they worked properly.

#### 5.11.3 Creation and Completion of the Single Player Environment

Finally, the single player scene was decorated appropriately. The arena is smaller, but still allows for some exploration for the players. UI Signs are placed around the scene for score visibility, and an option to start a timer is available next to the firing range for players trying to achieve a high score.



54 UI Signs visible in Single Player

Once the decorations were finished on both the single and multiplayer scenes, I finally added invisible walls around each one to stop the player from wondering out of bounds as a final layer of precaution.

Finally, I added a short tutorial to the single player section of the game. It teaches the basic movement and firing mechanics, as well as the way to navigate the UI in game.



## 5.12 Conclusion

This chapter discussed the technologies utilised for the creation of this project along with the development environment used. Alongside this, each Sprint was outlined and explained to give a clear depiction of the actions taken during each sprint cycle.

While each sprint is separated into its own section, there was a lot of overlap between them as new mechanics introduced had to be properly designed around old scripts and objects already fully functional.

## 6 Testing

### 6.1 Introduction

This chapter describes the testing that was undertaken for the application. The two types of testing utilised for this were Functional Testing and User Testing. Functional Testing involves performing specific actions to determine what parts of the game were functioning properly, while User Testing involves asking a game tester with no prior experience with the game to perform simple tasks possible in the application to record the results and gather feedback.

### 6.2 Functional Testing

This section describes the functional tests which were carried out on the app. These functional tests can be categorised as:

- World Navigation
- Menu Navigation
- Gameplay
- Multiplayer

This portion of testing was not performed by a randomly selected tester, but the developer of the project.

#### 6.2.1 World Navigation

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	<b>Move Character Around the Scene</b>	Moved the right thumb stick.	Character/Camera moves around the scene.	The player successfully moved around the scene.	Successful, player controller moves as anticipated.
2	<b>Rotate the Camera</b>	Moved the left thumb stick.	Camera rotates left or right in a snap turning motion.	Camera successfully rotates.	Successful, camera moves as anticipated.
3	<b>Navigate to nearby camp</b>	Use both thumb stick to move and turn towards the nearby camp.	Player navigates smoothly towards camp on their right.	Successfully used movement and turning at the same time to navigate.	Successful, moving and turning work together.

## 6.2.2 Menu Navigation

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	<b>Navigate between menus</b>	Open options menu, go back, open multiplayer menu	Navigates between menus	Player successfully navigates between menus	Successful, menu navigation functioning as intended
2	<b>Change turning mode from snap to smooth</b>	Open options menu, select smooth turning, test.	Smooth turning enables and disables snap turning.	Smooth turning enabled successfully.	Successful, snap turning disabled and smooth turning enabled.
3	<b>Load into single player</b>	Select single player from main menu	Single player scene is loaded.	Single player scene is loaded.	Successful, player loads into single player scene
4	<b>Load into multiplayer room</b>	Select multiplayer from main menu, click any room.	Multiplayer room is connected to successfully.	Multiplayer room is connected to successfully.	Successful, player connects to multiplayer room.
5	<b>Return to main menu</b>	Move to menu in game, select return to main menu, confirm.	Main menu scene is loaded.	Main menu scene is loaded.	Successful, player returns to main menu.



### 6.2.3 Gameplay

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	<b>Grab and Release the Bow</b>	Moved to bow, used trigger to grab bow, released trigger to drop bow.	Bow snaps to players hand, rotates and moves alongside player.	Bow moves and rotates exactly as intended.	Successful, bow functions as intended.
2	<b>Grab, pull back and release bowstring</b>	Grab bowstring while holding bow, pull back and release.	Bowstring flexes in response to player input.	Bow string is pulled back and released successfully.	Successful, bowstring pulls back, and resets as intended.
3	<b>Grab arrow from Quiver</b>	Reach over shoulder and grab arrow from quiver.	Arrow is spawned in players' hand.	Arrow is spawned into player hand after a one/two attempts.	Partial Success, arrow is spawned but placement of quiver makes it inconsistent.
4	<b>Notch arrow to string</b>	Grab arrow from quiver and release trigger next to string.	Arrow attaches itself to bowstring.	Arrow attaches itself to Quiver, occasionally falls to ground.	Partial Success, arrow attaches but can fail rarely.
5	<b>Fire an arrow</b>	Pull back string with attached arrow and release it.	Arrow flies from bowstring to pointed direction.	Arrow successfully launched from bow towards target.	Successful, arrow detached itself and launched in an arc.
6	<b>Start timer and score points</b>	Start the nearby timer and hit any targets on the range to score points.	Counter begins and score goes up once target is hit.	Timer begins and score successfully increases within the timer limit.	Successful, timer and score system working as intended.

#### 6.2.4 Multiplayer

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	<b>Connect to multiplayer room</b>	Join room from main menu.	Load into room with another player.	Load into room with another player.	Successful, player can join rooms with other users
2	<b>Score points in Multiplayer</b>	Score points by firing at the targets.	Player's score goes up as they hit targets.	Player's score goes up as they hit targets.	Successful, players can score points.
3	<b>Fire at other players targets</b>	Fire at other players targets.	Neither player's scores change.	Other players score increases.	Partial success, player firing the bow earns no points.
4	<b>Disconnect from multiplayer Room</b>	Use in game menu to return to title screen.	Player returns to main menu.	Player returned to main menu.	Successful, player disconnected from room.

#### 6.2.5 Discussion of Functional Testing Results

The results of the functional testing show that most core mechanics of the game are working as intended. In some cases, such as the arrow not properly attaching to the notch, changes are unable to be made as that is not an issue with the socket system, but rather the location of the notch relative to the location of the attach point of the arrow. Additionally, the multiplayer bug where player one can score points for player two is not intended, however it doesn't give any advantage to the player shooting at the targets, so it seems unnecessary to remove at this point of development.

Despite the small number of issues, it appears that the application is running as intended across all testing criteria.

### 6.3 User Testing

For user testing, two people were given various tasks to perform while playing the game. These users were all owners of an Oculus Quest 2 headset and so remote testing was conducted.

The tasks given to the users were as follows:

- Play the single player portion of the game and proceed with the tutorial.
- Score as many points as possible in the timed game mode.
- Navigate the single player environment.
- Connect to a multiplayer room and play with another user.
- Navigate the multiplayer environment.

Each user was tested on a different day to correspond with their schedules, and all were done remotely using online voice and video chat to gather results and feedback more easily. For the multiplayer portion of the tests, I filled in as the second user in the room to ensure proper functionality.

Both users had extensive VR experience and were avid gamers, so the controls were incredibly easy for them to learn and use fluidly. However, I received several comments and remarks on various mechanics or functions of the game. The most common were the following:

- The options menu didn't have a lot of choice, seeming to only change the players turning type.
- In the single player environment, the rays coming from the players hands would sometimes lock onto a random point of the map and would need to be moved around to become unstuck.
- The quiver could sometimes be annoying to grab arrows from.
- The bows floating in place after releasing them was distracting from the rest of the environment.
- Lack of a timer in multiplayer was disappointing.
- The voice chat was fun, but having no way to turn it off was annoying.

Despite the flaws, both users extensively enjoyed the time spent playing the archery game and gave a few recommendations for things that could be added, such as:

- More varied maps.
- Background Music.
- Built in quit button.
- Adding more options to the options menu, such as volume sliders or a mute function.
- Implementing a high score system.
- Adding a timed game mode to the multiplayer aspect of the game.
- Character customisation for multiplayer.

Using these responses, I worked to implement a few things that were suggested. In the limited amount of time left I managed to add background music and ambience to every scene provided by David Fesliyan, along with a functioning volume slider on the options menu using a tutorial created by Hooson on Youtube. Additionally, a Quit button was added to the main menu, along with another menu confirming whether players do want to quit, or return back to the main menu.

## 6.4 Conclusion

This chapter explained the two main methods of testing that were used to test various aspects of the archery game. The functional testing portion was much more in depth and thorough compared to the user testing, as the various intricacies of the application were being evaluated. This allowed for much more tests to be carried out in a shorter period, but lacked the randomness element that user testing can provide.

The user testing portion displays how users are interacting with the game from a player's point of view. Their actions and issues they brought to light are ones that wouldn't be seen during normal in-house testing, and as such they provided valuable information. Both forms of testing were invaluable for the completion of the application and allowed for micro adjustments to the final product.

## 7 Project Management

### 7.1 Introduction

This chapter describes how the project was managed. It shows the phases of the project, going from the project idea through the requirements gathering, the specification for the project, the design, implementation, and testing phases for the project. It also discusses Trello and as tools which assist in project management.

### 7.2 Project Phases

#### 7.2.1 Proposal

Initially, the project was going to be an Augmented Reality application developed for data visualisation. The project would have allowed users to input custom data from spreadsheets and display them in the environment through cameras. This idea was eventually discarded however for a more familiar subject: game development. It was decided that the game would be a VR Multiplayer Archery Game, expanding on work that had been done earlier in the year during our VR Game Development Module and creating a multiplayer experience.

#### 7.2.2 Requirements

The requirements section required thorough research into VR technologies, including both hardware such as various headsets or controllers, and software such as VR games or applications. User interface design was a key focal point during this and the games chosen for discussion all designed their UI in unique ways. Non-VR games were included in this research, as UI from regular applications could also influence the product.

Additionally, a survey was created and sent to various users, and several interviews were conducted to pinpoint key components for video games. The results of which would influence the final product and help it succeed.

#### 7.2.3 Design

A storyboard was created and drawn out on paper as the initial vision of the game. This included menus and navigation, map layouts and even a first-person sketch of the environment in VR. The UI was expanded upon further through the creation of a wireframe prototype in Figma that was fully functional, and the overall theme and colour scheme of the game was finalized. Research went into the functionality of various frameworks and diagrams were created to understand the application architecture.

### 7.2.4 Implementation

The application was developed inside of the game engine Unity, and all coding was done in the program Visual Studio Code as that is Unity's default editing software. A prototype of the game was first developed to ensure the bow and arrow code was functioning properly after following a tutorial online. Once completed, development on multiplayer was the primary focus. The Normcore documentation was referenced heavily along with several video guides to properly sync objects together and create a working multiplayer instance.

Many functions originally planned for the application had to be removed as development continued due to time constraints and general lack of knowledge on various areas of networking. Functionality of the game was not sacrificed however, and more time was spent ensuring what was in the game worked near perfectly.

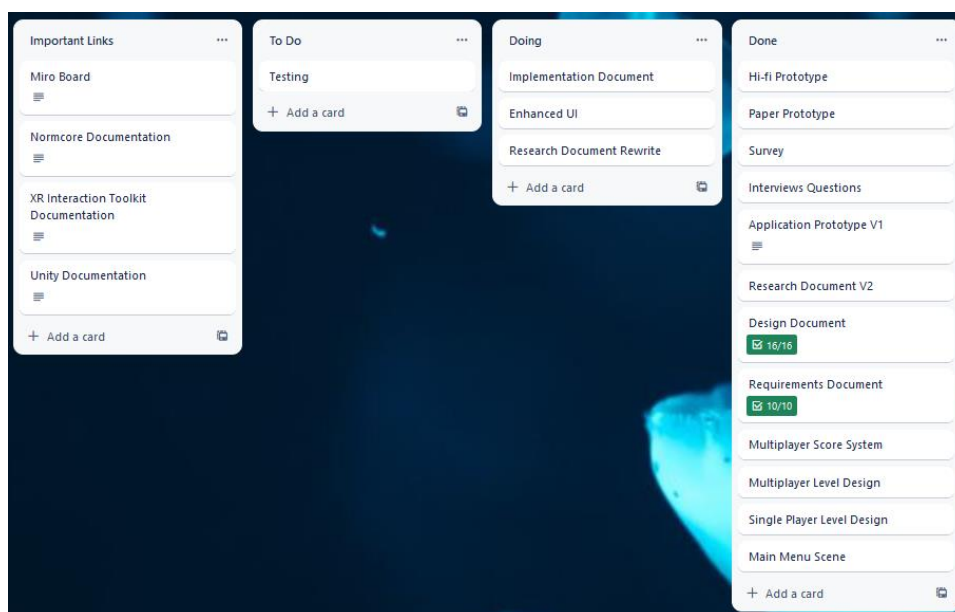
### 7.2.5 Testing

While testing the game, both User and Functional testing were used. Functional testing allowed for more precise and specific tests to be done to ensure each system worked without fail, and User testing allowed for feedback and input from users without any bias. The user testing also included receiving suggestions for what could be added to the application to improve the overall experience.

## 7.3 Project Management Tools

### 7.3.1 Trello

Trello is an online project management service developed by currently by Trello Enterprise. It is an advanced list making application, which allows the user to create various topics or categories and add tasks to them individually. These tasks can range from extensive checklists to single stage assignments and can be individually moved to different categories when completed or in progress.



55 Projects Trello Board

Above is a screenshot of the Trello board I used during development. As time progressed, I would periodically update the board and move components based on what was high priority at the time. This provided clarity and clear instruction on what components required work, and what could be set aside for the time being.

Trello aided in the development of the project massively and was a very powerful tool.

### 7.3.2 GitHub

GitHub is a cloud-based service for developers and programmers to store and manage their code remotely (Kinsta Inc, 2022). It allows developers to create 'branches' of their code to experiment or make changes without affecting the source code. This flexibility and ease of use draws many users, and the service is provided completely free of charge with optional subscription services.

Anyone who wishes to use GitHub can create a repository on their website after creating an account. A repository is where all files from a project will be uploaded to, and the initial upload is what is known as the Master branch. This master branch can be updated by uploading locally edited files to the repository either through the Git console, the GitHub Desktop app or on GitHub's website directly. GitHub also allows for a process called 'Branching'. Branching is when the user duplicates the source code and can safely make changes to it without affecting the original. These edits can then be merged to the master branch once the code works correctly.

For my project, GitHub was used extensively and frequently during development. Each time changes were made to the application and tested properly, an update to the Repository was made using the GitHub Desktop App. For the initial commit I first had to find a working gitignore file specifically for Unity applications which was easily done. Along with the projects source code, the projects thesis and APK were also uploaded to GitHub.

## 7.4 Reflection

### 7.4.1 Your views on the project

Overall, I am very pleased with how the project went. Development of such a large application alone was quite daunting at first, but once work properly began following the preparation I felt more prepared than I had at first. Working on this project taught me a lot about time management and prioritisation, and I dealt with burnout halfway through which ended up losing me a lot of time. Despite these setbacks however, I still managed to deliver a finished application that was enjoyable to work on and is enjoyable to play.

I feel as if my skill with using Unity and C# have improved dramatically, and my new knowledge on Networking has left me wanting to develop more multi-user experiences.

### 7.4.2 Completing a large software development project

The biggest thing I learned from this project overall was management. This includes time management, prioritisation, and even personal time. In the early stages of development, too much

time was spent trying to fix small issues that could have been pushed aside and tackled at a later stage. This in turn led to long hours of work without taking breaks, and as a direct result I fell behind.

Working long hours without end eventually resulted in a huge period where virtually no work was done, and I had realised upon returning to work that it was necessary to only work on what was important, and to take time away from the project. While it was a major part of my life, I let it overwhelm me and the final product suffered due to this.

#### 7.4.3 Working with a supervisor

Weekly meetings with my supervisor were a very helpful inclusion to the project. My supervisor would assist with assigning priority goals and ensuring that even when fallen behind I still had direction. When in person meetings weren't possible, they were conducted on Teams.

Having a supervisor often challenged me in ways I had not expected. Often, I would talk about ideas or would relay information about the project and would be met with questions and observations that I would never have thought of. It allowed my project to evolve uniquely throughout development and was an invaluable asset.

#### 7.4.4 Technical skills

Various skills have vastly improved during this project.

My knowledge of game development has grown tremendously, and developing a project for VR has not only introduced me to VR development, but multiplayer game development and networking also. I had previous experience working with Unity before, having spent time modifying existing game code or applications, but the project allowed me to build something unique and experience brand new technologies I had never touched. VR Development is challenging by itself, but including Multiplayer led to challenges I was not prepared for and learned much from as a result.

My competency with the Unity Game Engine and its primary scripting language C# has also improved as a direct result. Various tutorials were followed to aid development, but as time progressed, I relied on them far less.

#### 7.4.5 Further competencies and skills

While developing the application, I found myself spending a lot of time on world design and was proud of how each scene looks in the finished game. I believe this project has allowed my skills as a level designer to improve substantially and would feel inclined to pursue more projects that allowed for this level of expression.

## 7.5 Conclusion

This chapter clarifies how each section of the thesis was executed and includes my own personal thoughts on various aspects of the project.

Overall, I am pleased with how I managed the project. While there were areas that could have been improved upon, I do not believe that the product was hindered by the removal of features. A project of this scale has many aspects to it that require attention, and in the future, I will be better prepared when allocating time and energy. I believe my own skills have improved significantly, and will continue to apply them to projects in the future.



## 8 Conclusion

The overall aim of the project was to develop a Multiplayer VR Archery Game for the Oculus Quest 2 headset. The project was intended to be a competitive multiplayer experience, where players could compete against each other in one versus one scenarios. It would be simple to learn but with a high skill ceiling, and players could improve over time with repeated play.

The game was developed on the Unity Game Engine utilizing the XR Interaction Toolkit for VR functionality, and Normcore for managing the networking and multiplayer. Unity is a user-friendly game engine with a robust number of features and functionality to make game development significantly easier. The built in XR Interaction Toolkit made developing a VR project very straightforward, and Normcore implementation was intuitive due to having been developed with VR games in mind.

Prior to development, research was done on various forms of digital realities and their effect on modern game development. Extensive work was done exploring accessibility options for games, and both surveys and interviews were conducted to understand what players are looking for when they play a game.

Additionally, care was taken to ensure the game ran smoothly on the chosen platform (Oculus Quest 2). As such, a low poly art style and smaller maps were developed so that the game would remain responsive and perform optimally.

Many skills were learned during development, and many more were enhanced in various ways. Storyboarding, wireframing, networking, level design and multiplayer development were brand new skills learned, while skills in Unity, C# and game design were thoroughly improved.

The application can be improved in many ways. If further development was planned, effort would be put into making the multiplayer aspect feel more fully developed. Cosmetics and unique arrow trails could be added, alongside an online leaderboard. More environments could be developed to enhance gameplay, and the options menu could add more accessibility options such as teleportation movement or a vignette to aid with motion sickness.

## 9 References

- Altexsoft. (2021, July 23). *Functional and Non-Functional Requirements: Specification and Types*. Retrieved from Altexsoft: <https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/>
- Andrew. (n.d.). *Bow and Arrow for Unity XR*. Retrieved from Youtube: <https://www.youtube.com/watch?v=H0xTz4JtWil&list=LL&index=1&t=1s>
- Arc Speech. (n.d.). *Physical Disability*. Retrieved from Arc Speech: <https://arcspeech.ie/physical-disability/>
- Baranyi, P., Csapó, Á., Budai, T., & Wersényi, G. (2021). *Introducing the Concept of Internet of Digital Reality - Part I*. Acta Polytechnica Hungarica.
- Blacknight Interactive. (2020, May 19). *Golf With Your Friends*. Retrieved from Steam: [https://store.steampowered.com/app/431240/Golf\\_With\\_Your\\_Friends/](https://store.steampowered.com/app/431240/Golf_With_Your_Friends/)
- Burdea, G. C., & Coiffet, P. (2003). *Virtual Reality Technology*. John Wiley & Sons.
- Fesliyan, D. (2020, February 22). *"Relaxing Green Nature" by David Fesliyan*. Retrieved from Fesliyan Studios: <https://www.fesliyanstudios.com/royalty-free-music/download/relaxing-green-nature/548>
- Hoof, B. v. (n.d.). *bow\_target*. Retrieved from Sketchfab: <https://sketchfab.com/3d-models/bow-target-0c60e4afc8244703981435da8db5cfb6>
- Hooson. (2021, April 5). *How To Make A Volume Slider In 4 Minutes - Easy Unity Tutorial*. Retrieved from Youtube: <https://www.youtube.com/watch?v=yWCHaTwVblk>
- iPoly3D. (2020, June 23). *Lowpoly Wood Signs*. Retrieved from CGTrader: <https://www.cgtrader.com/free-3d-models/furniture/outdoor-furniture/lowpoly-wooden-signs>
- JustCreate. (n.d.). *JustCreate*. Retrieved from Unity Asset Store: <https://assetstore.unity.com/publishers/44390>
- Katharina Finger & Elvedin Mesic. (2019). *Digital Reality Explained*. Retrieved from Deloitte: <https://www2.deloitte.com/ch/en/pages/innovation/articles/digital-reality-explained.html>
- Kinsta Inc. (2022, December 13). *What is Github?* Retrieved from Kinsta: <https://kinsta.com/knowledgebase/what-is-github/>
- McCready, M. (2020, March 30). Retrieved from Youtube: <https://youtu.be/AH9BOhGwHbU>
- Mileva, G. (2022, February 09). *7 Benefits of AR and VR for People With Disability*. Retrieved from AR Post: <https://arpost.co/2022/02/09/7-benefits-ar-vr-for-people-with-disability/>
- NormalVR. (2021, September 3). *What is Normcore?* Retrieved from Normcore: <https://normcore.io/documentation/essentials/what-is-normcore.html#why-should-i-use-normcore>
- Normcore. (n.d.). *Getting Started with Normcore*. Retrieved from Normcore: <https://normcore.io/documentation/essentials/getting-started.html>

- Seyed Hassan, K. S., Mashita, S., Mohamad, A., & Rostam, Y. (2012). Accessibility for Disabled in Public Transportation Terminal. In *Procedia - Social and Behavioral Sciences* (pp. Volume 35, Pages 89 - 96).
- simonaskLDE. (2021, November 30). *Low-poly Survival Tent*. Retrieved from Sketchfab: <https://sketchfab.com/3d-models/low-poly-survival-tent-f3f5283fa07a47e798017bc3c055fe88>
- Unity Technologies. (2023, February 10). *XR Interaction Toolkit*. Retrieved from Unity3D Documentaiton: <https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.3/manual/index.html>
- Unity Technologies. (n.d.). *Unity Documentation*. Retrieved from Unity Documentation: <https://docs.unity.com/>
- Valve. (2020, March 23). *Half-Life Alyx*. Retrieved from Steam: [https://store.steampowered.com/app/546560/HalfLife\\_Alyx/](https://store.steampowered.com/app/546560/HalfLife_Alyx/)
- Vector, B. (n.d.). Retrieved from Unity Asset Store: <https://assetstore.unity.com/publishers/12124>
- VRChat Inc. (2017, 1 Feb). *VRChat*. Retrieved from Steam: <https://store.steampowered.com/app/438100/VRChat/>
- Wizard, T. W. (2022, November 26). *Part 3: Normcore Scene Loading & Sessions Management*. Retrieved from Youtube: [https://youtu.be/AGMKZcL\\_CuI](https://youtu.be/AGMKZcL_CuI)
- World Health Organization. (2021, April 01). *Deafness and hearing loss*. Retrieved from World Health Organization: <https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss>
- World Health Organization. (2022, October 13). *Blindness and vision impairment*. Retrieved from World Health Organization: <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>
- Young, J., Sharlin, E., & Igarashi, T. (2011). What Is Mixed Reality, Anyway? Considering the Boundaries of Mixed Reality in the Context of Robots. In *Mixed Reality and Human-Robot Interaction* (pp. 1 - 11). Springer Dordrecht.
- Zheng, J., Chan, K., & Gibson, I. (1998). Virtual reality. In *IEEE Potentials*. IEEE.

## Appendix

### Interview Transcript

**When discussing virtual reality, what would you consider to be a fundamental feature in any VR game?**

1. Player interaction with the world, movement options. How you interact with the world is very important in VR since you aren't looking at a screen, you are there.
2. As it is right now, VR is reliant entirely on utilizing a headset with optical lenses for visuals. So a VR game should utilize this in the most immersive way possible while still maintaining accessibility as an option as not everyone can move their head to change views.
3. Graphics are definitely the most important in any VR game cause the visual experience and immersion is the whole point.

**Do any games come to mind that implement that feature properly?**

1. Half Life Alyx, Hot Dogs, Horseshoes and Hand Grenades, VRChat, Blade and Sorcery, No Man's Sky
2. Currently I think VRChat does this well, you can swivel your head or use the joystick to turn yourself in game.
3. No Man's Sky is probably the best visually I've seen

**On the contrary, do any games come to mind that don't?**

1. Tabletop Simulator
2. Don't have enough experience with other games to answer this question
3. Nostos was the biggest let down for an immersive VR game I've ever seen

**Are there any accessibility features that you utilize in VR Games?**

1. Snap Turn as opposed to Smooth turn
2. I've tried utilizing the motion sickness setting in a few games, but the darkened edges takes away from the rest of the immersion.
3. None that come to mind

**What VR game, if any, have you spent the most time with?**

1. VRChat
2. VRChat
3. Phasmaphobia for sure because it's dumb fun

**Where does that game fail to meet your expectations?**

1. Poorly optimized, Lack of proper search function in game, the games Anti Cheat
2. That has two aspects, since most all content is community created, there are a lot of assets uploaded that are poorly optimized, causing poor system performance issues. As for the developer's shortcomings, I would say it has to do with the User Interface screens and settings. They are not intuitive and confusing to navigate.
3. Graphically the game is underwhelming but that's more of the games issue itself instead of a problem with VR

**Where does it succeed in meeting your expectations?**

1. One of the best VRGames for playing with friends, wide variety of content so you always have something to do
2. By having community driven and created content. It can allow anyone of almost any skill level to create something. Which means its friendly for people wanting to learn how to create to fostering extreme creative types to develop amazing content. That said, it is also nice that the game requires you spend some time in the game and meet certain criteria before doing so, in order to prevent mass spam updates of content from bot accounts, which I think is a good feature.
3. Ironically enough: being immersive. Even if it doesn't look that good it's still pretty scary.