University of Nevada, Reno

A Software Template for Multi-User Virtual Reality Applications

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Engineering

by

Lucas Emio Antonio Calabrese

Dr. Frederick C. Harris Jr., Advisor

August, 2020

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

Lucas Emio Antonio Calabrese

Entitled

A Software Template for Multi-User Virtual Reality Applications

be accepted in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Dr. Frederick C. Harris, Jr., Advisor

Dr. Sergiu M. Dascalu, Committee Member

Dr. Yantao Shen, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

August, 2020

Abstract

Virtual Reality (VR) is a new and exciting way to further immerse users in video games and is being explored to find new ways it can be used. Some of its uses include education, training, and entertainment. Virtual reality needs to be explored because of how it affects the user, whether it is due to the positive effects like increased immersion or if it is due to negative effects like fatigue or cybersickness. One area of Virtual Reality that needs to be explored is the aspect of multiplayer interaction.

A template that enables research in understanding this area is needed. This thesis aims to provide, a Virtual Reality multi-user template that other developers can use to assist their research. This template allows users to communicate via voice chat, provides samples of how to set up user avatars, provide samples of locomotion, and provides scenario recording and playback. The Unity game engine along with networking software is foundational to accomplish this. This template was successful in being used to create two applications: a simple ping pong game and an application for a user study in training medical doctors and nurses.

Dedication

I dedicate this thesis to my parents who provided me the option to work towards my Master's as well as helping me throughout life.

Acknowledgments

First and foremost, I would like to thank my parents for helping me through this journey. Next I would like to thank Professor Frederick Harris. He has provided aid and support that I did not expect, and has guided me towards completing my thesis. He also provided me with the means to explore a topic that I was curious about: multiplayer Virtual Reality, as well as work on a project related to my goals and interests. I would also like to thank my committee members Professor Sergiu Dascalu and Professor Yantao Shen for their time. I would also like to thank Justice Colby for helping me with the project throughout the semester and for being a positive influence. It has been an unusual last year given the pandemic. I must thank my labmates Christopher Lewis and Yifan Zhang who aided me in finishing an application for a user study when we gained access to the lab a short while before the application was due. Christopher has written some scripts that we used for gathering eye tracking data. I would also like to thank Steven Anbro for allowing me to assist him with his user study.

This material is based in part upon work supported by the National Science Foundation under grant number(s) IIA-1329469. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Tables

List of Figures

Listings

Chapter 1

Introduction

There are several multi-user VR frameworks in the literature, so why build another? One multi-user VR Framework that currently exists is Let's VR [26] which utilizes Unity and Unity's multiplayer system, UNet. However, UNet has been deprecated due to the Untiy team believing that "it does not meet the needs of of many multiplayer game creators"[24]. A similar framework uses Mirror [54] as a replacement to UNet [52], but it has some issues that we will discuss later. The VRChat SDK [77] provides users the ability to create worlds as well as set up avatars in Unity and upload them into VRChat, which is a online virtual reality multiplayer game that is used for socializing.

There are many applications that can be built off of a multiplayer VR template. Training applications can be made, which can allow users to be placed into situations that would otherwise be too unsafe in real life (training fire fighters, miners, military personnel, and others). VR has the advantage of providing immersive experience and we may want to have the training be as similar to real life as possible [63]. Multi-user VR training can be also used remotely so that users do not need to be co-located.

Multi-user VR applications can also be used for entertainment. While it is true that there are some limitations to VR, which include the possibility of inducing motion sickness, VR still provides immersive experiences along with fun motion controls. Multi-user VR can provide unique experiences that could not be experienced otherwise.

The rest of this thesis is structured as follows: In Chapter 2 we discuss multi-

player interaction, Virtual Reality, User Studies, and VR development. Chapter 3 shares an example application of Virtual Reality with multiplayer that uses Mirror for Networking. This chapter appeared as a conference paper in ITNG 2020 [12]. We found some shortcomings with the template that uses FizzySteamyMirror[54] and have developed a new template that utilizes Photon Unity Networking 2[18]. A new template that takes these issues into consideration can be found in Chapter 4. An example of the template's usage is then presented in Chapter 5. A user study run by Steven Anbro was done using this application is also discussed in this chapter. Finally we present our Conclusions and discuss Future Work in Chapter 6.

Chapter 2

Background and Related Work

2.1 Multiplayer Interaction

2.1.1 Networking Architectures

There are two different networking architectures for multi-user applications. These are peer-to-peer and client-server. Peer-to-peer is where the connection is established directly between clients with no dependence upon a separate server. It is called peer-to-peer because of this direct communication. Client-server, on the other hand, is where the connection is established between each client and the server [37].

2.1.2 Multiplayer

Typically in multiplayer games, one player has a copy of the game, while another player has another copy of the game. When we have interaction there is the concept known as authority. Authority is where one user's application (application1) has control over an object, while every other users copy of the game tries to match some or all of the objects properties with application1's. We need at least one application to have control over the object so that when different things are done to that object we know who is in charge of deciding. If we had no one in charge the object would have problems or conflicts while the other clients were trying to synchronize the properties of that object. Synchronizing objects is important to make users feel that they are within the same world when they interact with objects. In a client-server setup the server typically has the authority over all objects.

An Example of how a multiplayer game may work is seen in Figure 2.1. There are three clients. After some time has passed, the player from client #2 moves up, and all the other client's copy of that player moves up as well. The same goes for the red sphere, client one drops the sphere (which he has authority on), and that sphere's position is changed on the other client's. The blue sphere however is lowered by client #2, but since he does not have authority over it that sphere is not synchronized across all clients, it only drops on client #2's copy of the game. A case where we may not want synchronize an object's location may be a 3D menu where it would be unnecessary for another user to see it.

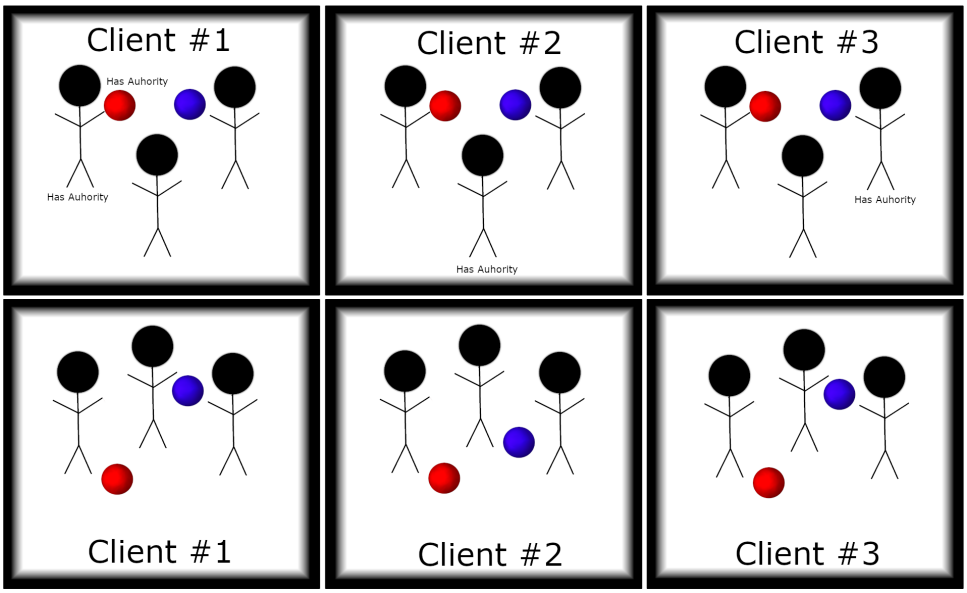

Figure 2.1: There are three clients represented in this image. The rows represent different states where the 2nd row occurs after the 1st row. Client #1 has authority over the left-most stick figure and the red sphere, Client #2 has authority over the middle stick figure, while the Client #3 has authority over the remaining stick figure

2.2 Virtual Reality

Virtual Reality is a technology used for interactive applications with aim to provide immersive experiences. There is currently a bit of research being done on the effects and the uses of virtual reality. But one issue that is actively being researched is

cybersickness because it limits Virtual Reality's growth. Cybersickness [35, 79] is an effect of virtual reality where a certain percentage of users who will feel nauseous after using VR. We will present some areas of research that have impacted our work in Section 2.2.1, mention some major VR Research labs in Section 2.2.2, and then talk about the hardware in Section 2.2.3.

2.2.1 Research

There are several areas of Virtual Reality research. The areas that have impacted our work include locomotion, education, avatars. Research includes, but is not necessarily limited to the following areas: locomotion, education and training, avatars.

Locomotion Locomotion is a very large topic. Locomotion methods are just methods to allow user move from point A to point B. Examples of locomotion include: Teleportation which is instantaneously moving from one position to the next. Glide locomotion which involves using the touchpad to slide to other locations. There is also walking where users can walk around in real life, and walk around in VR.

An example of teleportation can include pointing to a location and selecting that location by pointing to that location for a duration of time. [11] presented an overview of this and conducted a user study on this technique. A similar implementation can likely be done in Unity using Raycasting for selecting a position in space, and the function Invoke to call a function after a duration of time. There may be many different options for a user to confirm their selection for position. Such a technique may need to have additional rules to ensure that users do not teleport to far. There has been a thesis on teleportation where users do not use hands [67]. They used the following methods for activating the teleportation: controller, dwell, stomp, voice, and blink. There benefits to these methods as it allows those who may not be able to properly use their arms to play VR and it may provide options for other HCI options for others. Glide locomotion is used when you wish to move in a specific direction, possibly without a known destination. An example of Glide locomotion can be seen

in a tutorial book [41]

Another locomotion technique to be used may be redirected walking [55]. This uses a larger play area and is intended to allow the user to keep walking in the fixed play area without stopping in the VR world.

Some locomotion research compares methods. One paper compares [83] Walking, Walking in Place (WIP), and Arm Swinging. We have worked on user study that compares WIP and touchpad based gliding movement [4]. One author from that paper, compared different gait movements in his thesis [6].

There is a survey on locomotion[1] from 2018 that goes into the different forms of locomotion that have been researched and the taxonomies associated with them. These forms of locomotion methods were separated into four categories: walking, which involves different walking related options, steering, which can be compared to driving a car, selection, which includes selecting positions in space to move to, and manipulation. Walking methods include methods such as walking in place, where users raise and drop their feet. There are many ways for users to indicate when they want to move in the VE like, wiping feet, and tapping feet. Even Jumping can be used. Some other interesting options include arm swinging and finger walking.

They continued their work with Steering methods which include using the users gaze, steering with the users hands, and leaning, Selection Based: Select a position or object on screen, slowly drag there or teleport there. There are different ways for activating teleportation like jumping. Teleportation is considered useful due to lack of optical flow. If there is no optical flow, users are less likely to get sick due to mismatches between movement in the VE and movement in real life. Manipulation based methods include manipulating position, size or rotation. The paper goes into more detail and explains more methods than are presented here. There are clearly many ways to implement locomotion in VR, and each way can have many variations in parameters. The paper is useful in allowing researches to see what has already been researched, find more research papers, and to maybe even aid in giving researches new ideas for new locomotion techniques. Locomotion is one of the main research areas

when it comes to Virtual Reality.

Whichever locomotion is best depends on the context of the situation it is used in. For example, if there were a space training VR application, a locomotion system that is based on zero gravity would probably be appropriate, even though it may not be possible to provide users with the feeling of weightlessness. Teleportation in this case may not be appropriate since that is not what would be used in real life.

Education and Training: There may be some benefit to VR with regards to learning. There has been research on the effect of VR on memory [36]. This research includes a study where its goal was to see if memory palaces, like the ones used in classical times, could be useful for memorization using VR. The study compared using an HMD and a Desktop display and there were 30 male and 10 female participants. All participants used the same scenes on each display. They were given a challenge to memorize faces located in different areas in the palaces. The faces used were fairly recognizable. The participants' tasks were to indicate which face was at which location, after the faces are hidden. Based on their data the authors of the study believe that immersion as improved recall. While this may be one specific case that uses palaces, it is good to know that there may be some more educational benefit to using VR.

Another study looked into the effects of VR on different two types of learners [69]. This was a study that involved 32 university participants, where half were female. The introduction talks about the use of VR in learning, and how immersion and the senses are important to VR education. Among one of the tests the study used was rotation tests where users find an object that is the same as a presented object but rotated. Low spatial ability (LSA) learners, were shown to have improved using VR while High spatial ability (HSA) learners did not.

Steven Anbro has been working on a user study to see if virtual reality can be used for training of Doctors and Nurses in their patient hand-offs. More on this user study can be seen in Chapter 5. Some of his initial work can be found in [3]

Avatars: There is a journal article about using Vive trackers for body tracking [13]. The authors used inverse kinematics (IK) to estimate how the models should be animated in response to the user's movements. The article goes into the math of how to achieve this. In order to make humanoid models they used software called MakeHuman. The model along with the bones used to animate the model can be seen in Figure 2.2 They developed a tool to check the latency to help them sure that they are able to keep latency at reasonable levels. The authors used a case study, with 1 female and 12 male participants, to test the tracking. They found that users were interested in the body tracking, and that they would like to see it used in games in the future. In their study they only animated the arms. This may mean that another case study should be done to include leg tracking. They suggest that avatar personalization should be added in their future work section. Other results of their work can be seen in Figure 2.3

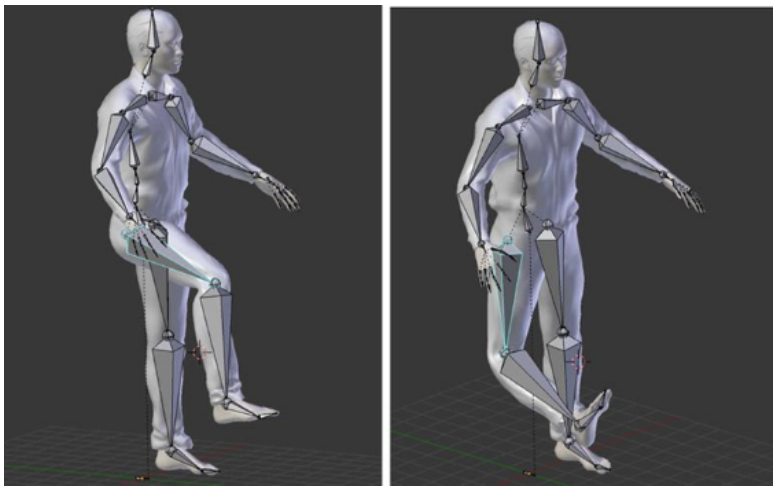

Figure 2.2: Image of 3D models and the bones that are used to animate them [13].

While it is important to know ways to implement full body tracking, it is also important to know how body tracking might affect the user. There is a conference paper where its main idea behind the paper is to look into whether or not using self avatars reduces cognitive load [68]. Some benefits claimed in this paper include that being able to see one's own avatar is useful for finding their location. The paper included an experiment that involved mental rotation and recollection. The paper

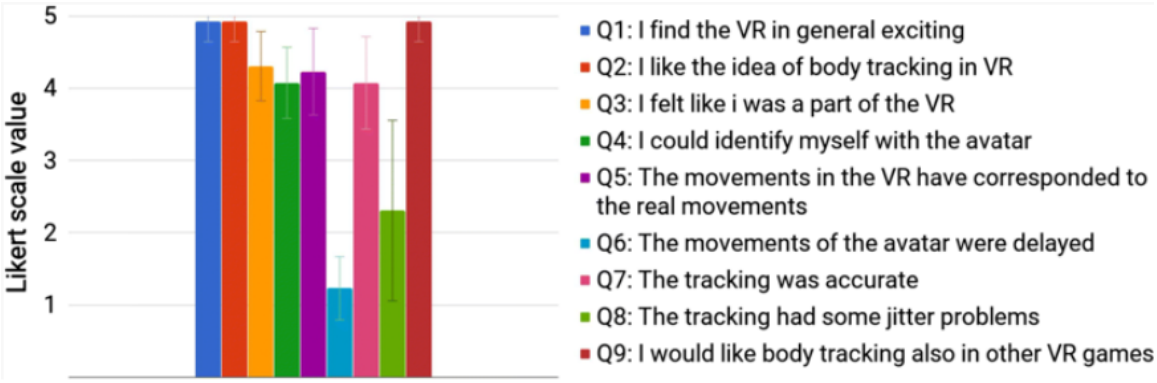

Figure 2.3: The graph shows results of the case study for the journal article on body tracking. [13]

mentions that body tracking with avatars may be for useful for communication. It also mentions that other studies found that full models are good for estimating distance. According to the study, utilizing gestures seemed to have improved recall. According to the graphs that mapped the movement of the users hands, self avatars were found noticeably increase the amount of gestures the users used. The ability to gesticulate appears to not only be useful for communication between one or more users, but it may also be useful for reducing cognitive load and improving recall. This may be useful knowledge for VR games that require communication and recollection. Maybe, it could be useful for problem solving tasks as well.

Avatar representation is a topic to be researched due to the affect that it can have on users. This can include social, accuracy, cognitive load, an immersion. It can be difficult to map a human user to the body of an avatar. There are differences in height, arm-length, and leg length. It may also be difficult to imitate the movements of real users, especially with fewer tracking devices to determine where the users arms are.

2.2.2 VR Research Labs

There are some notable labs and organizations researching the area of Virtual Reality. Some of these include, Facebook [21], LAVA at the University of Hawaii at Manoa [38], UA Little Rock George W. Donaghey Emergency Analytics Center [78], and Bowie

State Virtual Reality Laboratory [60]. The Facebook Lab's research [21] includes computer vision, among other topics such as graphics, audio, body tracking, and vision. They are planning on making AR, Augmented Reality, glasses in the future. Facebook's lab believes that Virtual and Augmented Reality will be common place. They have many publications, one paper Facebook Research has published was about animating faces. Some of their research includes social behavior in VR.

LAVA was founded by Professor Jason Leigh LAVA stands for Laboratory for Advanced Visualization and Applications. The lab has a CAVE known as CAVE 2. They have several published papers, like a broom flying simulator for a hybrid reality room [38]. One of the lab's main goals are to research data visualization for big data.

The UA Little Rock George W. Donaghey Emergency Analytics Center has a focus on data visualization [78]. An author of the original CAVE is from there. They work on training meant for helping people understand data. One project from that the lab is making a simulation for operations. This project uses data of the human body and is meant to be used for training medical students.

One of the current projects from Bowie State is Megacity: A Collaborative Virtual Reality Environment for Emergency Response, Training, and Decision Making [45]. The project focuses on using agent-controlled avatars for evacuation scenarios, as well as other disasters. This project aims to study strategies and potential scenarios for disasters. The agents are given different types of behaviors which include hostility, non-hostility, leader-following, goal-following, and a selfish behavior. These different behaviors are probably useful in helping users understand how people might act in dangerous situations.

Another project from Bowie State is Game-Theme Based Instructional Modules for Computer Science Students in VR [59]. The purpose of this project is to help with computer science and mathematics course. It uses gaming and virtual reality to try to help teach students. They have 7 instructional modules. Some of the games they have shown include, a duck game that is meant to teach programming loops, a game that visualizes concepts like stacks or binary search, and an array game. It is

interesting to see how VR might aid learning.

2.2.3 Hardware

There are several commercially available HMDs and input devices available. This thesis will use the HTC VIVE which can be seen in Figure 2.4 [29], The HMD for the HTC Vive provides a panoramic view.

Figure 2.4: The HTC Vive and wands [28]

The HTC Vive also has trackers which can be seen in Figure 2.5. The trackers are additional hardware that can be used to track other parts of the body, like the ankles of a user. The HTC VIVE system utilizes lighthouses to detect the positions of the HMD and controllers, as well as additional devices. The HTC VIVE Pro has some improvements added to the HTC VIVE [30]. The Vive Pro Eye enhances that with the hardware for eye tracking [31]. Eye tracking can be useful for knowing when the user is blinking (for locomotion or for menus) and to see what users are looking at. There are other hardware device to use virtual reality such as the Oculus Rift [22], Google Cardboard and the Google Daydream headsets [25].

The Hi-5 gloves as seen in Figure 2.6 may be interesting for research. They can be used for menus, interacting with objects, locomotion, and communication. An

Figure 2.5: Vive Trackers [27]

interesting menu used to show off the Hi-5 gloves can be seen in Figure 2.7. These gloves require some calibration to be used properly. There may be some interesting ways to use inputs from the user's fingers and removing buttons may help to preserve immersion. We have noticed in our testing that they do not seem to use rotations in more than one axis, except for the thumb. It can tell when the user is closing their fist, but it does not tell if the user is fanning out their fingers. This can be bad if a developer wants to use hands to allow users to communicate with sign language. Certain gestures are not possible without being able to rotate on more than one axis. However, we may be able to use gestures for other purposes, such as telling an army to regroup, opening a menu, giving commands to a virtual pet, or conjuring up a magic spell. Perhaps the gloves may be more immersive as well.

The VRFree gloves [58] appears to be worth looking into as they say they detect when the user is spreading their fingers.

Figure 2.6: Hi5 VR gloves[51]

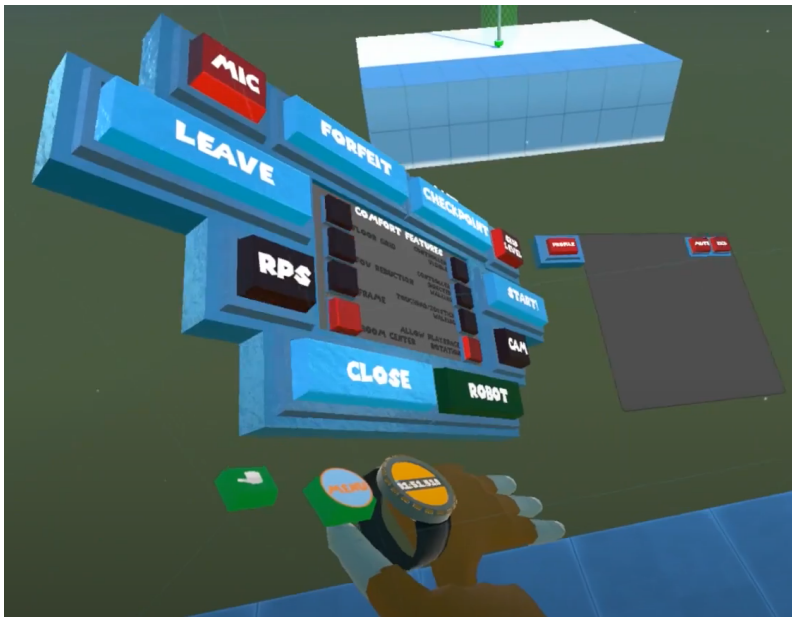

Figure 2.7: Menu for a game that uses the Hi-5 gloves[50]

Virtual Reality involves more hardware than just the head sets and controllers. For example, the CAVE and the CAVE 2 are examples of virtual reality. The CAVE2 can be seen in Figure 2.8.

Figure 2.8: Image of CAVE 2 taken from a YouTube video [40]

There have been some interesting devices made for Virtual Reality, one includes haptic shoes [66]. These shoes use an MR fluid, a fluid that changes viscosity when in contact with a magnetic field. It is capable of providing different sensations as the user steps. It can simulate different feelings such as the feelings of stepping in snow or sand.

Another interesting device, which can be seen in Figure 2.9 is a device that is intended to allow users to feel like they are stretching their arms further than they normally can. It does this by using a weight that moves to simulate the feeling of the arms being farther away than the user as well as a component that stretches part of the users arm.

One more interesting device, which can be seen in Figure 2.10, is the Gait Master which helps a problem with VR which is simulating walking on uneven terrain as it may be unusual for VEs to mostly be flat since real life environments are not mostly

Figure 2.9: Device that is meant to give the sensation of stretching arms [84]

Figure 2.10: Omni-Directional Gait Master [32]

flat [32].

Some hardware that can be used are treadmills. For example, the Virtuix Omni, is used to allow users to walk in place by allowing them to walk on a slippery surface [76]. It's website provides a source development kit that can be easily added to Unity. Some people say that using the Omni is a bit difficult to get used to and they feel a bit odd to have to lean in order to walk, but it is something that is worth looking into for VR. This is because it allows a form of walking in place that could not be done otherwise and it could help with the main issues with VR which includes preservation of immersion and reduction of cybersickness.

2.3 Virtual Reality Development

There are many approaches to developing Virtual Reality Applications. This section introduces a few.

2.3.1 OpenGL

Developing a game engine from scratch can be very difficult. There are many aspects to interactive games. One includes the graphics pipeline which can be incredibly complicated. OpenGL can be used to handle the graphics [57]. It is possible to use some existing libraries with OpenGL to develop a game engine. However, it may be a difficult task to accomplish making a template with OpenGL. It is likely much easier to use Unity or Unreal Engine 4.

When working with a CAVE development in OpenGL can be the primary tool available. There have been libraries developed to assist in this development. One of them is FreeVR [62]. This library handles all the frustum calculation of the multiple walls.

2.3.2 Unity

Unity is a free to use game engine. It is capable of developing successful games like A Hat in Time [23] (shown in Figure 2.11) and Ori and the Blind Forest [47], among others. It can also be used for to develop case studies. Unity uses C#. Unity has an asset store which provides software and other assets for developers to use. One example is Photon Unity Networking 2 [18] which can be used for multiplayer interaction.

It is important for a user to understand the basics of Unity to be able to make projects with it. Users should understand Unity's interface such as the hierarchy, project settings, scene, game view, assets folder, and the Inspector. Developers should understand what a `GameObject` is, what a Transform is, and what components are. Developers must know the difference between local position and world position. They

Figure 2.11: Image of A Hat In Time from the Steam Website [23]

should have an understanding of C#. They should know how to create and use scenes and prefabs. At any time if a developer is having issues understanding how to develop in Unity, they can use the Internet to find how others have solved the problem. Developers should also be aware of the asset store.

There are many functions that should be fully understood: Update, Awake, Start, FixedUpdate, LateUpdate, OnDestroy, OnEnable, OnDisable, OnTriggerEnter, OnTriggerStay, OnTriggerExit, OnCollisionEnter, OnCollisionStay, and OnCollisionExit. These functions' execution order should be understood as well. In addition, there should be some understanding of Unity's physics settings and RigidBodies. A developer should also have a basic understanding of animations, animation settings, model importing, and Unity's Animator. Being able to understand how to code shaders and use particle systems may be beneficial, but is not always necessary. Users should also have an understanding of different colliders such as the box collider or the mesh collider. Developers should have some background with math that utilizes 3D vectors. You may be able to learn from some of our mistakes mentioned in this thesis as well, in order to make better decisions for projects in the future.

Unity allows users to attach scripts and other components to `GameObject`s to give them functionality. Variables attached to scripts can be accessed and scripts can depend on other scripts (making scripts depend on other scripts may add complexity to the system, and may make the code less flexible, since errors may occur if certain variables are not available).

There are several networking options available for Networking. As mentioned before, Mirror is on of these. Mirror was based on the UNet system, but modified to be able to handle Massively Multiplayer Online (MMO) games [46]. There are also many transports available for Mirror which may include different transfer protocols. A previously mentioned framework utilized that utilized Mirror used the Peer-to-Peer transport, FizzySteamyMirror, which required a Steam app ID and each time a player would like to connect they would need to enter their Steam ID's [54]. This framework had at least one server that was also treated as a client. Mirror provides remote procedural calls for communicating to clients and commands for communicating to the server [46].

There is a simple to understand API for Photon Unity Networking 2 [18]. It also provides an easy to understand process for swapping authority. One option which involves marking `GameObject`s, that have a PhotonView script attached, as to how they should handle requests for authority and using `RequestOwnership()`. It also is able to use TCP and UDP and Photon Voice 2 [19] provides remote procedural calls to clients and provides several options for who the calls can be sent to for example if they send to all clients or all other clients.

Photon can either use their cloud or you can self host the server. Cloud usage is limited to 60 GB of traffic for the total amount of data that can be sent using the service or you can pay for more data [20]. One of it's main benefits is that it already provides code for voice based communication across clients. Self hosting is another option which allows the user to use up to 100 concurrent players without a limitation on traffic. But using a server can be expensive. Photon Voice comes with voice chat as well as sample scenes which can easily be built off of, and it is

client-server based [18].

Two other networking options are Dark Rift 2 and Forge Networking Remastered [8, 14]. Both Forge Remastered and Dark Rift 2 do not have a limit for concurrent players. Forge networking Remastered can be used for MMO, real time strategy, first person shooters and more [8]. For all of these networking tools, Dissonance voice chat could probably be used for voice chat, but it currently costs $75 USD [64].

2.3.3 Unreal Engine 4

Unreal Engine 4 is another game engine than can be used. Unreal Engine 4 uses C++ rather than C# [16]. Unreal Engine provides documentation for their Networking [15] and according to their documentation, they use a Client-Server model. The documentation provides a tutorial on how to get started on using multiplayer with Unreal Engine 4 and provides tips, such as how often Remote Procedural Calls (RPC) should be used and to use UDP if there is a (RPC) once per tick.

The networking used by Unreal Engine 4 has some similarities with Photon and Mirror's networking, they both use RPCs and they have similar methods for synchronization. With Unreal Engine 4's actors, which are like `GameObject`s, position, rotation and scale can be replicated from the authoritative client by enabling Replicate Movement [15]. Mirror and PUN2 use scripts to enable synchronization in a similar way.

The Unreal Engine 4 documentation also includes documentation for VR development [17]. This includes documentation for SteamVR.

2.4 User Studies

2.4.1 Background

As stated previously, there are effects that virtual reality can have on users. Since users may experience virtual reality differently, it is important to gather data on how users are affected by the software and or hardware. For example, if a researcher comes

up with a new locomotion technique, they will want to know if users feel sick after using the method, if they feel tired, if they feel immersed, and perhaps other effects. A user study can be done to see if a method of locomotion makes users walk more, rather than just teleporting [53]. We want to ensure that we validate our software, using users outside of those developing the software [61]. Findings from a user study may help contribute to finding the best methods to use in different situations and validate new ideas and techniques.

2.4.2 How to Perform a User Study

In order to conduct a User Study at UNR, we must get the user study approved first and undergo Citi Training. Paperwork must be submitted for IRBNet's approval. For a full guide on this, see Kurt Anderson's Thesis [6]. In his thesis he described in detail the user study he conducted and the process and paperwork to go through in order to obtain necessary approval for a user study.

When designing questions, those in charge of the study must consider what is important data to record and how participants will answer them. Perhaps comfort, feelings of motion sickness, frustration, and ease of use may be items to look for in participants when doing a user study on locomotion. What needs to be looked for depends on what the goal of the study is.

In order to gather that data, you can ask participants to fill out questionnaires to see how they reacted. Participants need to be able to fully understand what the question is asking for in order to provide accurate responses. It may be good to consider asking each individual question verbally to the participants to ensure that they fully understand the question and that the case study leaders gather a sufficient amount of information from the response. Likert scales can be used to help determine the level of a response, for example, how sick did the user feel? Labeling each value of the Likert scale may help to alleviate issues with ambiguity.

Hard data can be collected. This data can be used as further evidence of potential claims of methods used. For example, if a user says they were feeling fatigued, and

we see that their progress has slowed down during testing, we can use the hard data to as further evidence that a method causes fatigue. The time that progress starts to decline may also tells us that there is a limit to how often a method can be used. That limit may be reasonable enough to use for certain types of applications. For example, if users become fatigued due to constant walking, we can have applications where users take breaks at times to prevent the fatigue.

Whether we need a control group depends on the type of study we are doing. If we are comparing two locomotion methods and we want to see which method users prefer, we may want to see how users respond to both methods. However, we may want to have separate groups in certain cases to ensure that repeating through the experiment does not affect our goal. For example, if we want to compare VR training to real life training, we may not want a user to experience both, since they will have practice from the previous method.

2.4.3 User Study Examples

Some studies use multiple methods to compare one method to another. This can help to show what one method may excel. For example, one study [10] showed that users are less likely to cheat when they are given feedback when they interact with walls.

In Figure 2.12 it can be clearly seen in the users paths walked, that users who did not receive sensory feedback would walk through walls more.

It is important to have both male and female participants. Female participants may be more prone to feelings of motion sickness than males [2]. There may also be other differences that affect their experience during a user study. So we want to make sure we see how it affects both males and females.

There have many user studies on locomotion which can be seen in the survey paper by M. Al Zayer, P. MacNeilage, and E. Folmer [1]. There have also been studies on the affects of user avatars. As we mentioned before, there was a study that looked into the affect that models may have on the user's cognitive load. There was also a study where there where the art of the models were used to see how

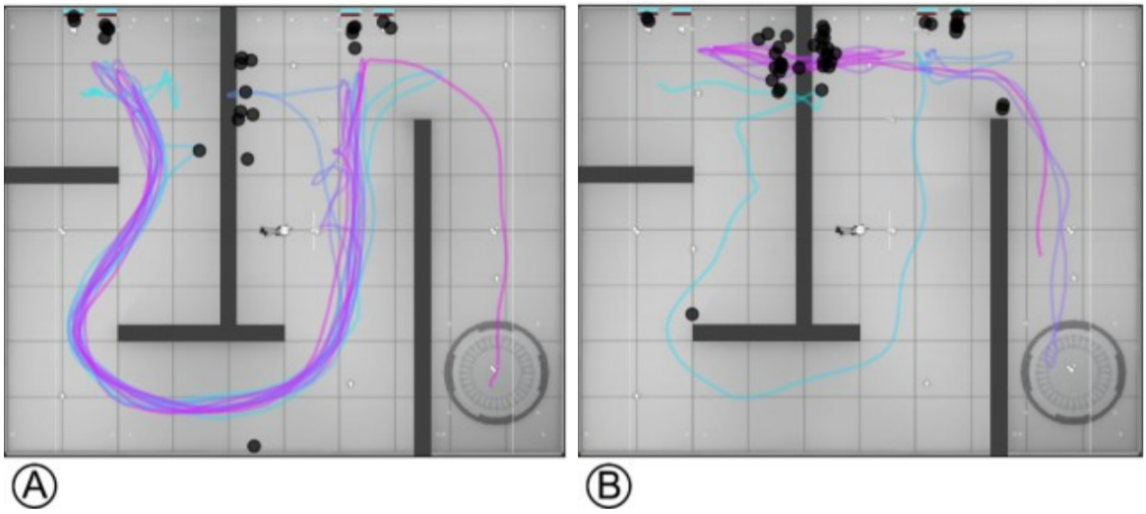

Figure 2.12: Top down perspective of a VR user-study. A: User received sensory, feedback B: User did not receive sensory feedback.The black dots are collisions with the wall, while the paths change colors based on time that passed. The dark gray lines are walls [10].

they affect the user's feeling of ownership over their avatar [33]. In this study there where 12 participants. Users had less ownership of the realistic model than the cartoon-like version. There is another paper that discusses ownership and it relates to anthropomorphism [43]. Another looked into the emotional affect of avatars [34]. A user study that looked into the accuracy of high-fiving between two users was also done [33]. There were 14 users in total. The study involved two human participants where they try to high five each other. It compared high-fiving when using just hands to using models.

User studies can have limitations. There may be some missing pieces that future work can add to see how the results change. For example, in a user study based on a locomotion technique, if the users move at a relatively quick speed for both techniques, how do we know if the user will feel differently about the same techniques at a slower speed? We have come across this issue when comparing a walking in place locomotion method with a controller based method [4].

We now move to multi-user Virtual Reality applications. Our first attempt appears in Chapter 3. The structure we used had some issues which we attempt to

correct with our software template in Chapter 4.

Chapter 3

First Attempt: Multi-User VR Cooperative Puzzle Game

This chapter first appeared as a Conference publication in ITNG 2020:
[12] Lucas Calabrese, Andrew Flangas, Frederick C. Harris, Jr. (2020) Multi-User VR Cooperative Puzzle Game. In: Latifi S. (eds) *17th International Conference on Information Technology–New Generations (ITNG 2020). Advances in Intelligent Systems and Computing*, vol 1134. Chapter 39, pp 293-299. April 6-8, Las Vegas, NV. Springer, Cham, DOI `https://doi.org/10.1007/978-3-030-43020-7_39`

Abstract

Multi-user virtual reality (VR) games are at the cutting edge of interpersonal interactions, and are therefore uniquely geared towards real-time interactive games between human players. This paper describes the process of designing a cooperative game where the obstacles are designed to encourage collaboration between players in a dynamic VR environment. This is done using the Unity game engine and the Blender graphics modeling tool. We demonstrate the progress of our scheme in a multi-player cooperative game, as well as the importance of the VR interface for encouraging cooperation. The VR experience provides a realistic human-human interaction improving on generic game-play, as our system utilizes the real-time interface to create an entertaining VR experience.

Keywords: Multi-player, virtual reality, Real-time, interactive, Unity

3.1 Introduction

The advancement of VR technology has opened the door to many different possibilities considering the numerous applications for it, one of which being gaming. To explore how VR technology can be used in multiplayer games involving a virtual environment (VE), this paper will discuss the process of designing a two-player cooperative VR game. This game was customized for the HTC Vive headset and the Steam VR software. The game was designed for two players to work together to overcome obstacles. Teamwork is not only encouraged, it is required if the players wish to successfully advance through the levels.

Games such as this will encourage multi-user VR scenarios [81] and create a more social atmosphere for players to enjoy. In this game, the players utilize different powers that come in the form of crystal balls that can be picked up, and that are placed strategically throughout the game world in a way that the players will have to make use of their problem-solving abilities to reach them. Once the powers have been obtained, the players will have to use them in a specific way to solve the current obstacle in front of them. Multiple improvements can be made to make the game better as a whole that is described later in Section 3.2.2, but due to time constraints, these improvements are not present in the prototype version of the game.

The rest of this paper is structured as follows: The Creation Process is described in Section 3.2. Gameplay is presented in Section 3.3, and Conclusions and Future Work are covered in Section 3.4.

3.2 The Creation Process

3.2.1 Blender Modeling

The initial stages of the creative process involved developing the models for the game using the open-sourced 3D computer graphics software toolset Blender [9]. The witches were constructed by molding two mirrored cubes together to create the torso and then the rest of the body. Other objects were attached to the body to create the

arms and shoes. Blender's bezier curves were used to create the hair of the witches and were set as children of one of the bones after being imported to Unity [73].

The next step was to create the animations for the witches. One of the better animation papers was written by Narang, Best, and Manocha [49]. A basic algorithm has been implemented into Blender using the Rigify add-on to use a human rig. Our model, the Rigify, and animation controls can be seen in Fig. 3.1.

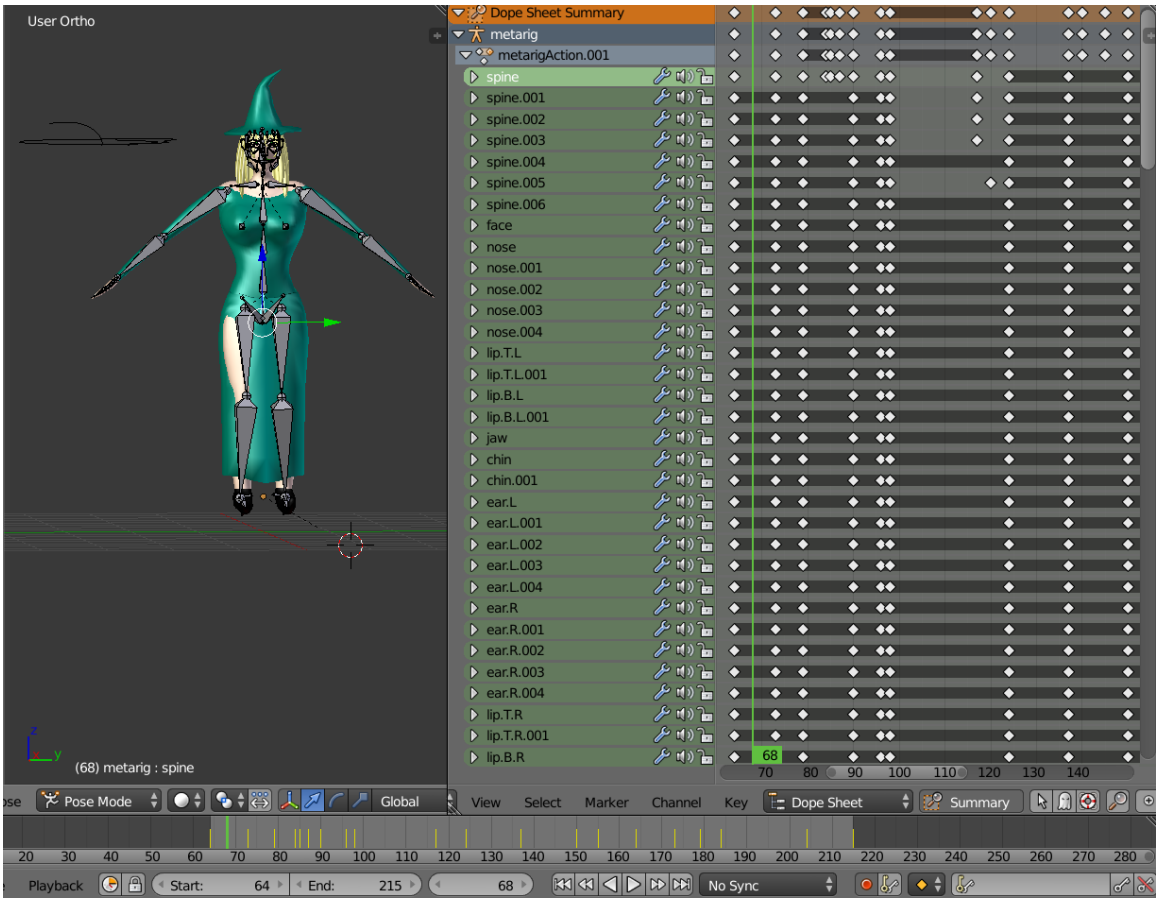

Figure 3.1: A Screenshot in Blender which shows how to setup animations using the human rig for the witch model.

Automatic weights were used for the animations, but some adjustments were made with weight painting. The animations for the witches included walking forward and backward, sidestepping, and jumping. The arms were intentionally not animated for walking so that they could be controlled with Inverse Kinematics. Once the two witches were created, they could then be used for the initial stages of the development

of the game in Unity. A scroll to act as the selection menu for the powers was also created by molding a single cube. The next models to be designed were the crystal balls that the witches collect and use in the game. The crystal balls were comprised of transparent sphere objects with an animated object in the middle that represents the power that it grants. To go along with the crystal balls were the crystal ball stands to keep them in place and to spawn them. The crystal balls and their stands can be seen in Figure 3.2

Figure 3.2: Crystal Balls representing powers on their Stands

Later in the developmental stages of the game, Blender was used once again to design an octopus-like creature with four tentacles and a water projectile for it to shoot at the players. The octopus started as a single-cylinder that was molded into the shape of the head, and then four mirrored cubes were used for the tentacles. Blender's Inverse Kinematics was used to make tentacle animations. It was given idle, walking, and attacking animations. The head was given a bone so it could look up and down, while the entire model rotates to face the player. The water projectile was also given a rig to create the animation of it swelling and bubbling like a ball of water. Later in the development process, the levels were designed in Blender and then imported into Unity. The final model can be seen in Fig. 3.3.

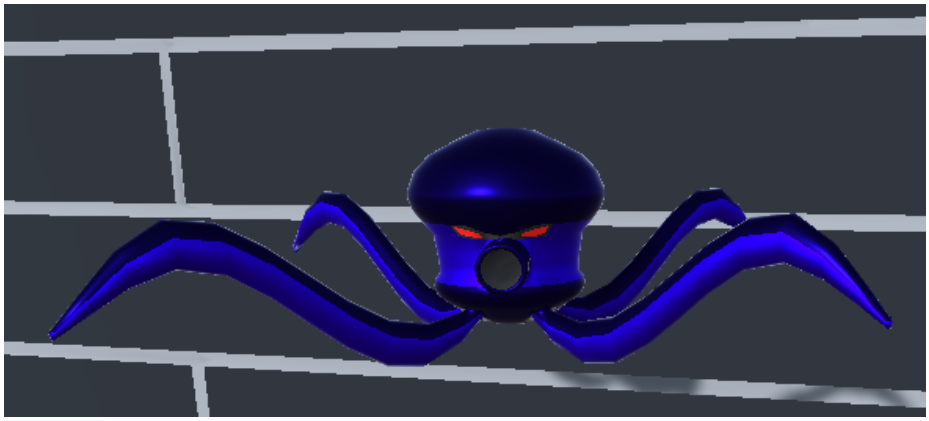

Figure 3.3: The Octopus model

3.2.2 Development in Unity

To test the powers that the witches use, as well as other gameplay features, a sandbox was created with a single plane as the floor of the scene with four walls surrounding it. The first object created in the scene aside from the planes and walls was the player prefab. The player prefab consists of a VR camera along with a right and left-hand object.

Then it was time to attach the scroll to the transform of the right-hand controller. The transform of the controller was used so that it can be rotated more freely than if it was attached to the model's hands. The purpose of the scroll is to act as a selection menu for whichever power the player wishes to use, as well as keep track of the number of powers the player has picked up. For buttons that activate powers, the scroll used models of the crystal balls that were scaled to look like buttons. A box collider was used for each of the witch's hands to register when the hands were touching a button. Text Mesh Pro was used to display the amount of each of the crystal balls collected by the user. Two more buttons were added to the scroll, a stop button to cancel any power currently being used, and a swap-hands button for the scroll so right or left-handed people can choose the setting that is most comfortable for them.

A script was used to fix the model's position slightly behind the camera. The

character controller that is used for detecting collisions adjusts its central location to keep all players at the same height regardless of their height, or whether or not they are sitting down. Cloth was used for the witches' dresses, in which capsule colliders attached to the model's bones were selected to allow them to collide with the dress. Because the cloth would get stuck on the colliders, a script was added to reset the cloth under certain circumstances such as when the model jumps.

3.2.3 Developing the Powers

After the Inverse Kinematics and scroll were set up, it was then time to focus on the coding of the powers. It was decided that there would be five powers: swap, shrink, freeze, bomb, and a fire power for this prototype. In order to obtain a power, the user must find and have their player touch a crystal ball representing that power.

All five powers and their associated effects are illustrated in Fig. 3.4-3.8.

Swap Power: The swap power (Fig. 3.4) is used to instantaneously switch the position of the players with other `GameObjects` in the scene.

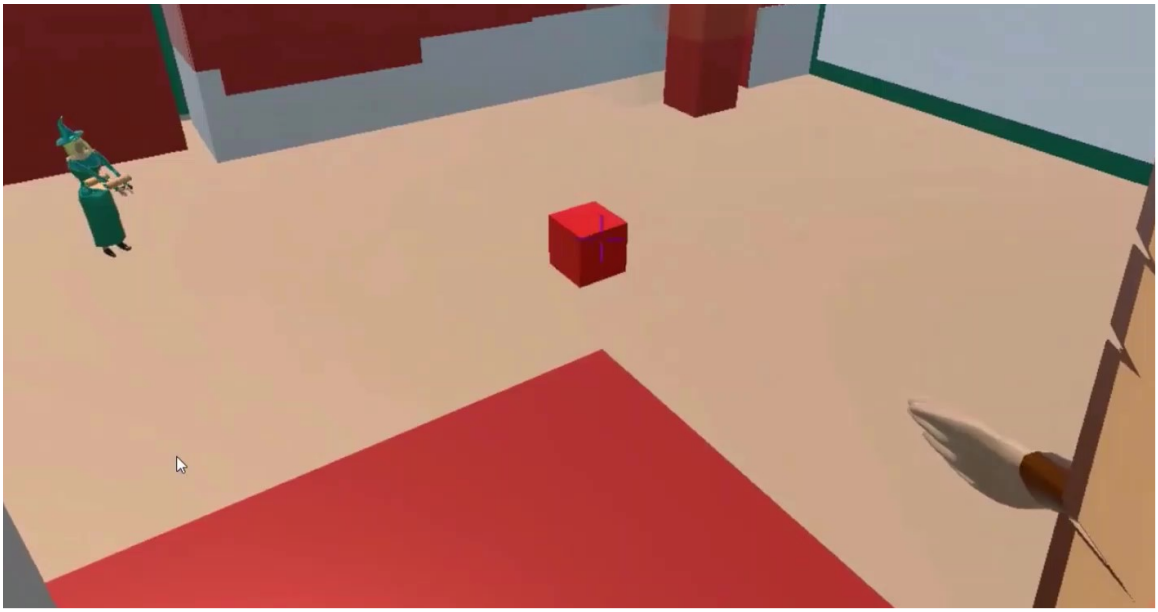

Figure 3.4: A first person view showing the object that the swap power applies to has its material changed to red. After the swap power is applied, the player will switch places with the cube.

Shrink Power: The shrink power (Fig. 3.5) is designed for the players to fit through small tunnels or other similar obstacles by making the player significantly smaller.

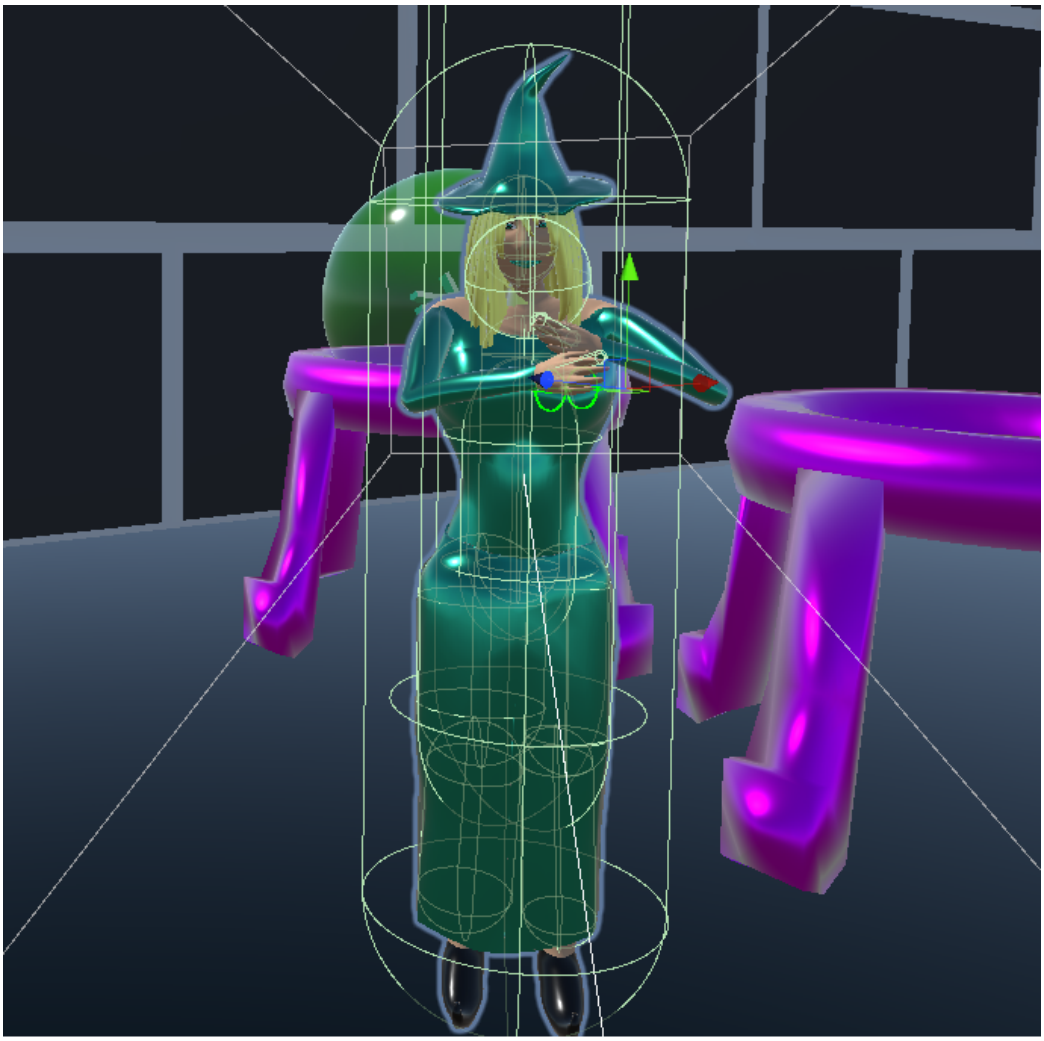

Figure 3.5: The shrink power scales the player to a considerably smaller size. Notice the crystal ball stands next to the player.

Freeze Power: The freeze power (Fig. 3.6) is used to turn the water projectile the octopus shoots at the player into a cube of ice, and then to use the cubes of ice as a jumping platform. This was meant to encourage teamwork as the octopus would follow one player around and shoot a bubble of water at that player and when it is turned to ice the other player who can use it as a platform. An additional purpose that

was added to the ice power was causing a balloon object to descend when activated due to the change in the balloon's volume due to its cold temperature.

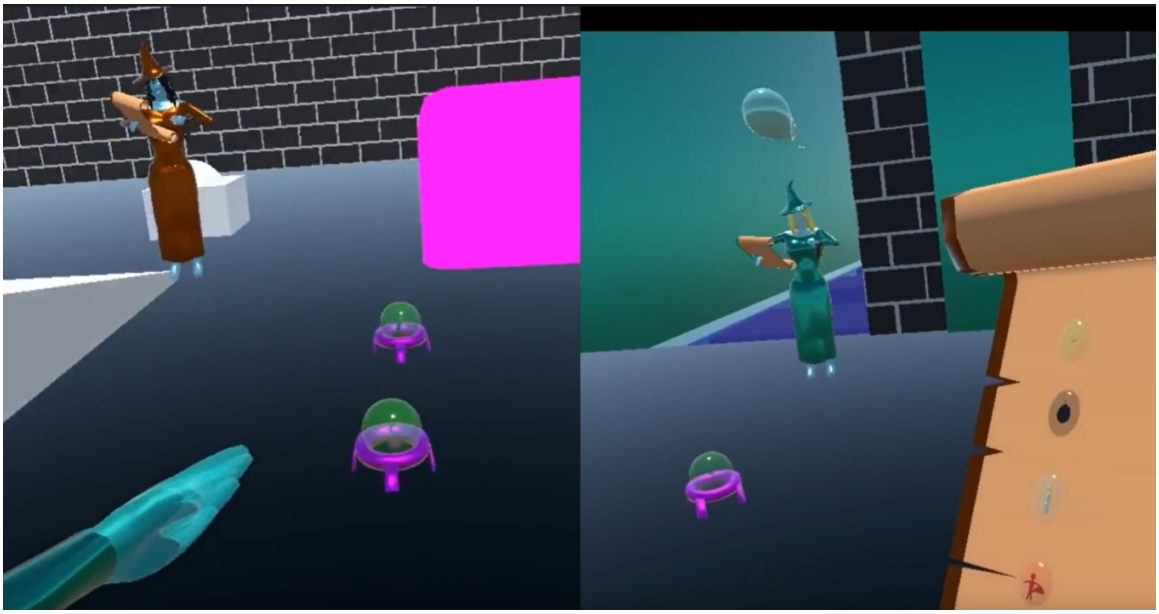

Figure 3.6: A split screen (two images from different player's screens). When the ice power is selected and activated it changes the material of your witch's skin into a light blue color. The ice power is bringing down the balloon seen in the right window.

Bomb Power: The bomb power (seen in Fig. 3.7) appears in the hand of the player when selected. The player can then grab and throw the power at something else in the game. This bomb power then explodes on contact.

Fire Power: Lastly, the fire power (Fig. 3.8) is designed to melt the already frozen cubes that are created using the ice power and cause balloons to ascend. Some functionality could still be added to the ice and fire power to give them more use.

To make these powers accessible to the players via the scroll, a powers script was created and attached to the witch `GameObject`, which was a child of the player object. In this script, each of the powers are stored in a queue, and only accessed when the player presses one of the buttons. The queue stores crystal balls. These objects are returned to their stands either 7 seconds after use, or if they are away from their stands for 7 seconds. `GameObject`s were stored to make the transforms of the crystal balls and the setActive function easily accessible. It was essential to create a UI in a

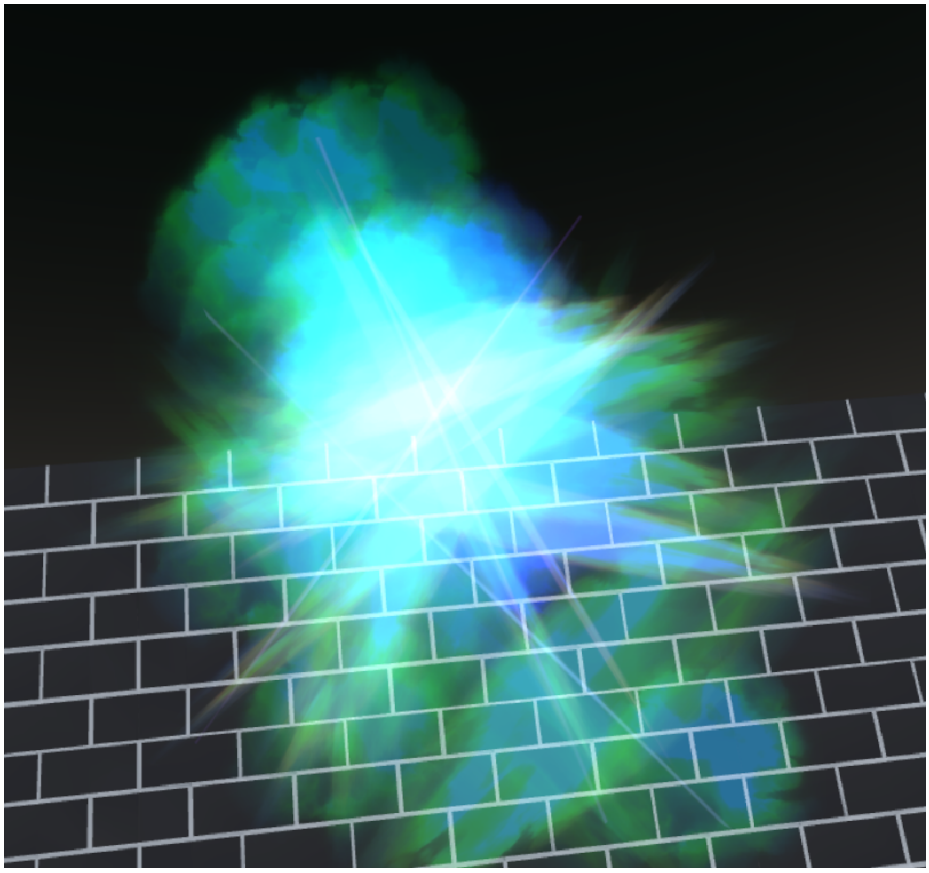

Figure 3.7: The fireball is used for the bomb explosion power.

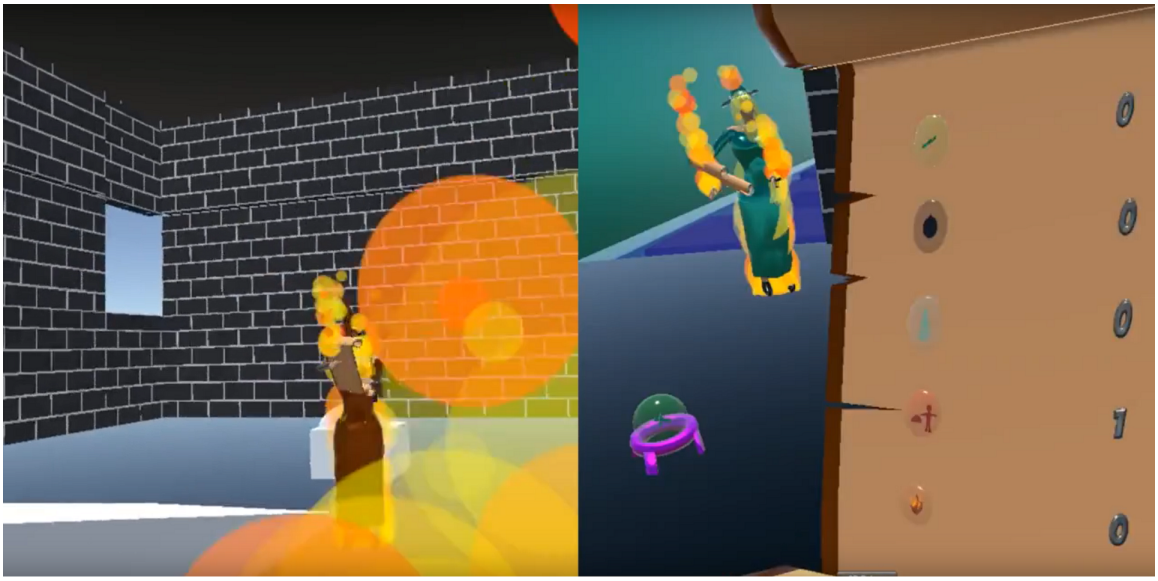

Figure 3.8: A split screen (two images from different player's screens). When the fire power is selected by both players, it activates the fire particle system.

meaningful and useful way [39] for the user. When the swap power button is pressed, a function gets called within the powers script which then accesses a function that is located in a separate swap script. A similar method is used when selecting the bomb power, in which there is a separate script for the bomb power that is attached to the explosion prefab that is accessed in the powers script. The remaining powers are accessed and implemented in the powers script while having other scripts attached to the parent `GameObject` for networking purposes.

To get the fire and ice powers to work properly, the OverlapSphere physics function is used to detect when the hands of the witch are touching the ice cube or the water projectile. While a player has the ice power activated, they can freeze the water projectile. When the player has the fire power activated, they can melt ice cubes and cause them to disappear. For the fire power, it was also necessary to add an OverlapBox to melt the ice cubes when a collision is detected between the ice and the rest of the body. When the fire power is selected, a fire particle system is activated that engulfs the witch object in flames. When the ice power is selected, it changes the materials used for the witch's skin color into a transparent light blue

material. The shrink power changes the local scale transform of the player prefab to a smaller size. The swap power moves the player prefab in a way that allows the witch model and camera to move to the position of the `GameObject` it is switching with. That `GameObject` then moves to the position of the witch model. The bomb power creates a custom prefab fireball object and when the fireball object detects a collision, it instantiates an explosion prefab.

Sounds had to be added to each of the powers. Royalty-free sounds or sounds we recorded were used for the powers. They are essentially open-source and can be used by anyone. The sound used for the swap power sounds like a slab of concrete being shifted across another hard surface. The sound for the shrink power sounds like rubber being stretched. The bomb power makes a loud bang when the fire ball collides with another object, using a royalty-free sound. The ice power is a custom sound made by crumpling a piece of paper and then editing the effects in an online music tool called Audacity [44]. The fire power uses a built-in sound in Unity that comes attached to the fire particle system. To attach each of the sound effects to the powers, a sound source component was attached to the witch and then specified in the powers script when the sound was supposed to be heard. The only sound that had to be specified differently was the bomb power, in which the fire ball spawns an explosion and the sound source is attached to the explosion.

3.2.4 Level Design

Demo Level: The first level was initially modeled in Blender, additions and edits were added afterward. The levels had to be designed according to the powers that would be used in that scene. There would have to be small constrained passages for the shrink power, platforms placed at higher locations that can only be reached by creating ice platforms from the octopus's bubbles, and empty spaces to place objects to either swap or blow out of the way with the explosion power. All these factors had to be taken into consideration when designing levels that would complement the usage of the powers.

The demo level features a puzzle that involves using the ice and fire powers to manipulate the position of a balloon. The objective is to use the ice power to make the balloon drop in height and the fire power to make it rise. The players repeat these actions until the balloon makes it out of a winding tunnel. Once that happens, the balloon rises above a platform. This is so a player can then use the swap power on the balloon to get to a higher location. The ice power was used to create a platform out of a water projectile to reach a swap power. After using the swap power to swap positions with the balloon to reach a high platform, a bomb power is then collected. The player on the high platform uses the bomb power to knock over crystal balls that contain the shrink power so that the other player can grab them. One of those balls is then passed to the other player so that both players can shrink and to reach the end of the level. The level can only be concluded once both players touch the square block at the end of the level. Upon doing so, a congratulatory message appears.

New Puzzle Level: As the demo level was made to illustrate how the powers could currently be used, another level was made to test the puzzle aspects of the game. The designing of this level involved the creation of several new models in Blender, which are purple barriers and buttons that are used to open them. The objective of this level is to figure out how to knock down crystal balls with the shrink power that are guarded by three barriers. A picture of the level can be seen in Fig. 3.9.

There are three buttons that correspond to the three barriers that are placed in separate ends of the map, while a lone cube sits in the middle of the level. While the buttons are pressed, their corresponding barriers are disabled. Two of the platforms require one player to use the other as a platform so that they can reach it. One player provides a hand for the other player to jump on to allow that player to reach these high platforms. There are also three swap powers and one bomb power available. The solution involves some set up. One player will need to bring the cube up to one of the platforms, while the other player will need to collect all swap powers that are within the level. One player will be called Player1 and the other Player2. Player2 will use Player1's help to reach a platform that is in front of a long highway filled with three

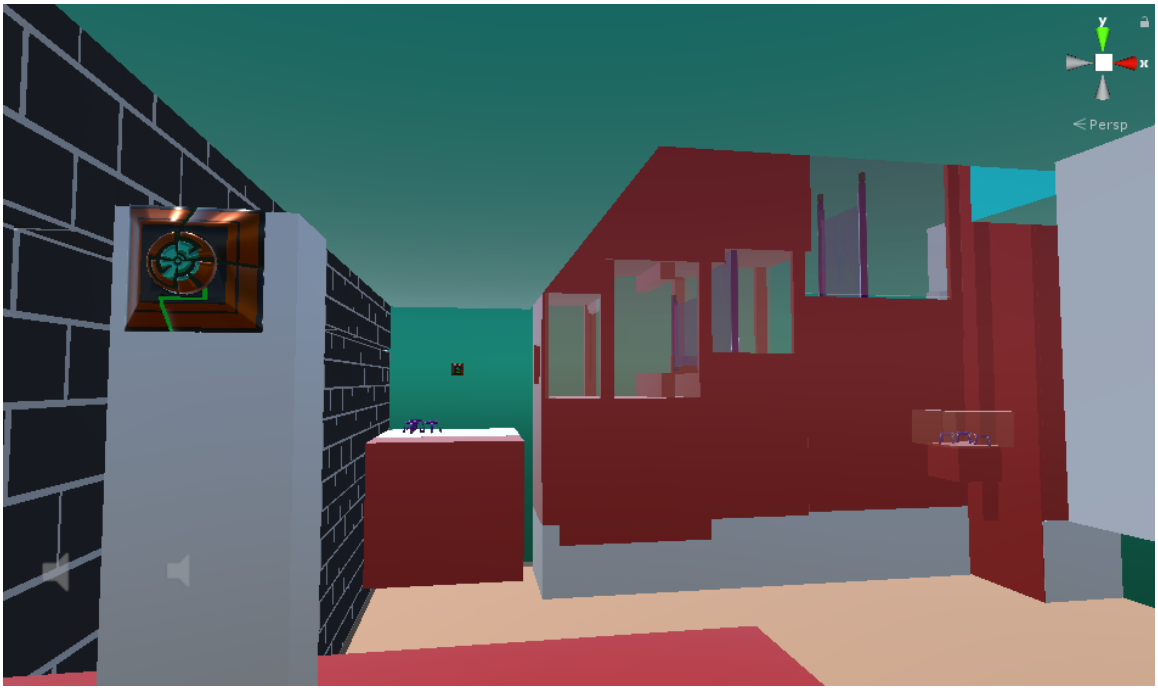

Figure 3.9: The puzzle level with the barriers, as well as the buttons the players use to open them.

barriers that ends with crystal balls that each contain shrink power. Player1 will go to the platform that does not have a cube. Player2 will slowly release a bomb spell towards the shrink powers. Since the bomb power is still active, Player2 cannot use other powers. Player1 will swap with Player2 so that Player2 can press the button located at the position Player1 is at. This opens up the first barrier. Player1 then swaps with the cube to press the button that is at that location. As the button is pressed, the next barrier is released. Once the bomb spell has passed that barrier, Player1 will swap with the cube again and then reach the final button as Player2 goes to collect the shrink powers as they fall down. The level is then completed.

3.2.5 Mirror Networking

Unfortunately, during the time this game was being developed the unity networking feature known as UNET was deprecated. The alternative used was the Mirror networking API found on the asset store or the Mirror public GitHub repository. There is a sub-branch of the Mirror API known as FizzySteamyMirror that allows the users

to link a host and client-server using their steam IDs [52]. Once FizzySteamyMirror was downloaded and installed successfully into Unity, the next hurdle to overcome was to sync up the player's movements between the server and client. To accomplish this, a networking transform child was added to the appropriate `GameObject`s of the player prefab, along with a script to disable any action that does not belong to the player on their side. After these tasks were accomplished, the player's arm movements and walking animations were visible on each others' screen.

After both of the character's movements were visible on both the server and client, it was time to make sure that the powers worked online. To achieve this, a networking script had to be added to the root `GameObject` of the player prefab for each of the five powers. These scripts are there to ensure that the game is synced over the network. After completing all five scripts, the powers used from either player could be seen by both users. Then Mirror's scripts were added to the appropriate power orbs so that the displacement of the power orbs, whether they are picked up or knocked out of place, as well as the position of the octopus, can be seen objectively on the same server.

To allow the players to see each others' arms move, the inverse kinematics scripts were kept enabled. They used the information about location of the hands and HMD sent from the other player to use for the inverse kinematics scripts to approximate the arm placement. For the bomb power, the local player had control over the spawned spell prefab. For the ice, the local player's materials are swapped and information is sent to the other player to change the materials of the non-local player. Something similar is done for the fire and shrink powers. When the client spawns a player prefab, the witch's materials for the clothes, hair, lips, and eyes, are changed so that the characters are distinguishable from the player spawned by the server. Networking eventually turned out to be a success, Fig. 3.6 and Fig. 3.2 show the two players interacting in different environments. There was other networking related work involving correctly spawning objects like the octopus, the balloon, and the spells and these are covered in detail in [52].

3.3 Gameplay

3.3.1 Locomotion

The locomotion method used included both room-scale and the controller to move. It was similar to glide locomotion [42]. Using the trigger by itself moves the player in the direction the player is looking in. Touching left or right on the touchpad, and then pressing the trigger would allow a side step. Touching back on the touchpad and then pressing the trigger would allow the player to move in the opposite direction that the player was looking in. The trigger was used since the touchpads seemed to jam easily. This locomotion method was chosen as it seemed simple to implement. A method such as teleportation was not used as it could look unusual to see a model repeatedly and instantaneously moving to different positions. Jumping and gliding were also added. When a player falls, they will fall at a constant, slow rate and can use the touchpad to move forward, left, right, or backward while gliding.

3.3.2 Positive Outcomes

The goals for the gameplay of this project included the possibility for depth in gameplay, the feeling of being on a team, variety in puzzles, and to make use of the motion controls that VR provides. One way that the game tries to encourage the feeling of being on a team is that players in the game can stand on each other. One player can hold out their hand to provide a platform for another player. This can be used as a method to separate players, or to make areas inaccessible without having to use powers. Another way the game tries to encourage a feeling of teamwork is the ability to pass collected powers to teammates. This is done by holding the model's hands to the button on the script and pressing the side buttons on the Vive controllers.

As mentioned before, the ice power was meant to encourage teamwork by allowing one player to freeze bubbles shot by the octopus while the other player tricks the octopus into sending them over. The fire and ice powers are not necessarily complete, as the original idea involved players not being able to enter certain areas unless those

powers were activated. For example, not allow a player not using the fire power to reach hot surfaces.

The balloons are meant to encourage teamwork by using the ice and fire power to cause the balloon to rise and fall. This step is repeated until it is in a position where a player can use the swap power on it. This idea was not explored greatly, but we believe it is usable for interesting puzzles. The bomb power takes advantage of the motion controls as it allows an explosion spell to be thrown. The swap power utilizes VR controls by using the HMD to aim. This power can use other players as objects to swap with, which encourages teamwork as it may be necessary to move a player to another location. Also, when a player is using a power, another power cannot be used. The other player would have to be in charge of using other powers which encourages players to choose roles.

Another important component added later in the game's development was the voice chat feature. Voice chat allowed the players to communicate with one another in the game while pressing and holding one of the touchpad buttons on the HTC Vive controller. This was done using mirror to send data over a network and using audio sources to play them [52]. The feature is a necessity since players will need to communicate their ideas to solve puzzles.

3.4 Conclusions and Future Work

3.4.1 Conclusions

This project demonstrates only a few of the countless exciting and innovative features programs like the Unity video game engine and Blender have to offer. However, the game was a successful project in the sense that it meets all the criteria initially set for it. It is a co-op game that not only reinforces teamwork but also requires it to make it through the demo. The five powers all have interesting visual effects and sounds attached to them, as well as situations where the players need to implement them. There is room for improvement in many areas of the game, but overall it is sufficient

for what it is intended. That being a great VR learning experience.

3.4.2 Future Work

While there are many items which could be added here, we will point out a few that we feel are important. Comfort mode [42], a method in which the user can turn their head without changing direction in the game could have been added for users who prefer it. Jumping, even with its potential to cause VR sickness [82], and the possibility of affecting immersion were kept in the game as it was deemed useful for gameplay purposes. A user study should be done to see how users feel about jumping and to gather feedback on the prototype. Since this game is still a prototype, the powers could be adjusted and more interact-able assets could be added. Other multiplayer services or libraries could be added such as Photon Unity Networking 2 [18], Dark Rift 2 [14], and Forge Networking Remastered [8]. All of which are available on the Unity Asset Store [74].

More levels could be added to test ways that the powers can be used and how they need to be adjusted. There also should be more `GameObject`s to interact with to help make puzzles more difficult and interesting. The original design of the scroll was intended to be dynamic and have buttons that represent powers placed on it in the order it was collected. This way, more than just five types of powers could be represented on the scroll, and the maximum amount of powers allowed to collected by an individual player could be how many buttons could fit on the scroll. But to save time the scroll had all powers displayed next to a number.

Chapter 4

A Software Template

4.1 Introduction

4.1.1 Positives with First Attempt

Our first attempt at multi-user Virtual Reality applications allowed several groups to make multi-user VR applications. Our Puzzle Game was one of those applications and it was covered in Chapter 3. There were other applications built with this structure such as a tower defense game [48] and an underground mine evacuation simulator [5]. It was relatively simple to figure out how to synchronize the positions and rotations of `GameObject`s with this structure, and the framework that these projects all used [52] also provided voice chat.

4.1.2 Issues with First Attempt

The framework that we all used [52] was built around Mirror and FizzySteam [54] because UNet was deprecated. There were some issues with this framework that we would like to address. These issues included:

- We did not want the Networking and the voice chat to be tied to Steam. We do not want players to be required to have Steam accounts to play the game.
- The framework only provided a simple head and hands avatar with spheres to represent the heads and hands. We feel that this is an issue since some developers may want to use full-body models and they may need a sample to

figure out how to achieve that.

- The framework also only used glide locomotion with room-scale, and it did not include a sample for teleportation.

- The framework also did not provide a 3rd person view or a video recording option. A video recording option would be useful so that builds of games can use recording without worrying about external software. Video recording is useful for research as we can observe user's behaviours and actions. We can also use the recordings to share research in conferences. Recordings may have some use for teaching as well, as teacher's may be able to give advice or commentary at certain points of a recording during training or teaching.

- Another issue was that students who have used the framework had trouble finding an easy to understand API for Mirror. This adds unnecessary difficulty to create a project.

4.2 Design of the Template

With the first attemprt in mind (both the positives and issues), we set out to design a template for multi-user VR applications. In following the standard software engineering practice we set out to put together a list of Functional and Non-Functional requirements. These types of requirements are described in detail in Sommerville [65]

For this template we need voice chat. This is because other forms of communication may be too difficult when a user has a headset on and two controllers in their hands. Using text may be too cumbersome since the user will have to accurately choose letters from a menu in VR. We also need object synchronization/ownership so that objects are properly synchronized on all clients. A sample for joining games is necessary to allow developers to easily begin developing a game, and to code connecting players. We also want to provide samples for avatar representation including just the the head and hands and using a body model. And finally we need to include

recording (audio and video). This collection of Functional Requirements can be seen in Table 4.1.

Functional Requirements		
Req ID	Priority	
FR1	1	The Template shall allow users to connect to each other with at least two computers
FR2	1	The Template shall allow users to see each other's avatars when their games are connected
FR3	1	The Template shall allow users to send information about one or more objects' position, rotation, and scale
FR4	1	The Template shall provide user representation as head and hands
FR5	1	The Template shall allow users to use and learn from an example game
FR6	1	The Template shall provide voice chat
FR7	1	The Template shall provide a 3rd person mode with video and audio capture
FR8	2	The Template shall allow users to build functioning executables
FR9	2	The Template shall provide a sample user Interface for joining games
FR10	2	The Template shall provide a 3d model with arm IK scripts
FR13	3	The Template shall provide a user interface for joining games that can be accessed with either the HMD and or the Desktop Display.
FR14	3	The Template shall include two samples of locomotion
FR15	3	The Template shall include a Ping Pong Sample Game
FR16	3	The Template shall provide scaling of full body models for different sized people

Table 4.1: The functional requirements for the framework. The priority levels indicate how important the functionality is

For the Non-functional requirements, as seen in Table 4.2, we have that users should be able to use networking software and a game engine that has easy to read documentation to ensure that there is not any added difficulty for understanding how to implement their project ideas. We wish to allow users to maximize the amount of traffic they can use, because we do not wan to limit their testing and require them to keep track of the amount of traffic they are using. We want the template to be reasonably simple to understand as we do not want to add any extra inconvenience to

Non-Functional Requirements		
Req ID	Priority	
NFR1	1	The template will use networking software and a game engine that has easy to understand documentation
NFR2	1	The networking subsystem shall not be tied to Steam
NFR3	1	The Template shall make sure that the software works with a reasonably new version of Unity
NFR4	1	The Template shall maximize the amount of traffic users can use
NFR5	1	The Template shall be designed in a way that assumes that users have never used Unity or VR
NFR6	2	The Template shall ensure that the interface is simple to use
NFR7	2	The Template shall ensure that games made using the framework can connect to different locations on campus
NFR8	3	The Template shall reduce the monetary costs of the framework as much as possible. With the exception that the expensive software is already available.

Table 4.2: The Non-functional requirements for the framework. The priority levels indicate how important each constraint is.

the developer using the template. We wish to ensure that applications using our on premise server are able to at least be able to be used campus-wide because we want students to be able to perform user studies in any building around campus.

In order to create this template, we need subsystems for the following: Networking, a game engine, and Virtual Reality. These are illustrated in Figure 4.1. For the game engine, we will use Unity, for Networking we will use Photon and Photon Voice2 with an on premise server, and for Virtual Reality we will use an HTC Vive and the SteamVR Plugin [75]. In order to record videos we used Video Capture by RockVR [56].

4.3 Implementation of the Template

As previously mentioned in Chapter 2, two of the networking options available with Photon Unity Networking are: using an on premise server or using cloud data. In-

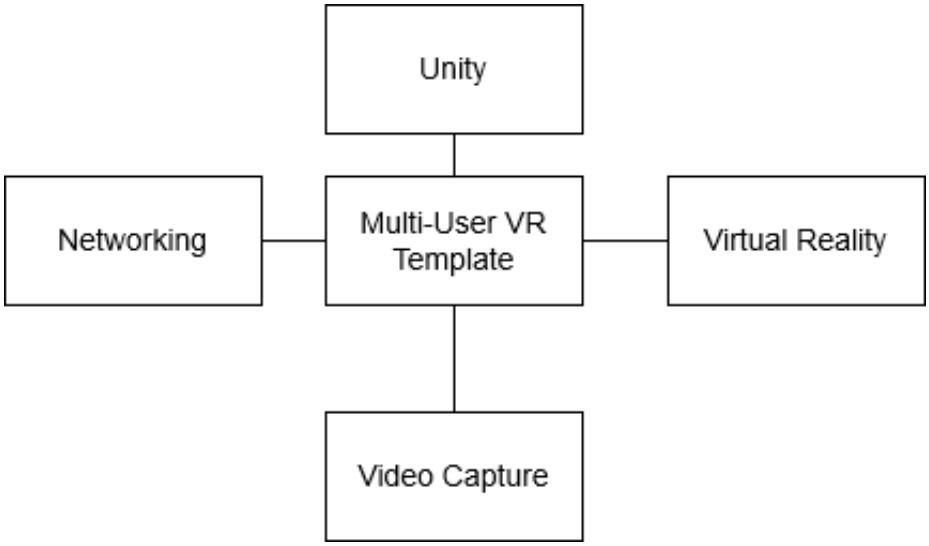

Figure 4.1: Subsystem Diagram for a Multi-User VR Template

structions for how to set up the on premise server or for using cloud data can be found on the Photon website. On this website, more products can be found.

4.3.1 PhotonServerSettings

Figure 4.2 shows the server settings that determines how the application will connect. For our purposes we will enter the IP address as the Server value, and the Port will be 5055. More information can be found online in Photon's documentation.

4.3.2 Scene Set-Up

In the case that we want players to join a room immediately, we use scripts in the scene to automatically spawn players. We would attach `ConnectAndJoinRandom` and `CharacterInsantiation` from Photon Voice's sample scene for using voice chat called `DemoVoicePun` to an empty `GameObject`. `ConnectAndJoinRandom` connects us to a room, while `CharacterInsantiation` spawns the character. The `Character Insantiation` script can be modified to handle a different number of players. It initially chooses an array of four prefabs, but we can change this by changing the size of the array, and the values in the Inspector. We would also need to make sure

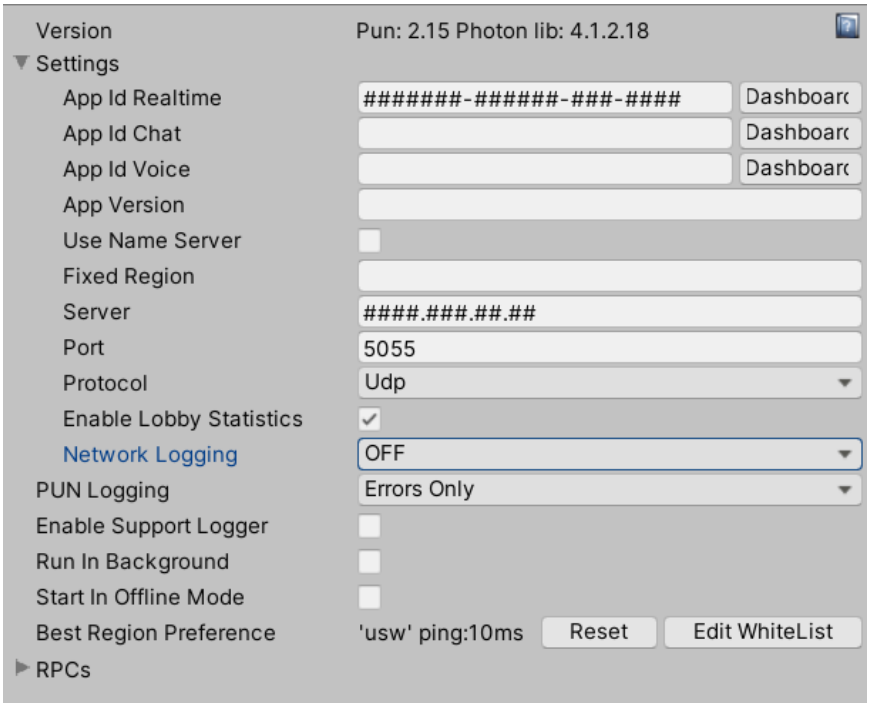

Figure 4.2: The Photon Server Settings. We can use this to choose to use cloud data or to use an on premise server

that the code will not attempt to use an index outside of the range of indices when accessing the array. The `SoundsForJoinAndLeave` script can be used to notify users when a user joins or leaves the room. In order to easily set-up a room, we can use the initial parameters used within the `DemoVoicePunScene`. We would just need to drag the empty `GameObject`, named PUN, into our asset folder to create a Prefab, and move that Prefab into a new scene.

For voice chat, we can use an empty `GameObject` with the following of Photon Voice's components: `Recorder` and `PhotonVoiceNetwork`. Our player prefabs should have their Photon Voice component's attached as well. The recorder has some public variables like `TransmitEnabled` where we transmit the voice if it is set to true, otherwise we do not transmit the voice. It also has parameters for sampling rate, bitrate, and more. For our template we prefer to have voice detection enabled so that users do not need to press a button in order to communicate with another person. The parameters we are currently using for the voice components that start within the

scene can be seen in Figure 4.3. In order to have voice chat, we will also need to add Voice Chat components to the player Prefabs we will be spawning. We explain how to do this next in Subsection 4.3.3.

4.3.3 PhotonView and Synchronization

In order to synchronize the positions, rotations, and/or scale of `GameObject`s, we can use attach a `GameObeject` with a `PhotonView` component as well as a `Photon TransformView` component. It is important to make sure that the `TransformView` is set as one of the observed components in the `PhotonView` component. If we want to synchronize the position of a game object with a Rigidbody component, we should use a `PhotonRigidBodyView`. This is because the velocity on one client may not necessarily be the same as on another. For example if a player picks up an object and sets the velocity of the object to zero, we want the other client to also see it as zero. Otherwise the object may act like it has velocity and not stay in the position. Both the transform view and the `RigidbodyView` can be attached to the same `GameObject`. Photon's older transform view component gives options for interpolation which can smooth out the path of the `GameObject` if it appears to jittery on clients other than the client that has authority.

`PhotonVoiceView` can be used to allow users to use voice chat. In order to properly set this up we must have the proper components attached. See Figure 4.4 to see our components and parameters. `FollowActive` is one of our own scripts that makes the `GameObject` it is attached to to follow the camera that the user is currently using. We do this along with synchronizing the position to make sure that the sound is correctly based on the position of the `GameObject`. Unity's sound system allows it to simulate 3D sound, so a far away sound will sound farther away. We can choose to modify the Audio Source's parameter to fit our needs for the voice chat.

Remote procedural calls (RPC) can be used to synchronize things such as materials. If one user changes the color or material they are using on a model, we may want to tell all the other clients to change that color as well. An RPC essentially is

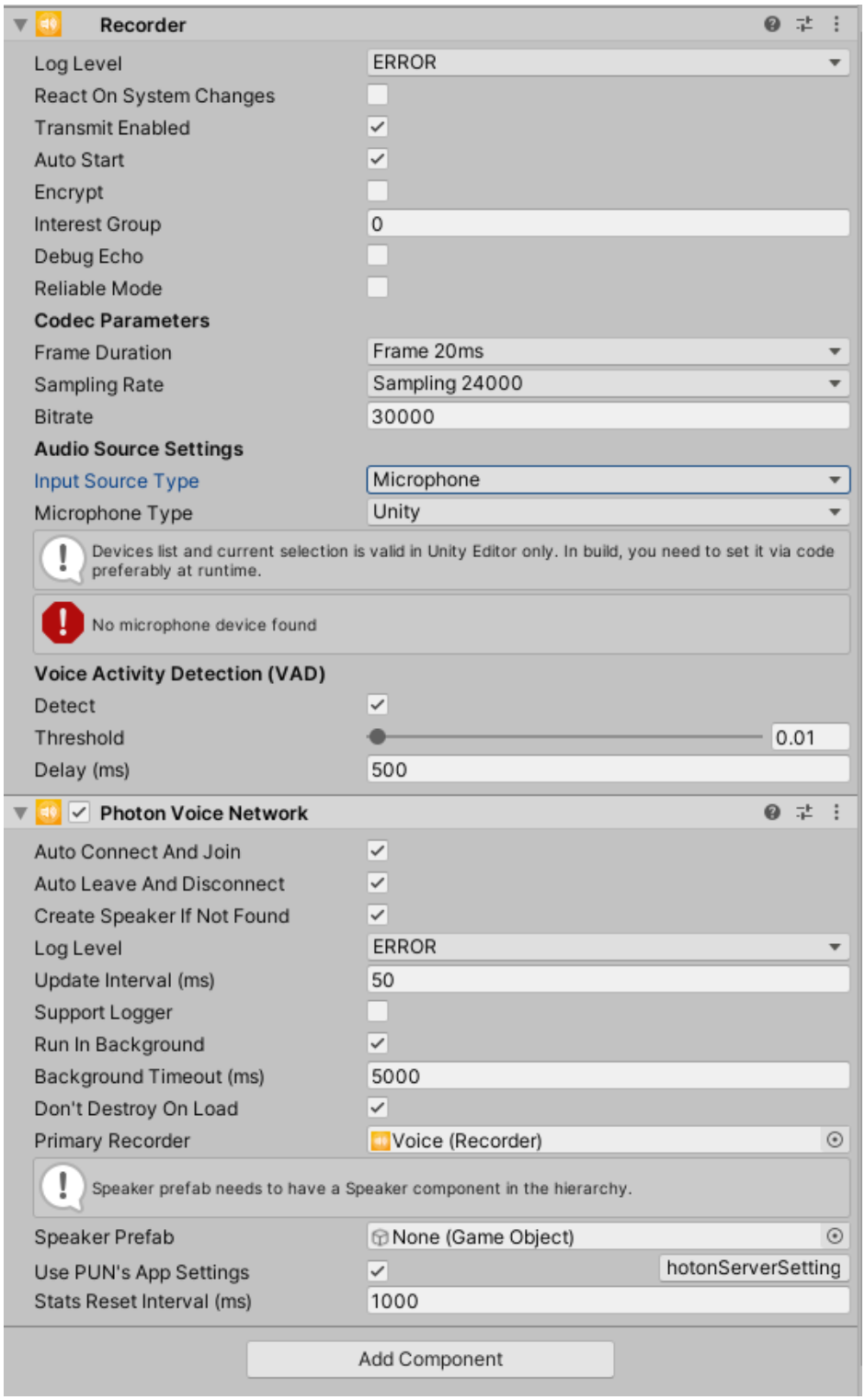

Figure 4.3: The empty `GameObject` with voice related scripts attached.

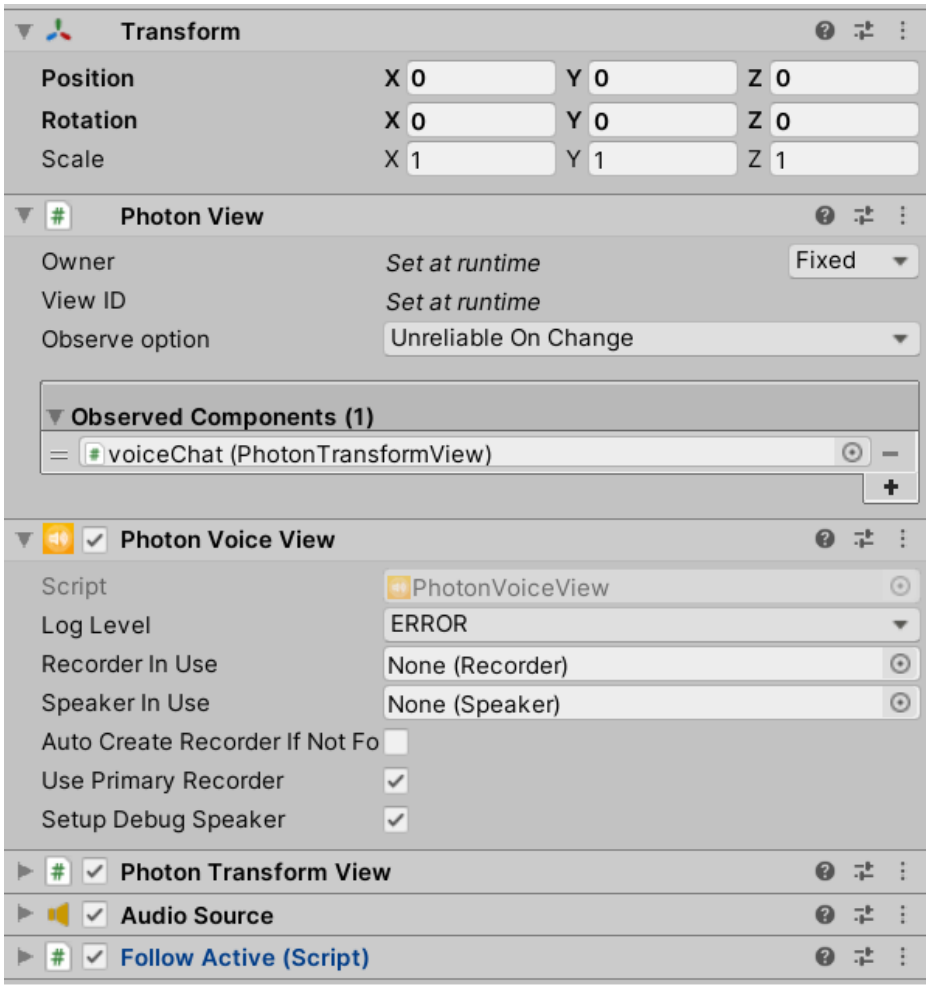

Figure 4.4: The components we used for voice chat attached to our player Prefab

telling other clients to execute a function. When an RPC is called, the function will be called on the copies of the `GameObject` that called it.

There are several targets we can send RPCs to. We can send them to every client, which includes ourself, all other clients, and we can buffer RPCs [18]. We want to buffer RPCs just in case a new client joins the game, so they can have all the changes that the other players have made on their end. For example if a player changes their models color to red and sends an RPC to all other clients that they changed their model to red, a client that joins after that RPC was sent will see the player as their default color rather than red. So we can choose to buffer RPCs to ensure that the correct changes are made when the new player joins. Parameters can be passed through RPCs, however there are some limitations. For example it is not possible to send a `GameObject` as a parameter, but it is possible to send a float as a parameter.

4.3.4 Authority

Photon uses a concept known as authority. We need to base our synchronization on at least one user, in other networking models it may be the server. So when a user has authority over an object, all other clients try to copy the properties have that users copy of the object. Photon provides the ability to swap authority of a `GameObject` using the `RequestOwnership()` from the `PhotonView` that is attached the `GameObject`. Though we must set the `PhotonView` as "Takeover" so that users can take over the authority of the `GameObject`.

As seen in Listing 4.1, we access the the `PhotonView` of the `GameObject` we wish to take control over. Then we request for ownership and use an RPC to ensure that it is detached from a user that may be holding it. We have not yet tested if it is necessary to use the RPC to detach the object. But based on testing it appears to have worked.

The `HeldBy` function, as seen in 4.2, is called when a new object is attached to the user's hand. This function tells the local copy of the held object the person who

Listing 4.1: Code demonstrating swapping authority and allowing a user to take control of an object another users is holding

```
void Update() {
            GameObject current = hand.currentAttachedObject;
            if (current != null && prevAttached != current) {
                HeldBy hB = current.GetComponent<HeldBy>();
                hB.authority = this;
                PhotonView tempView = current.GetComponent<PhotonView>();
                if (!tempView.IsMine) {
                    tempView.RequestOwnership();
                    hB.photonView.RPC("ChangeHolder", RpcTarget.Others);
                    prevAttached = current;
                }
            }
            if(hand.currentAttachedObject == null) {
                prevAttached = null;
            }
        }
public void release(GameObject obj) {
            hand.DetachObject(obj);
        }
```

is currently holding it. The idea behind this code is that we want to tell the user who previously held the object to let it go. So we call a function on the held object that tells all copies of the object on all clients to see who was holding the ball last, and tell them to let the ball go.

4.3.5 Joining and Creating Rooms

Joining and creating rooms is actually a simple task. We can choose to have users connect to rooms immediately or to allow them to choose to connect to a room. It depends on the scripts, and what is contained in them.

If we want an application to immediately have users connect to a room and join a server, we can base our scene set up based on Photon Voice 2's sample scene for voice chat. This includes an empty `GameObject` called `PUN`. `PUN` has two important scripts attached: `ConnectAndJoinRandom` and `CharacterInstantiation`.

For joining and creating rooms we used Photon's `ConnectAndJoinRandom` in or-

Listing 4.2: Code essentially tells us who last held the object

```
public class HeldBy : MonoBehaviourPun
{
    public Valve.VR.AuthoritySwap authority;
    [PunRPC]
    public void ChangeHolder() {
        if(authority != null) {
            authority.release(gameObject);
            authority = null;
        }
    }
}
```

der to understand how to code it. So there may be some similarities between the two. To join a room, we first connect using: `PhotonNetwork.ConnectUsingSettings()`, and then we join the lobby using: `PhotonNetwork.JoinLobby()` within the `OnConnectedToMaster()` call back function. The `OnConnectedToMaster()` function is called when we have connected to the master using `PhotonNetwork.ConnectUsing Settings()` We can use the `OnRoomListUpdate()` call back function to update a list of rooms. Listing 4.3 shows how we have done this. We receive the `roomInfo` list. It gives us some information on each available room which includes the max players allowed, the amount of players that are in the room, and the room name. We can use these strings and display them to a user that wishes to join a room. In this case they are being added into a drop-down menu. If a room is full the user will probably not join it. The room name should identify the room so that if another user wishes to join, they can. Once we join a game and the scene is loaded, we can use scripts within that scene to spawn the player.

To create a room, we may want to fill in a couple of parameters. One is the max players allowed, another is the scene. We could allow players to name the rooms they create on their own, but we decided to just name them based on the scene name chosen, and add a number at the end of the scene name to make sure that the scene name is unique.

Listing 4.3: Code For listing room names

```
public override void OnRoomListUpdate(List<RoomInfo> roomList) {
.
.
.
System.Collections.Generic.List<Dropdown.OptionData> optionData =
    new System.Collections.Generic.List<Dropdown.OptionData>();
            int i = 0;
            roomInfo = new RoomInfo[roomList.Count];
            foreach (RoomInfo room in roomList)
                roomInfo[i] = room;
                i++;
                optionData.Add(new Dropdown.OptionData(room.Name +
                   " " + room.PlayerCount + "/" + room.MaxPlayers) );
            }
            dropdown.options = optionData;
        }
.
.
.
}
```

In our implementation we create the room in the `roomListUpdate` function. We chose to do this so we could receive a list of the current rooms so that we can use a unique room name. First we set some room options including the max players allowed and the scene choice. Then we create the room with `PhotonNetwork.CreateRoom`.

If the application is meant to be used by both VR and non-VR users an application using this template may wish to have both 2-D and 3-D interfaces. Figure 4.5 shows a 2-D menu for creating a room which allows the user to chose a scene and the max amount of players allowed.

Figure 4.6 shows a 2-D menu for joining a room. This menu contains a dropdown of available rooms. The components used to accomplish this were from Unity's UI assets.

Figure 4.7 shows a 3-D menu. It uses parenting to move the buttons, and it's position is made to only be able to be moved toward or away from the menu. The menu is a set as a child of one of the user's arms. This is to allow the user to This

Listing 4.4: Code For Creating Rooms

```
public override void OnRoomListUpdate(List<RoomInfo> roomList) {
 public void CreateRoomButton()
    {
        maxPlayers = int.Parse(maxPlayerAmt.text);
        sceneChoice1 = sceneChoice.text;
        PhotonNetwork.ConnectUsingSettings();
        blnCreatingRoom = true;
    }
    public override void OnRoomListUpdate(List<RoomInfo> roomList)
    {
    ...
    if (blnCreatingRoom)
        {
            int i = 1;
            foreach (RoomInfo room in roomList)
            {
                if (room.Name.Substring(0, sceneChoice1.Length) ==
                    sceneChoice1)
                {
                    i++;
                }
            }
            blnCreatingRoom = false; blnCreatingRoom = false;
            RoomOptions options = new RoomOptions { MaxPlayers = (byte)
                maxPlayers };

            PhotonNetwork.CreateRoom(sceneChoice1 + i, options, null);
            SceneManager.LoadScene(sceneChoice1);

            Destroy(transform.root.gameObject); //destroy the player object
                that uses this
        }
        ...
    }
```

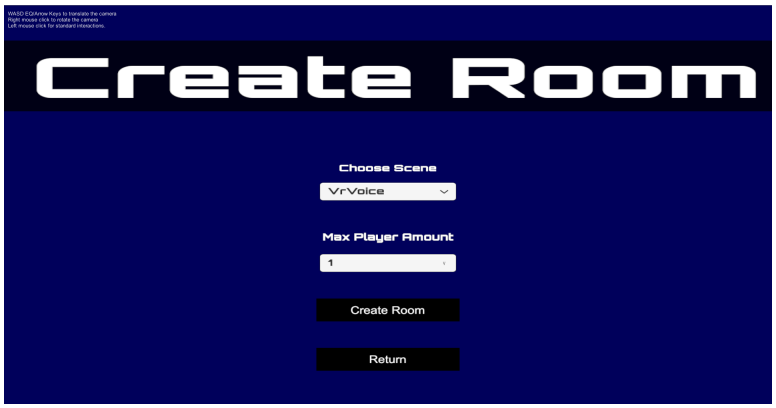

Figure 4.5: 2-D menu for creating a room

Figure 4.6: 2-D Menu for joining a room, with an instance of the game showing that room

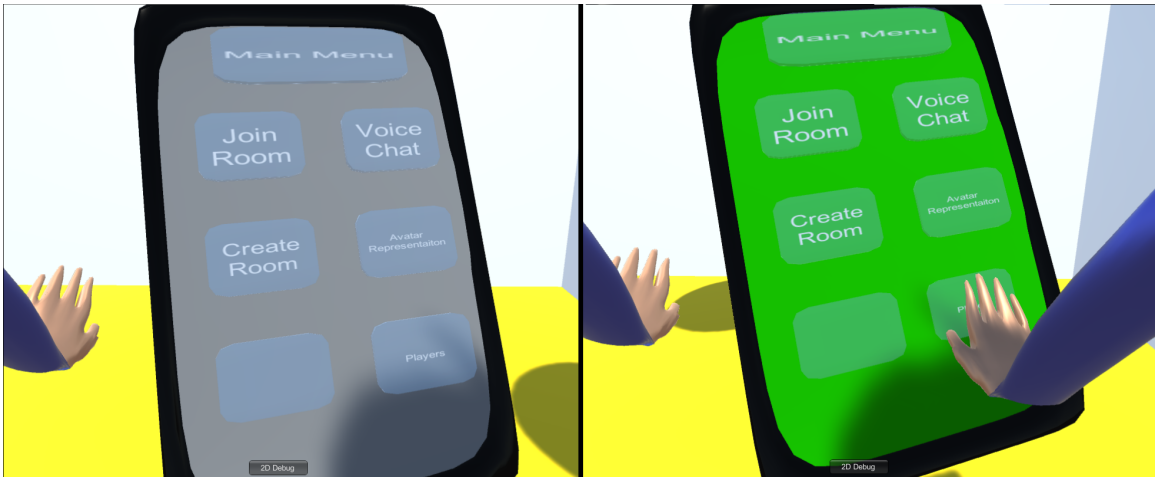

Figure 4.7: 3-D menu. When the the menu and its sub menu are enabled, the grow and translate away from the menu. Uses `GameObjects` for options and when a button collides with the gray area, it temporarily turns green

menu allows users to change rooms without having to take off their headset. We had a limited amount of room so the menu only currently showed five rooms to join when selecting a room.

4.3.6 Enabling Components

Because we are spawning copies of a player to multiple users, we will want to have certain components enabled on copies of avatars from other clients and certain components disabled on them. For example, there can be only one Audio Listener in the scene at once. So if a copy of a client's avatar has an Audio Listener, and the local user has an Audio Listener, we would have to disable the Audio Listener from the other client. Following an idea from [52], we have a script attached to the root of our avatars that allows us to disable certain components.

Figure 4.8 shows an example of script components to disable. We want to disable other cameras, since we want the local player to use their own camera. In order to ensure that the local player does not control the clients, we should also disable the hand related scripts from the SteamVR Plugin. Within our script, we check who has authority over the avatar with `PhotonView.isMine` and disable it by setting its

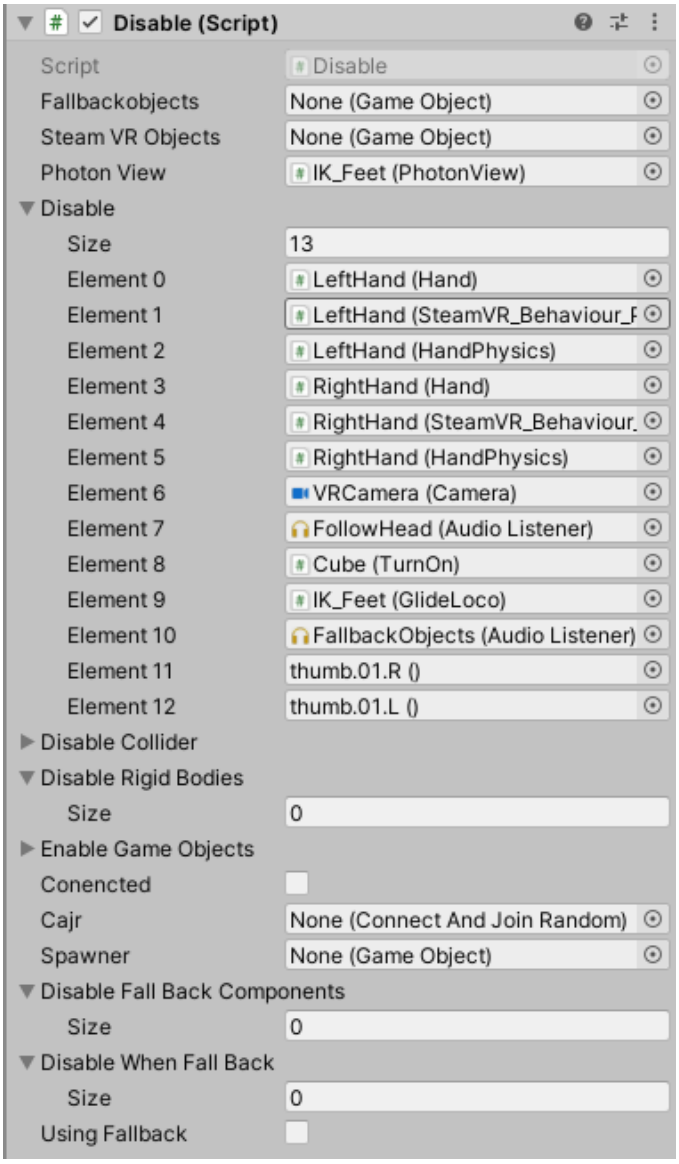

Figure 4.8: Array of Behavior scripts to disable

enabled value to false, if `PhotonView.isMine` is false. It's also possible to disable a script if it is not owned by the local player within the `Start()` function within that Script so long as the script has access to the a PhotonView script that is attached to the local player. We have this component attached to the root of our object.

In this iteration we disable components, but we plan on changing this to only enable components instead, so other developers have an easier time developing with the software.

4.3.7 Locomotion Samples

There are arguably an infinite number of implementations of locomotion possible for users. But we will just go over two. One is teleportation, while the other is touchpad movement.

A simple locomotion method to use is locomotion based on the touchpad. It is arguably unnatural as users press a button to move rather than move their body. There are many ways to implement this. One example is that the position of a user's thumb on the touchpad can be used to determine which direction users want to move in. The direction of the headset can be used as well. Another option may be to use controllers to point into the direction the user wishes to move.

Listing 4.5 is an example of how to use the camera direction to use make the character move. The script is attached to the root of the avatar. Collisions can be used to prevent the user from moving through obstacles. The root collider can have it's center be set at the center of the avatar in the update function so that it is at the correct location for collisions, otherwise the collider may be separated

Listing 4.5: Code for movement based on camera direction

```
transform.Translate( Vector3.Normalize(new
    Vector3(cam.forward.x,0,cam.forward.z) )* Time.deltaTime*3);
    //move in direction of camera
    //use time.deltatime to ensure the distance moved
    //does not depend on the speed of the computer
```

from the camera since the camera's local position can change and not be at the same position as the root object. Since users will move with the controller rather than in real life, they may feel sick due to the mismatch in perception. One positive of this method of locomotion is that it looks more natural to other players in multiplayer than teleportation.

Because of the way our player prefab is set up, the camera can have a different position than its root position. We found that we were unable to move the camera position when a VR headset was in use. So in order to teleport the user we used the root transform. Since the root transform and the camera, which is lower in the hierarchy than the root, can be in different positions, teleportation where we just move the root transform's position into a new position may cause undesired outcomes. This is because the user will not teleport to the exact location that they intend to teleport to (the root object will). In order to deal with this, we can move the root object in way that the camera is placed in the position we want it to be placed. An example of this is in the Listing 4.6.

Listing 4.6: Code for Teleporting to the a position where we expect the camera to be when the user is standing up straight

```
playerTransform.position = new Vector3(teleportLocation.position.x,
    teleportLocation.position.y - deltaY, teleportLocation.position.z);

playerTransform.position = 2 * playerTransform.position - cam.position;
```

There may be other ways to do this but, this is how we are currently handling teleportation. In the code in Listing 4.6 we move the root to a position. Then we move the root Transform to a position that moves the child Transform to the desired location. We will also want to take the VR camera's local position into consideration since if the user bending when they teleport, their camera position when the teleport may not be in the position we expect. We can't expect the user to be standing up when teleporting.

4.4 Network Prefab

In Figure 4.9 we see an example of how a player Prefab may look. `SteamVRObjects` contain `GameObjects` that represents the user's hands and head. When VR is not in use, the `NoSteamVRFallbackObjects` become enabled as the `SteamVRObjects` become disabled. `FemaleDoc1` is just the model of the avatar that we are using to represent the user. Depending on the settings we provide for it, it will contain the bones for the avatar. This can be useful as we may sometimes wish to add scripts and/or colliders to the bones. `foot1` and `foot2` are targets for IK for the model's feet. It will help us find the rotations for the thigh and the lower leg. In our case, we are just using them to allow the user's knees to bend.

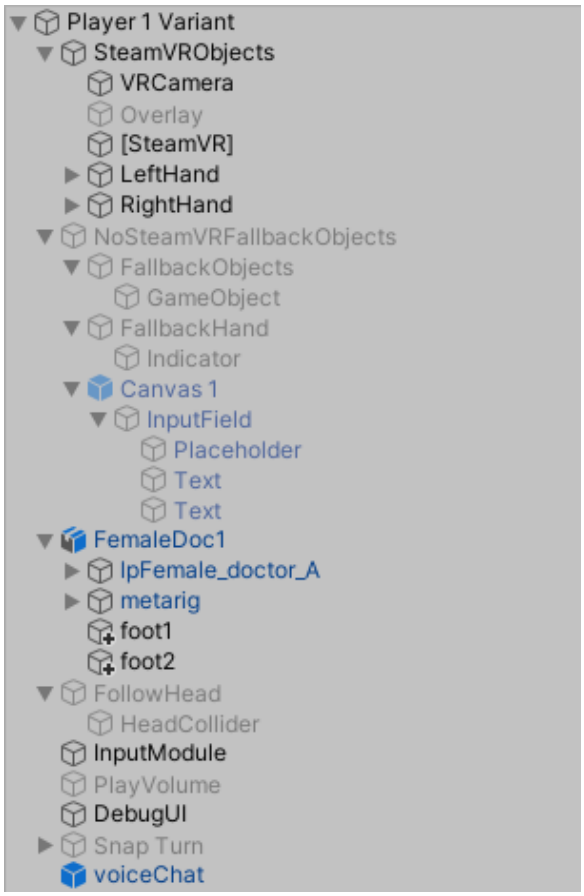

Figure 4.9: What the Hierarchy of a prefab with a body model and 3rd person may look like

4.5 Body Models

There may be discrepancies between the avatar and the human. Arm length, leg length, height, etc might not match the model. We only have 3 tracked points for matching the model's movements with the user's. IK is an option to estimate the rotations of the arms. Unity thankfully, provides easy to use functions for this. Trying to match the model with the user may require manual avatar customization or extra hardware and software that utilizes computer vision techniques. It may be possible to use images of a person to attempt to make an accurate 3-D model representation of the person. We also do not know if it users will wish to be another person and "escape" or to be themselves. The effect of Avatars on immersion is also a concern. Research may still need to be done to understand the psychological influences of user avatars on the user.

4.5.1 Blender

In Figures 4.10 and 4.11 we see a way to easily use a what we will call a skeleton, but it is also possible to make a skeleton from scratch or edit one of the samples. Skeletons can be modified in editor mode. We can assign the skeleton model, while using the option to assign automatic weights, to assign the skeleton to the model.

In Figure 4.12 We can make modifications to how much a bone affects the mesh when it is translated, rotated, or it changes scale. We are uncertain how much these weights are affected Unity's import options, but at the moment we believe it plays some role in it.

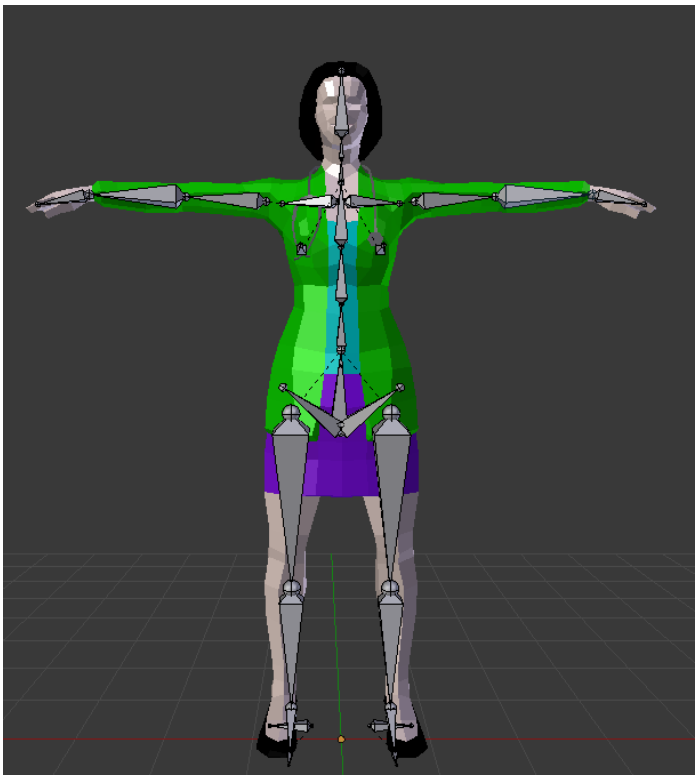

Figure 4.10: The bones allow us to move part of the mesh. If the bone is moved, scaled, or rotated, the assigned part of the mesh will be moved, scaled or rotated [72]

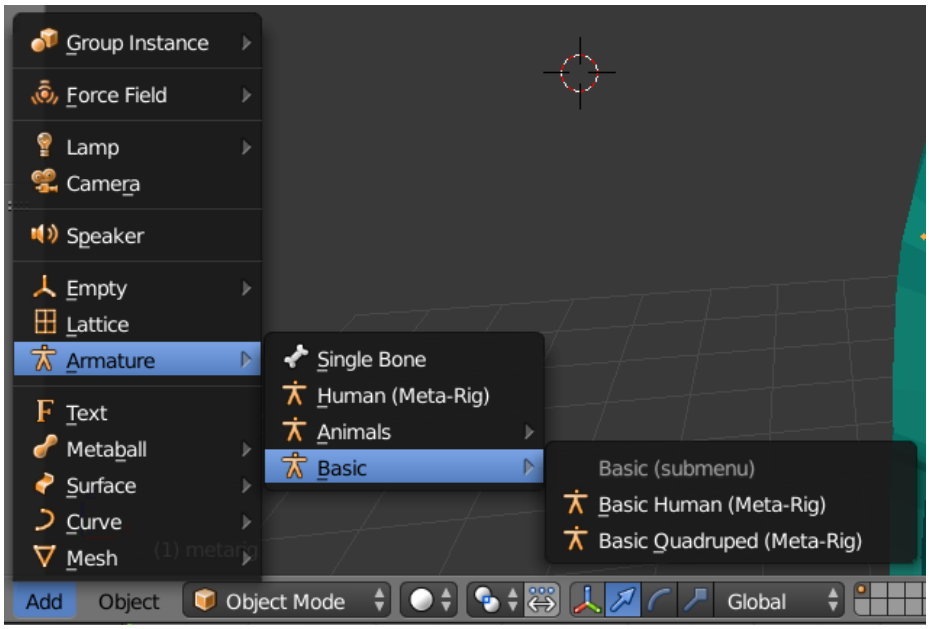

Figure 4.11: With the Rigify Add-on we can easily create a humanoid skeleton

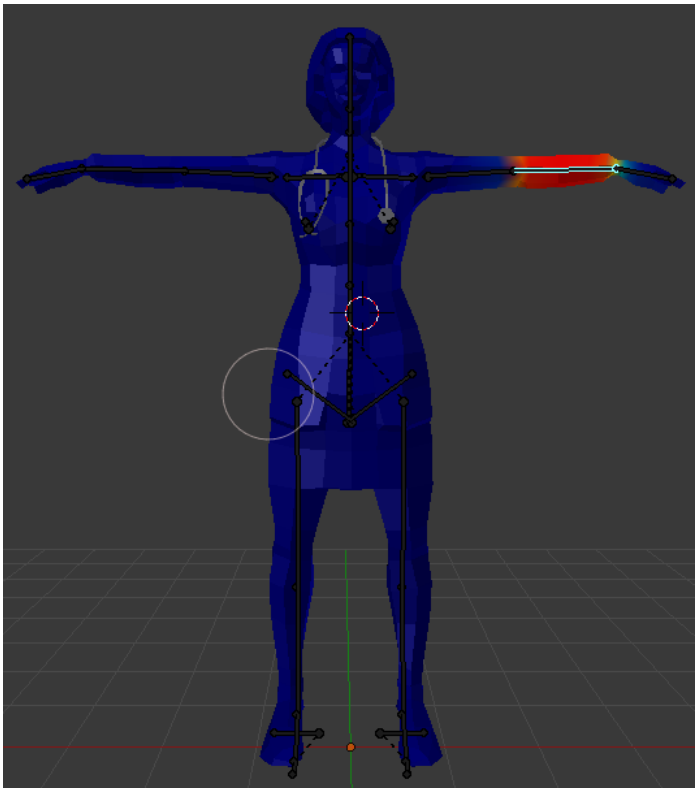

Figure 4.12: Weight painting allows us to make adjustments to how the bones will move the mesh

4.5.2 Animation Settings

For our implementation, we used a looping animation where the character model is essentially in the same pose. We have the IK pass option checked as true in the Animator. The Animator can be seen in Figure 4.13. The Animator allows us to control the animation states a model is in and how it transitions into other animation states. For example, if we want to show that a model is falling, we can check use a Boolean to tell the animator that we want the model to play its falling animation clip. But when we are not falling, we would want to transition back into walking or an idle animation. Since we want to give users control over there arms in VR we probably will not want to use animation states to change the pose of the arms. If we give the user control over the legs, we would want to match the user's feet positions to the model's feet positions. In that case we would not want to change the pose of the feet or legs either. We would want those animations to be live.

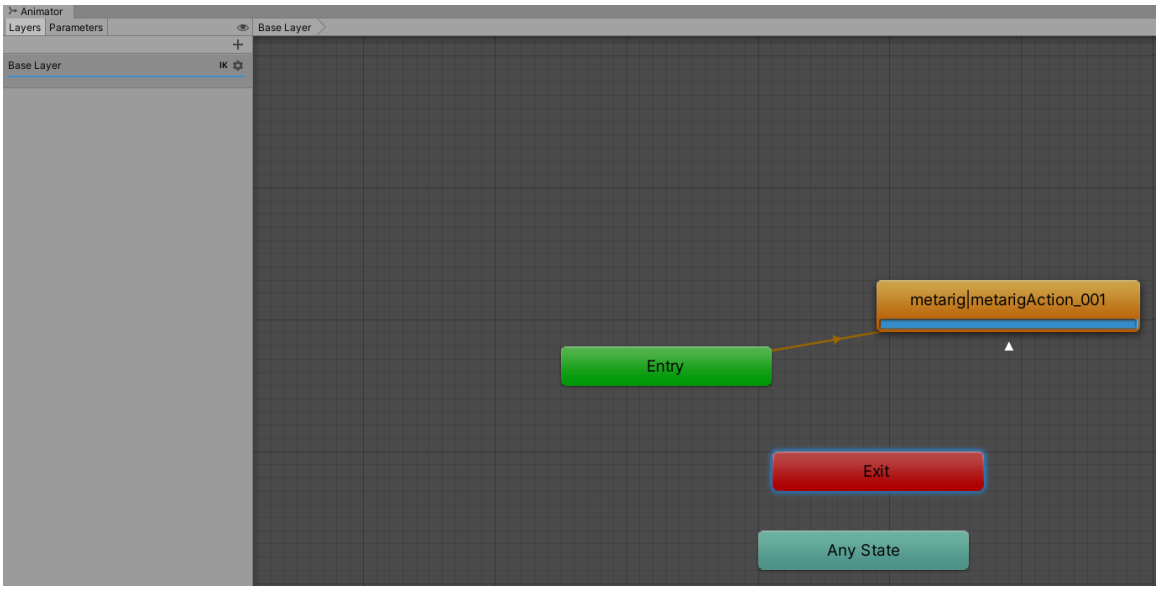

Figure 4.13: Animator that shows us the animation state a model is currently in as well as the transitions and between the states

The settings in Figures 4.14 and 4.15 were what we used. We were able to have Unity's IK working with these settings. For our animation clip, we have a looping pose where the model essential remains in the same pose without moving.

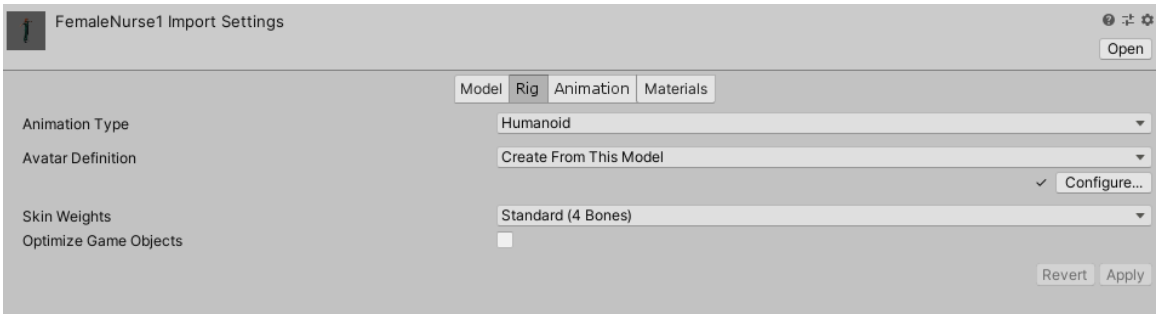

Figure 4.14: These are the import settings we used for the Rig, based on the instructions for using IK from Unity's documentation

4.5.3 Scripts

In order to have the head rotate the same as the user's head rotate, we match the rotation of the head using the Euler angles of the HMD. We attached the script show in the Listing 4.7 to one of the model's spine bones.

In Listing 4.8 we what happens during calibration when the user first joins a room. This is in the `LateUpdate()` function so we are updating values every frame. We decided to place the user's head at the head of the model. We did this rather than attempting to change the scale of the models. The way we did this is we have a `charactercontroller` and we that try to push down the entire player prefab downwards when the user starts playing. The `charactercontroller` will not move the player downwards if it is colliding with a floor. We have the center of the character controller follow the model. The model's position is always made to match the position of the camera. So if the user happens to be crouching and then stands up straight, the model and camera will move up, but the root `GameObject` will be pushed down. When the player stands up straight and then drop's their head's position, the camera will remain at the position of the model's head. This way the user's head will be at the model's head. Also, the We will call this process calibration. We also position the model slightly behind the camera, so the user does not see inside the model.

In the start function we have we call `Invoke()` which will call `nowAdjusted()`

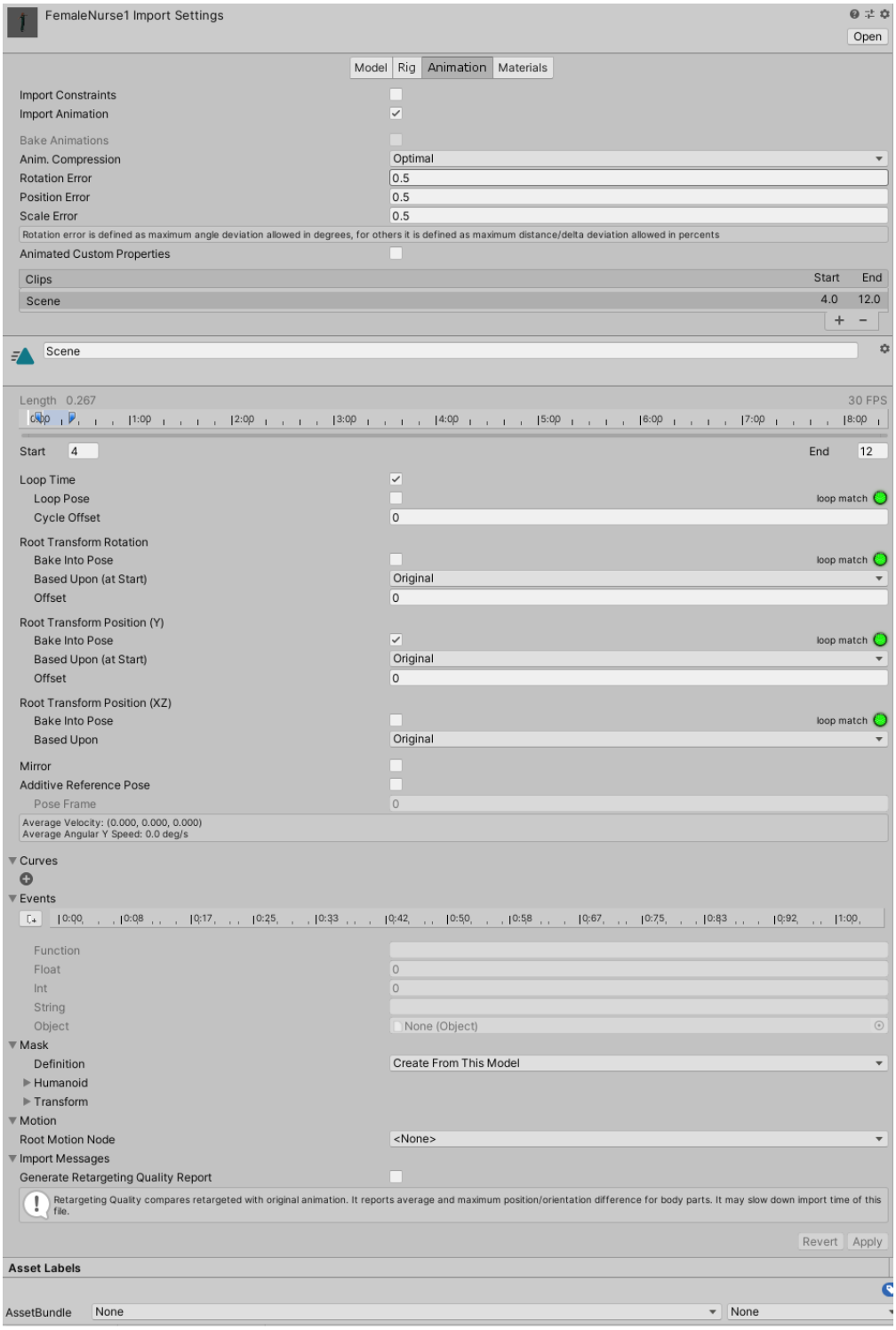

Figure 4.15: These are the import settings we used for the animation

Listing 4.7: Code for rotating the model's head based on the user's head

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
/**
 * HeadRotate is used specifically for full body models. Attach it to the
    prefab of the model.
 * It uses camera rotation for the rotation of whatever bone controls the
    head's animation
 * It rotates the players head, and makes the models less static looking
 * This is meant for head and hands only, not using any extra trackers.
 * Attach this to the the correct bone to rotate
 */
public class HeadRotate : MonoBehaviour
{
    /*The transform of the camera it contains position, rotation, and
       scale, Transforms are from Unity*/
    public Transform cam;


    // LateUpdate is called once per frame after update
    void LateUpdate()
    {
        transform.eulerAngles = new Vector3(cam.eulerAngles.x,
            cam.eulerAngles.y, cam.eulerAngles.z);//make the rotation the
            same as the camera rotation
    }
}
```

Listing 4.8: Code demonstrating the calibration state

```
void LateUpdate()
    {
        transform.position.y + fixedPos, controller.center.z);
        if (!adjusted)
        {
            gameObject.transform.position = new Vector3(camera.position.x,
                camera.position.y + cameraYOffset, camera.position.z) +
                transform.forward * ahead;
            gameObject.transform.eulerAngles = new
                Vector3(transform.eulerAngles.x, camera.eulerAngles.y - 90 +
                angleAddition, transform.eulerAngles.z);//this makes the
                model rotate to face the direction of
        }
    }
```

(Listing 4.9) after a number of seconds. We do this so we know where the camera location was when the user was standing up straight. We also disable the `charactercontroller` at this point. We set the `adjusted` Boolean variable to `True` so that we know that we are no longer calibrating.

We may not want the model's head to always be at the same position. This is because we may want the model to crouch when the user crouches. To do this, we can disable the `charactercontroller` and have the model continue to follow the location of the camera, but not move higher than its height after calibration. But this comes with a problem, the model's feet will sink into the ground. The solution for this would

Listing 4.9: Code for updating values when we switch states

```
public void nowAdjusted() {
        adjusted = true;
        if (m_PhotonView.IsMine)
        {
                adjustedY = camera.localPosition.y;
                adjustedY2 = camera.position.y;
                controller.enabled = false;
                adjusted = true;
        }
```

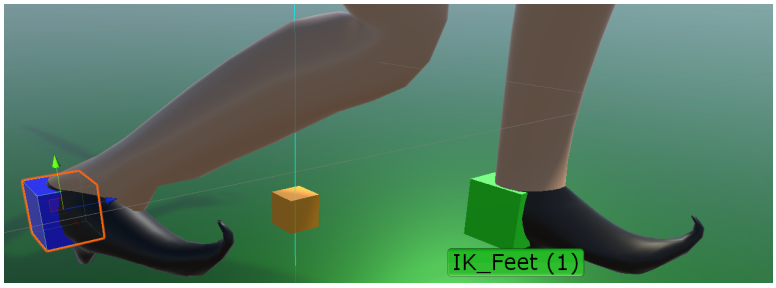

Figure 4.16: Attempt at using tells from the VR camera to guess where the user's feet are. The cubes are IK targets for the feet. They change color based on the state they were in for debugging purposes.

be to either use IK for the feet, or to rotate the legs in the `LateUpdate()` function. We chose to use IK for the feet and set the targets at the height of the ground level, but they are still in a position where they would be expected. This way when the user crouches, their feet will try to move to their IK targets. We do have another problem however. If the user rotates their neck to look down, will the model begin to bend it's knees as the user is lowering their heads position. For now we just use a threshold to determine how far the user needs to bend in real life until the model bends down as well. In Listing 4.10 we see the code that allows our model to move up and down in the Y-axis. We use a `findThreshold()` function to determine when the model should be allowed to descend.

4.5.4 Feet Estimation Attempt

We attempt to guess where the user's feet are using only sensors from the headset and the wands. We made the assumption that user's feet will always be the same distance from their headset. We also assumed the headset was always be between both feet. When the user started to move in a direction we choose where the foot target should go based on that position. If the user moved in some direction that is to the right and not to the left, their right foot would be chosen, otherwise their left foot was chosen.

There are subtle movements of the head when a user walks. This is likely because when a human moves their foot out to take a step, their head must move down as

Listing 4.10: Code demonstrating state after calibration

```
private void Update()
    {
        controller.center = new Vector3(transform.localPosition.x,
            (transform.localPosition.y + fixedPos),
            transform.localPosition.z);

        if(adjusted)
        {
            if(camera.position.y >= adjustedY2)
            {
                determinedY = adjustedY2 + cameraYOffset;
            }
            else
            {
                float threshold = findThreshold();
                if(  adjustedY2 - camera.position.y > threshold)
                {
                    determinedY = camera.position.y + cameraYOffset +
                        threshold;
                }
            }
            gameObject.transform.position = new Vector3(camera.position.x,
                determinedY, camera.position.z) + transform.forward * ahead;
            gameObject.transform.eulerAngles = new
                Vector3(transform.eulerAngles.x, camera.eulerAngles.y - 90 +
                angleAddition, transform.eulerAngles.z);
        }
    }
```

the human moves their body down to put their foot on the ground. They may move upwards as they bring their other foot towards the foot they just used to take a step. We tried to use this to tell when the user is switching feet. However, we have had trouble on finding thresholds for when to consider if the user was moving down or up as they switch their feet, or if they were moving for other reasons. For moving the feet we would recommend animating them before hand, and making adjustments to them. It's also possible that our method for choosing which foot to move is too unreliable since if the user moves forward but moves their head to the left while using their right foot, the wrong foot would have been chosen. Though if we did just use pre-made animations the chosen foot would probably not be correct anyways.

One point to note is that we did not use thresholds to wait before guessing where the foot is moving. Since the update function is called in a relatively short amount of time, we may be picking the foot too soon. It may be worth looking into different points saved at runtime before choosing a foot to move. This may cause a brief snap in the animation as the chosen foot will have to change position after the threshold was chosen, but choosing the right foot to move is very important. Cases where the foot is too far away from the user should probably also be considered and it can look very odd for leg to try to be facing towards an IK target that is too far away. The model would essentially look like it's levitating. Another idea is that perhaps a user can train the model to move it's feet as they do.

There are many ways a person can walk and move their feet. The estimation idea would not have been able to replicate every possible case. A user could have moved one of their feet up without walking in a direction, and that would not have been replicated by our attempt at a foot estimation algorithm.

4.6 Controllers and Trackers

Using the standard wands is very simple, one just needs to use a sample prefab that SteamVR provides.

Using the Vive trackers is also easy. One just needs to attach it to the appropriate location in a prefabs hierarchy and then add SteamVR's `SteamVR_TrackedObject` script[75] to it. This script needs to have it's index public variable set to the index of the Vive Tracker.

One issue our current method for using body models is that the back of the model is always straight up. This can look unnatural. Perhaps it may be possible to add a Vive Tracker to the user's torso, by one of their spine bones, so that we can use the positions and the forward directions of the Vive tracker and the Headset to help determine the rotations for the spine bones. We would probably use this Vive tracker for where to place the user avatar in the Virtual Environment. This is because the other parts of the body would be able to move away from the spine. We may guess the rotations of some of the spine bones if Unity does not provide IK for this. But there are IK libraires in the asset store.

If we were to use more trackers for the user, we would have to expect that if we were expecting to make a multiplayer game, we would want other users to have trackers as well, or we would want them to be able to choose how many points of tracking they would need. In a competitive game, we would want to make sure that any additional trackers do not give a player an unfair advantage. It should also be considered that requiring extra trackers may be a disadvantage because it makes the overall product more expensive, it may require more calibration, and it may take more time for users to set up to play a game. However, we do believe it is still worthwhile to look into these options as it will still further our understanding of how we can animate these models and how users react to them.

4.7 Video Recording

4.7.1 Software

While other recording software does exist, we wanted to use software that was present within Unity as opposed to a stand alone recorder. The reason for this is so that we can use one program. We chose to use Video Capture [56] from RockVR, because it was free and it provided video and audio recording.

4.7.2 NoSteamVRFallbackObjects

Using one of SteamVR Plugin's example prefabs, we can use the NoSteamVRFallbackObjects `GameObject`. When an HTC Vive is not presently being used. This `Gameobject` allows the user to move in 2D. We can also use these `GameObject`s to know if we want to display the user avatar or not. We can use Unity's `OnEnable` function to know if the `NoSteamVRFallbackObjects GameObject` has been enabled. Under `FallbackObjects`, a child of `NoSteamVRFallbackObjects`, we have one of the SteamVR Plugin's scripts attached to allow the user to move around the camera. This movement uses the WASD keys and it allows the user to right click rotate the mouse to rotate the camera. The `NoSteamVRFallbackObjects` child, called `FallbackHand` allows users to interact with interactable objects.

4.7.3 Recording Components

Through our testing we found that having multiple active copies of certain components from RockVR will destroy the root `GameObject` that those objects are attached to. To get around this, we only add the components when we confirm that a player prefab that was spawned was controlled by the user. This way we could guarantee that there was only one copy of each component present. Another option may have been to have those components disabled and enable them when we confirm that the prefab is controlled by the user. This would have may have better option.

As seen in Figure 4.17 we have the root component of the `NoSteamVRFallback`

`Objects`. Here we add components: `VideoCaptureUI`, the UI menu for the video capturing, `VideoPlayer`, which is different than Unity's `VideoPlayer`, and `VideoCapture Ctrl`. Unity's `VideoPlayer` allows the playing of videos.

Figure 4.18 shows the first child object of the NoSteamVRFallBackObjects. For this `GameObject`, we add the following components: `AddAudioCapture` adds the `AudioCapture` script component from RockVR. The Camera component attached to this `GameObject` is used for the user to see what is in the scene. `addRecording Components` is the script that adds these components, and adds any values that need to be added to the `GameObject`'s components. For example, `VideoCaptureCtrl` uses the `VideoCapture` Component in an array and an Audio Capture component that are attached to different `GameObject`s in the Hierarchy.

Finally we have the `GameObject` named GameObject that we added as a child to the FallbackObjects. The `AddVideoCapture` script adds RockVR's `VideoCapture` Script to the `GameObject` Component. The Camera component attached to this `GameObject` is used for video capture.

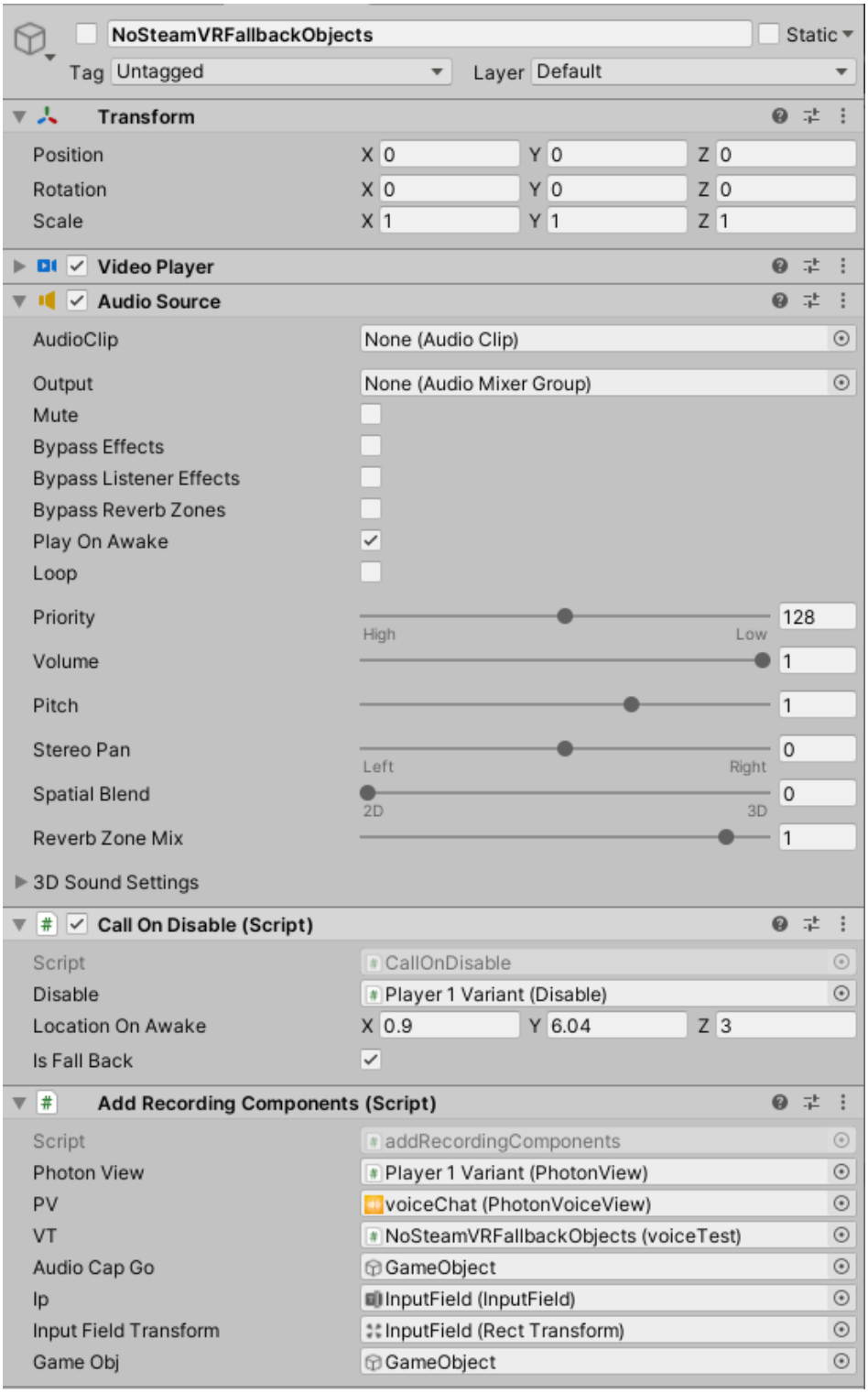

Figure 4.17: The root of the NoSteamVRFallbackObjects

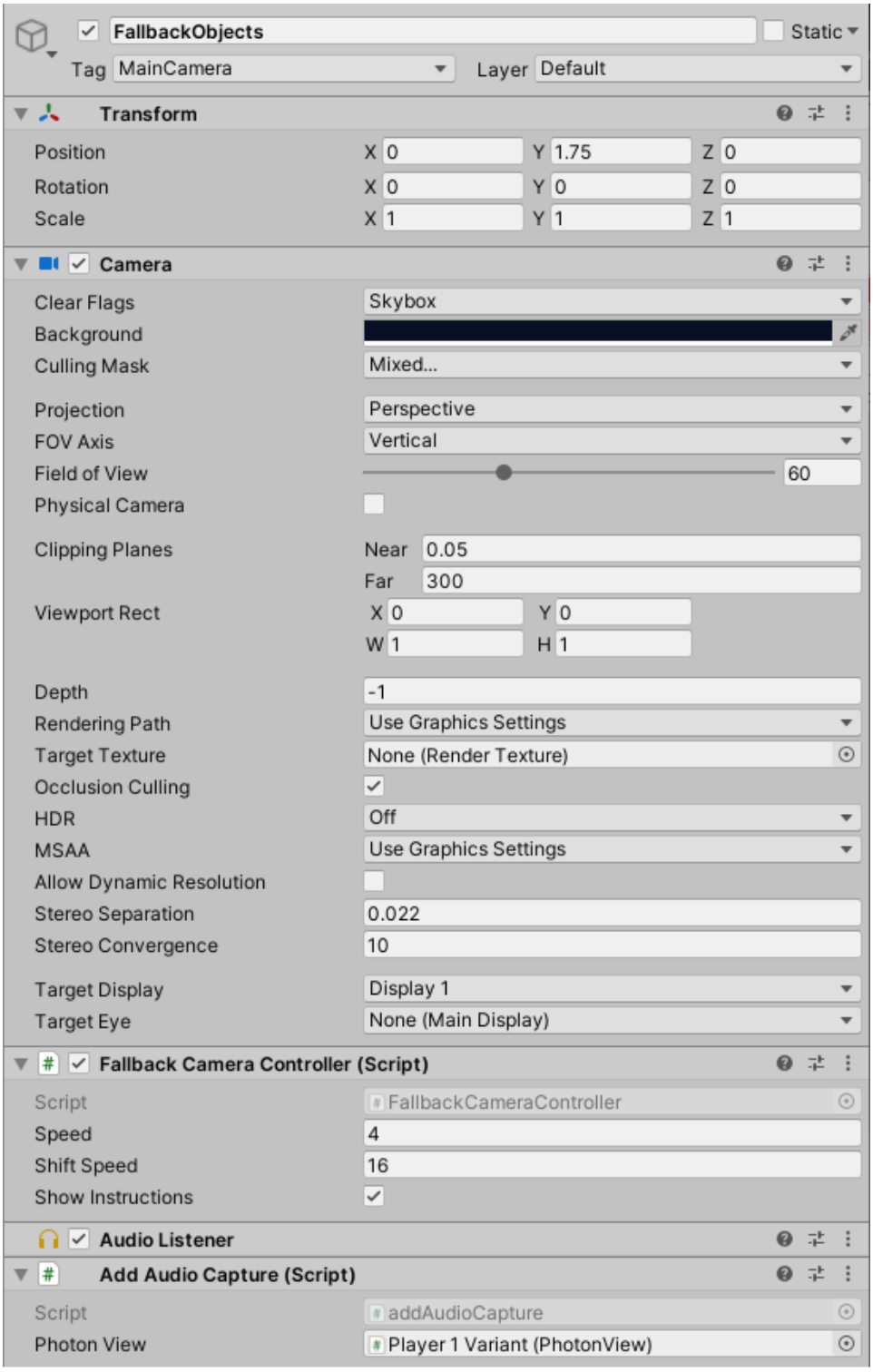

Figure 4.18: Child of the NoSteamVRFallbackObjects, called FallbackObjects

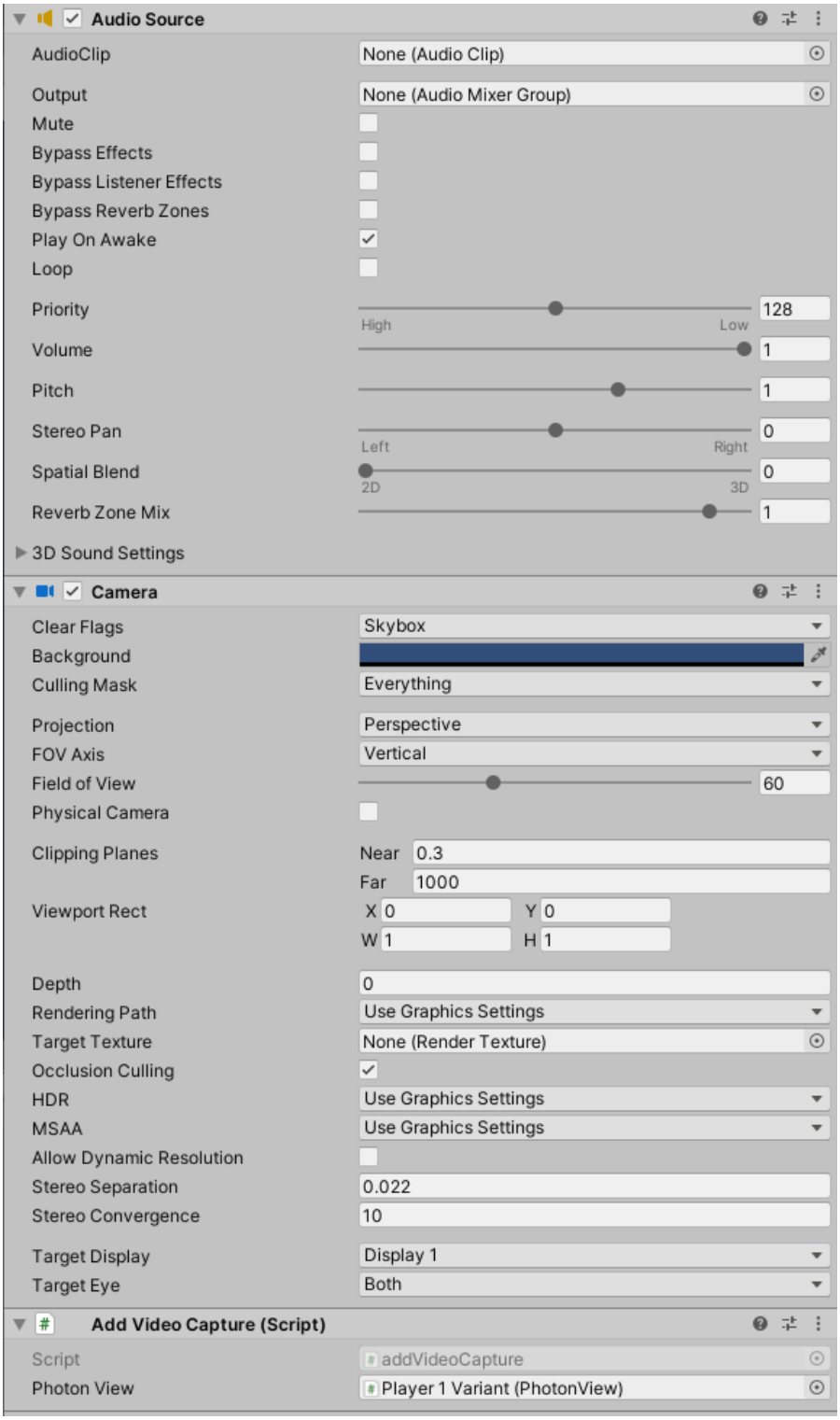

Figure 4.19: Child of the FallbackObjects, called `GameObject`

4.8 A First Sample Application: Ping Pong

We made a simple application to test the beginning of our template and to test how to swap authority. Currently the game just allows players to hit the ball back and forth and rules for points have not been added yet. A simple box collider is would be for the net, we would have used Unity's cloth simulation, but it appeared to not be functioning properly from our testing, using one of the more recent versions of Unity. Currently there is no avatar representation present within the application, but users can still see each other's paddle. Users are able to communicate with each other via Photon Voice's voice chat system.

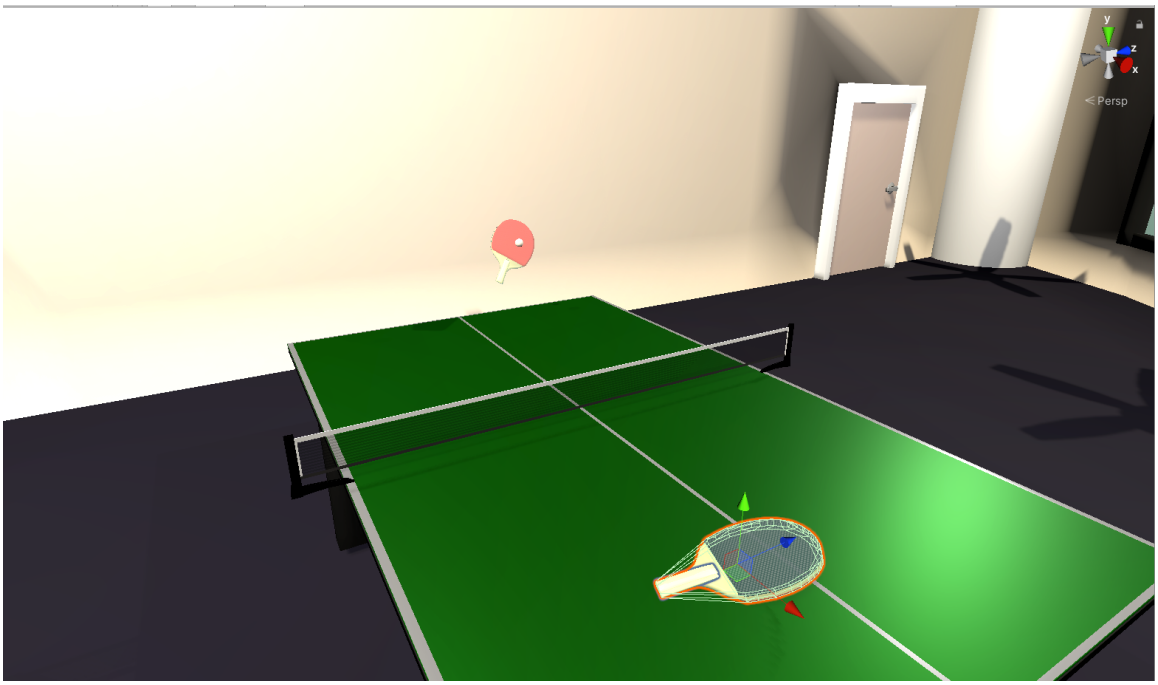

Figure 4.20: A scene that allows users to play ping pong

4.8.1 Physics Interaction

With VR we have continuous input from the hands, so depending on the application we need a method to allow the user to interact with the physics engine. One may think that it may be as simple as attaching a `GameObject` as a child to one of the

hand `GameObjects`. However, this will not necessarily make the `GameObject` interact with other objects like expected. An option to have hand held objects like bats, paddles, etc., interact like we think they should, we may want to have the object follow the users hand through using the difference vector between the positions of the hand and the object to set the velocity of the handheld object. The larger the difference, the larger the velocity. We do this instead of having the paddle as a child of the hand. This is what we did when attempting to create a ping pong game. The idea for the physics interaction came from a tutorial on how to make a bat in VR from 2016 [80]. In this game, we added high friction, using a physic materials, for more spin and increased the project's max angular velocity setting to help simulate ping pong. Through trial and error, we found parameters for the physic materials of the ball, the table, and the paddle. A parameters for the physic Material can be seen in Figure 4.21.

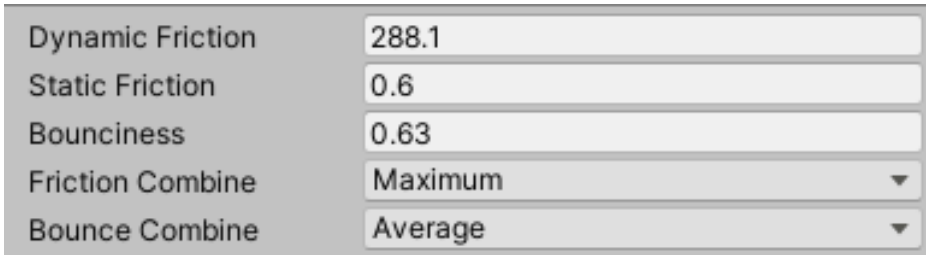

Figure 4.21: The Physic Material used for the Ping Pong paddle

The controllers used for the ping pong game were the Vive trackers attached to a paddle, which can be seen in Figure 4.22. They were a little top heavy, but they help to make the user feel like they are using an actual paddle in the game.

4.8.2 Authority

In the current version, we swap authority when the ball is on the local player's side of the net. We currently have the collision of the paddles on the non-local players disabled. If we did not we may have been able to take authority based on when the ball exited a collision with the non local player's paddle or if it entered a collision with the local player's paddle.

Figure 4.22: Controllers used for the ping pong game

Chapter 5

Application: Doctor Nurse Interaction

5.1 Introduction

Now that we have a template defined we set out to test it in the real world. We were approached by Steven Anbro who is a PhD student in Psychology. He wanted to conduct a user study to compare retention of training methodologies. Doctors and Nurses have to do patient handoffs in a variety of situations. They typically go through some training and are then evaluated on their performance in certain scenarios. Steven set up a user study where there were two rounds of the patient handoffs. In between some of the students were given a specific training and the others were not. The data was collected and is currently being evaluated.

We were requested to construct a Doctor-Nurse patient handoff application in Virtual Reality. This would be the ultimate stress-test of our template. We would need the multi-player capability with audio chat, but we would also need to mimic what the instructors were already used to, and that was a third person camera view of the room that the handoff would take place in. And this third person "teacher view" would have to have the ability to listen the student scenario as well as record it. The extra part was that they wanted information on where the Doctors and Nurses were looking. So we had to use the HTC Vive Pro Eye [31] and have a beam coming from the doctor and nurse avatars that was only visible in the teacher view.

5.2 Application

5.2.1 Models

Avatars: The first thing to do was to select avatars. We went to `turbosquid.com`. We prepared a list of models and model sets for Steven Anbro to look through. He selected a set and we purchased the license to use and modify them. The models were available in the following formats according to the website: 3ds Max 2015 (Native), Unity 2018 (converted), FBX 2014 (exchange).

In Figure 5.1 we have the models to represent the Doctors and Nurses. We liked that the models did not have too many defining features and figured they could be used for customization. We started by importing the FBX files into Blender, which included adding a new armature (skeletons) for them, and setting Materials to certain faces. By faces we mean parts of the mesh, not the human's face. We did this, because the original models seemed to use a single texture for color. This also made it easier to do color modifications later.

Figure 5.1: Doctor and Nurse models used in the study [72]

Equipment: In Figure 5.2 we have some other models (also from `turbosquid.com`) that we used in order to add standard equipment into the room. This also gave users objects to look at while in the room, which would provide more eye tracking data. According to the turbosquid page for these models, the model's available formats are: 3ds Max 2013 (native), FBX (Exchange), OBJ (Exchange).

Figure 5.2: Medical room equipment [7]

Exam Room: The next step was to model the room. Luka Starmer from @one-Reality in the Knowledge Center took care of this part. He used Photgrametry to automatically take pictures of and construct a 3D model of the room. We received the model as a .obj file. Luka sent us the texture to be applied to the model in Unity. In Figure 5.3 you can see the model of the room from the outside. Figure 5.4 shows the model of the room with no colors or textures (so it looks just gray) The full model can be seen later in Figure 5.9

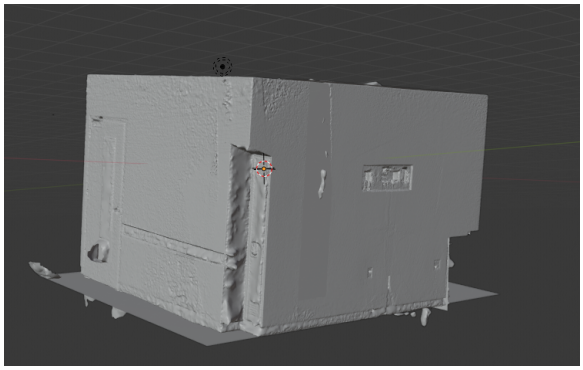

Figure 5.3: The outside of the room model provided by Luka using photogramy in blender

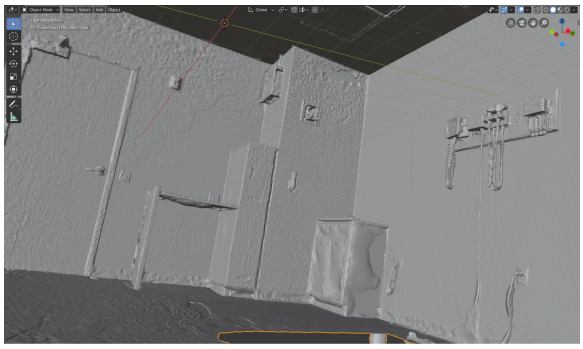

Figure 5.4: The inside of the room model provided by Luka using photogramy in blender

Clipboard: Once the room and avatars were operational, the next item requested was a clipboard. Doctors and nurses usually have notes on their patients that they refer to when discussing specifics. So we were asked if there was a way to add some note capability, and we all agreed upon a clipboard. Figure 5.5 shows the clipboard used. It uses Text Mesh Pro to display the text. We used a file on the computer to read in these notes. In Figure 5.6 we have a fist person view of the clipboard. One of the participants asked if they could just read the notes of the other person in the study, and the answer is no. The clipboards do not show the non-local player's notes.

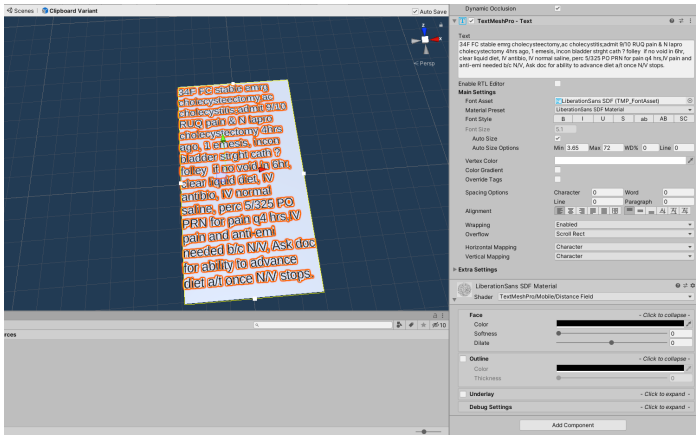

Figure 5.5: Clipboard with user's notes and TextMeshPro settings on the right.

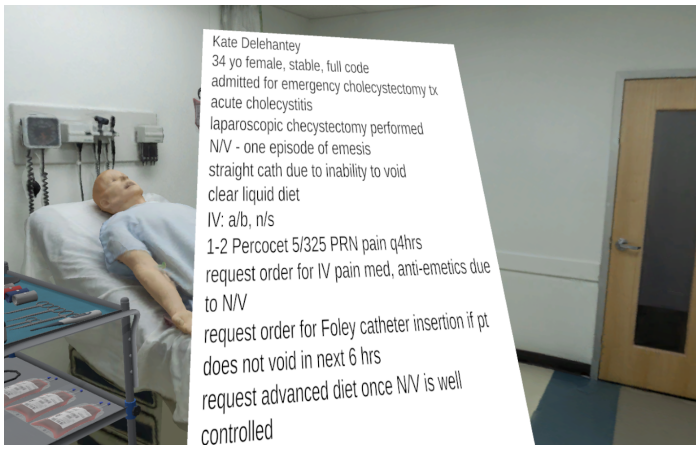

Figure 5.6: First person view of Clipboard

5.2.2 Scene

In Figure 5.7 we see the two rooms the users will use. Users will start in the customization room, select the model they want to represent them, as well as hair color, and skin color. They will then teleport into the lower room and begin the user study.

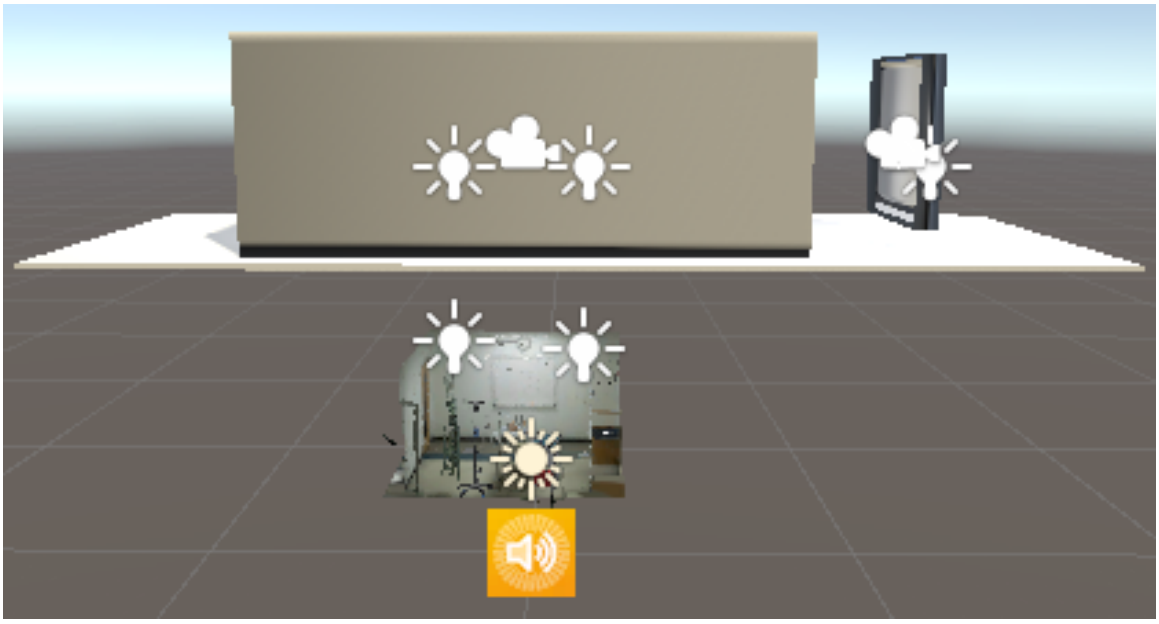

Figure 5.7: An overview of the scene used in the Clinical Scenario in Unity. The top room is the customization room while the bottom is the medical school's room

In Figure 5.8 we see the customization room which was used for avatar customization. The mirror was made by using a Camera and a render texture. The mirror is there to allow users to see themselves before they leave the customization room. There are 3 menus present. Each have five choices. The menu on the left is for skin color, the menu on the right is for hair color, and the menu in the middle is for choosing models. There is also a button on the opposite side of the room that is used for teleporting to the medical school room.

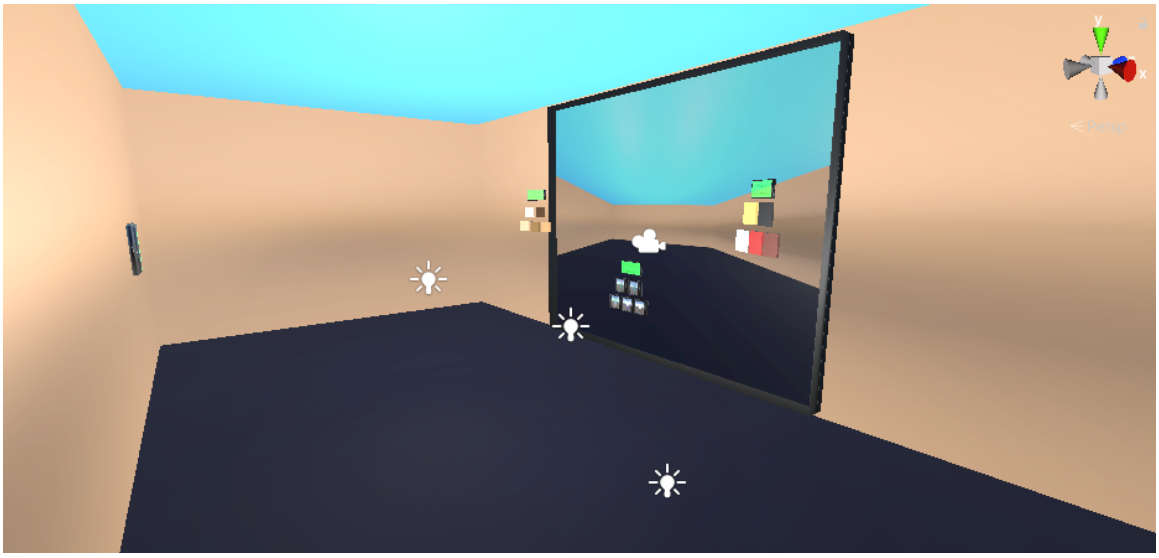

Figure 5.8: The first room users will enter. Here users make choice on customization

Figure 5.9 shows the inside of the medical school patient room. It has been filled with props that have colliders that are used for eyetracking. Users teleport to opposite sides of the room near two doors. The teacher starts out in the upper corner shown in the figure.

Figure 5.9: The medical school room

5.2.3 Customization

Avatar Menu: A close up of the character select menu can be seen in Figure 5.10. Users can either touch the buttons or use a laser pointer to select which model they will use. When selected, the user is not allowed to move for about one second, which is meant to give the user time to stand if straight so that they their head position will be at the same position as the model. After this happens the model will be able to move and the model will also start to bend their knees if the user lowers their head. When selected an Buffered RPC is sent to all other clients to ensure their avatar appears the same to the other clients.

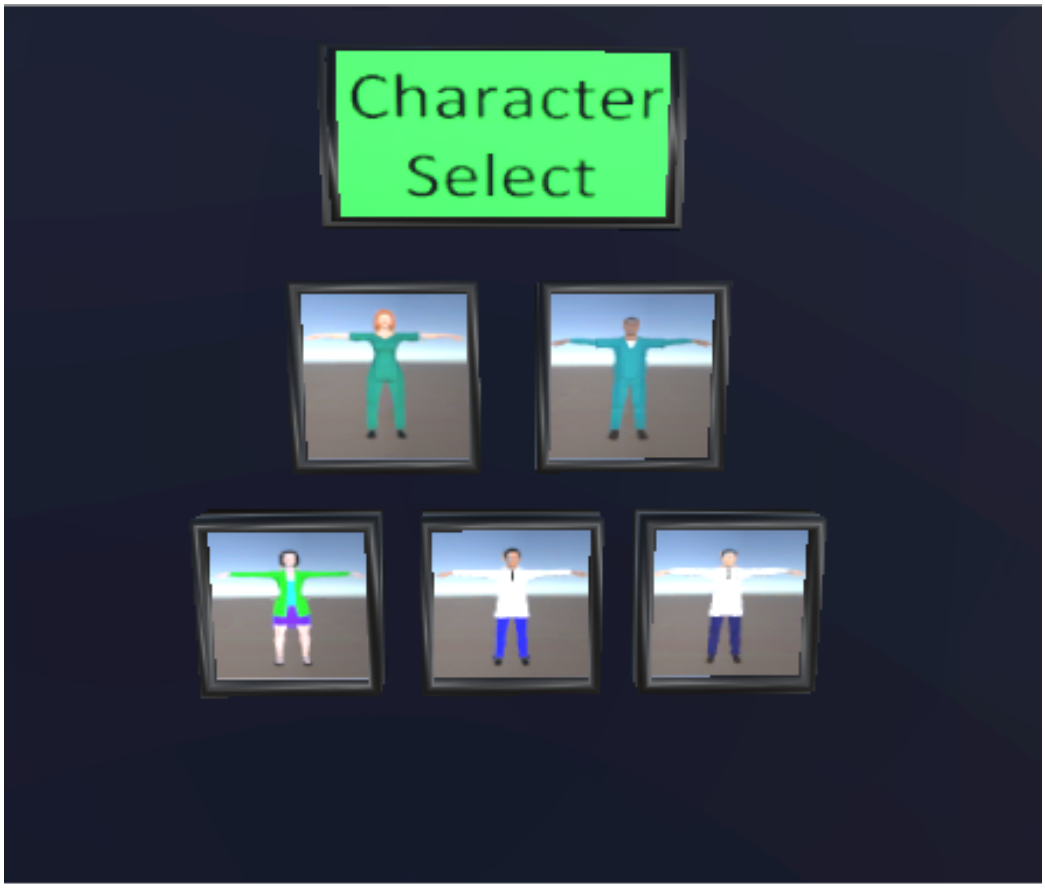

Figure 5.10: The character select menu. Users use these buttons to choose which model they want

Hair and Skin Color Menus: These buttons shown in Figure 5.11 will change the user's hair color. They function in a similar way to how the character select menu functions.

Figure 5.11: The hair color select menu. Users use these buttons to choose which hair color they want

These buttons shown in Figure 5.12 will change the user's hair color. They function in a similar way to the skin color menu.

Figure 5.12: The skin color select menu. Users use these buttons to choose which skin color they want

Menu Selection in Action: A user is deciding which model to use in Figure 5.13. The blue laser collides with objects. This is used to tell what object the user is selecting.

Figure 5.13: Example of user using laser point

A user chooses to have a red hair color in Figure 5.14. When a user wants to make a selection, their pointer will turn green. In this state, if the laser collides with an object that can be selected, a selection will be made.

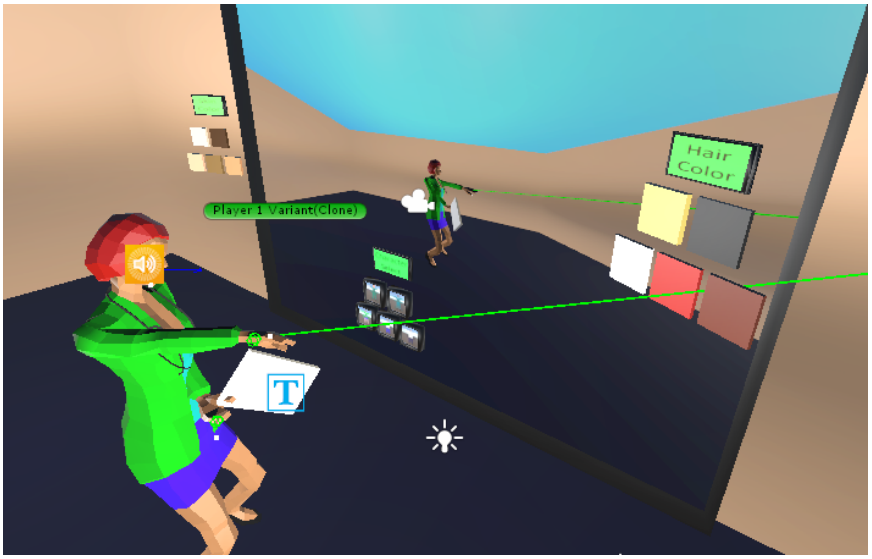

Figure 5.14: The laser turns green to show the user is pressing the controller button

5.2.4 Eye Tracking

We used the tobii eyetracking SDK [71] along with the hTC Vive Pro Eye for eye tracking. We attached scripts to every single object we wanted to record data from. A Boolean value was used to tell the script to write the recorded data. The Boolean value is only set to true if both users pressed the begin button in the customization room. This is to help ensure that we do not store unnecessary data and so that we do not start recording data before the training begins. Both users set a timestamp when they press the button. This was added since the clocks on the computers may have different times. We need to do this to accurately compare data. When a user enters the medical school patient examination room, they record their current time, then a BufferedRPC is used to save a new timestamp created base on the other computer. We needed both participants in the customization room before one of them entered the medical school patient examination room. Otherwise the timestamp recorded on the client of a user that joined late of the other client who joined earlier would be incorrect. Each object will create a file and store it in a folder with a timestamp for a name. The files were named after the `GameObject`s that were being looked at. That timestamp is used for a folder containing each file created, as well as the choices users made for customization. In Listing 5.1 we can see a sample of the output provided by the eye data code. The data is ordered in the following order starting TimeToFirstFixation, Fixation Count, Fixation Total Seconds, Fixation duration list, where TimeTofFirstFixation is the the first value, and the duration list is the last value. TimeToFirstFixation is when the user first looks at the object. Fixation count is the number of times the user looked at the object Fixation Total Seconds is how much the user looked at the object. Fixation duration list is when the seconds of looking at the object starts, and the amount of time the object was being looked at after the user started looking at it.

Mesh Colliders: In order to track when users looked at the mannequin and bed, we create a mesh in blender, then used Unity to add a Mesh Collider so that we could

Listing 5.1: Eye Capture Data

```
6/12/2020 11:33:36 AM
6/12/2020 11:47:27 AM
TimeToFirstFixation, Fixation Count, Fixation Total Seconds, Fixation
   duration list
136.3807, 143, 72.80031,
{136.3807, 0.2106476}, {136.9691, 1.321564}, {139.0789, 0.2225037},
   {140.9894, 0.3887482}, {155.05, 1.566101}, {159.5927, 1.021332},
   {162.991, 0.01104736}, {163.013, 0.8887787}, {164.5683, 0.5438843},
   {166.4895, 0.6554565}, {172.6536, 0.2218628}, {173.8313, 0.0438385},
   {174.0631, 0.7337799}, {178.2179, 0.5441132}, {195.6432, 0.5999451},
   {196.3098, 0.2110291}, {196.6765, 3.031891}, {201.8194, 0.3991547},
   {204.2396, 0.9328613}, {206.9946, 0.04396057}, {210.0371, 1.754822},
   {215.3127, 0.1998138}, {215.657, 0.5666046}, {217.0789, 0.01121521},
   {217.7342, 0.010849}, {218.989, 0.06596375}, {219.4332, 0.6330414},
   {222.7427, 0.03347778}, {232.4608, 0.03338623}, {238.6136,
   0.02270508}, {243.1561, 0.01144409}, {243.5005, 0.4331055}, {245.0665,
   0.3883514}, {245.5775, 0.02203369}, {245.6217, 0.5330048}, {255.7507,
   1.477737}, {259.027, 0.04370117}, {268.2119, 0.0112915}, {268.2342,
   0.4107361}, {271.2328, 0.8111572}, {272.6544, 0.4665833}, {276.1864,
   0.04452515}, {279.8627, 0.1885681}, {280.1956, 0.3222961}, {284.0384,
   0.03250122
```

detect eye tracking on a more unusual shape. We tried to minimize the amount of vertices used, and at least on our computers, we did not notice any notable toll on the performance of the program. The mesh collider and mannequin can be seen in Figure 5.15.

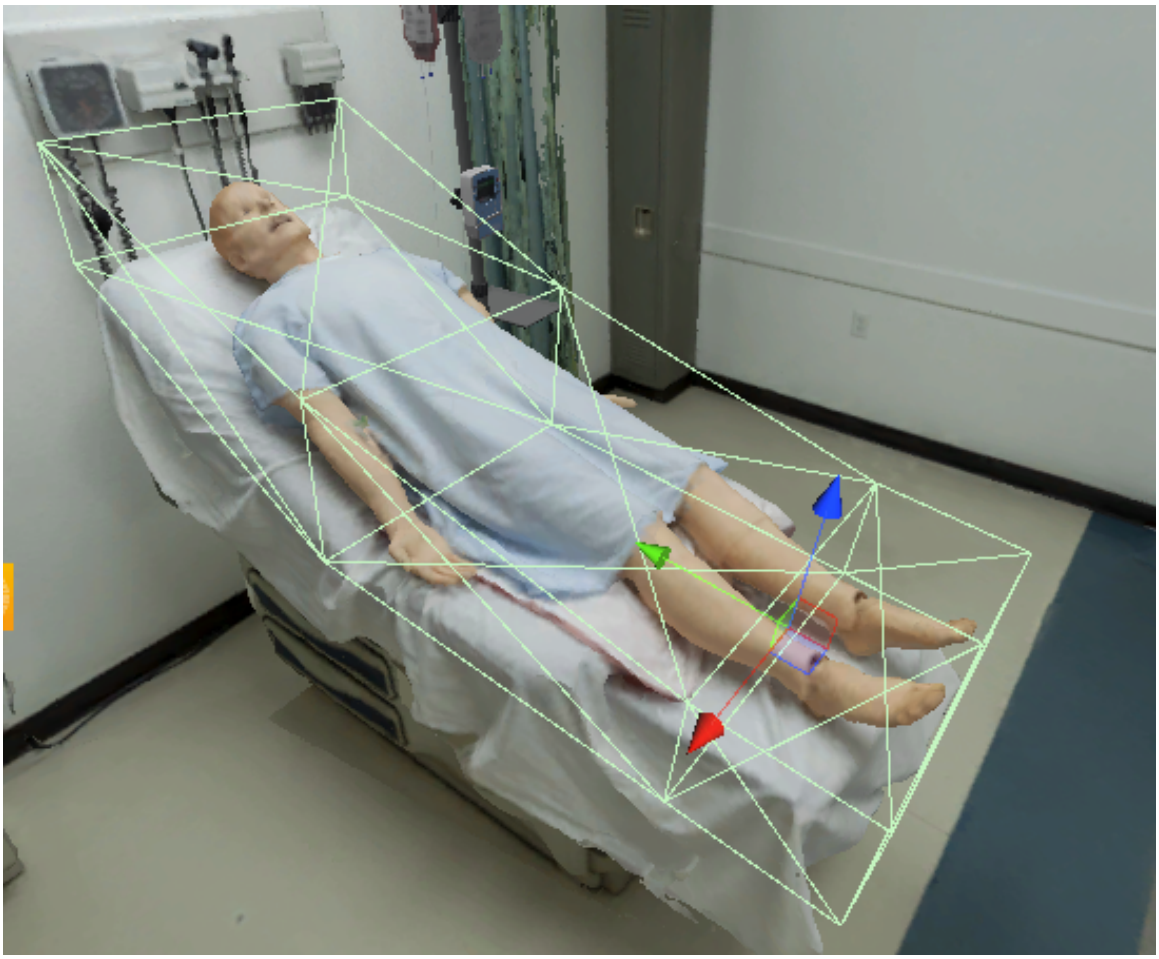

Figure 5.15: Mesh Collider Used for Mannequin and Bed

Eye Direction Lasers: Once the eye tracking was set up we needed to show where the participants were looking when we had the teacher view on. We did this with laser lines coming from the participants head. But we did not want those lasers in the participant's view.

The code in Listing 5.2 shows how we chose to use eye data in order to draw lines based on where the user is looking. We used some code from tobii's online

Listing 5.2: Code that sends over the origin and direction that a person is looking frame

```
void Update()
      {
          if (photonView.IsMine)
          {
              var eyeTrackingData =
                  TobiiXR.GetEyeTrackingData(TobiiXR_TrackingSpace.World);
              if (eyeTrackingData.GazeRay.IsValid)
              {
                  // The origin of the gaze ray is a 3D point
                  var rayOrigin = eyeTrackingData.GazeRay.Origin;

                  // The direction of the gaze ray is a normalized
                      direction vector
                  var rayDirection = eyeTrackingData.GazeRay.Direction;
                  lineRenderer.SetPosition(0, (rayOrigin));
                  lineRenderer.SetPosition(1, (rayOrigin + 10 *
                      rayDirection));
                  this.photonView.RPC("drawRay", RpcTarget.Others,
                      rayOrigin.x, rayOrigin.y, rayOrigin.z,
                      rayDirection.x, rayDirection.y,
                      rayDirection.z);//the RPC that sends the data
              }//turning off line renderers in local copy of game

                  if (turnOffRenderers || steamVRObjects.activeSelf)
                  {
                      characterInstantiation.activateOtherRenderers =
                          false;
                  }
                  else
                  {
                      characterInstantiation.activateOtherRenderers = true;
                  }
              }
      }
    [PunRPC]
      public void drawRay(float x, float y, float z, float a, float b,
          float c)
      {
          lineRenderer.SetPosition(0, (new Vector3(x,y,z)));
          lineRenderer.SetPosition(1, (new Vector3(x, y, z) + 10 * (new
              Vector3(a, b, c))));
      }
```

documentation namely the usage example for `GetEyeTrackingData` [70]. When the line data is received, we use `setPosition()` to set the position of two points on the line. Position 0 is set to the origin point of where a user is looking, while position 1 is set to the origin plus 10 * the directional vector. We probably could have also used a mesh to represent where the user was looking, this could have been more visually appealing. We also could have tired to make the ending point of the LineRender at the point that the user is looking at. Right now the laser goes for a fixed length (sometimes through what the participant is looking at).

Several examples of the eye direction laser lines taken from the instructor's view are shown in Figures 5.16 5.17, and 5.18.

Figure 5.16: One user is looking at the other's board while the other is looking at the other's avatar's face

Figure 5.17: One user is looking at the clipboard notes, while another is looking at the mannequin

Figure 5.18: One user is looking at the nurse while he is describing his patient

5.3 User Study

5.3.1 Overview

Covid-19 has caused many issues world-wide. In our case it impacted how we ran the user study. We had to take measures to protect users in the study. We made sure to disinfect the controllers and headset between each use, and provided disposable mask shields to protect the user's face.

The users were split into pairs and then the pairs were split into two groups, the group using VR, and a group that did not use VR. those two groups were then broken in half based upon the type of patient hand-off training they received. Every pair went through the the study twice (with the training in between the phases). The second time was about a week after the first time. The idea was to compare the control group with the experimental group to see if the experimental group would perform better, just as well, or worse than the control group. The second item looked at was how did VR impact the performance.

Before the hand-off of the patient the participants were allowed to write notes about their patient. If they were in the VR groups they typed out their notes into a text file. This was transferred on a USB drive to be used in the application. In the study there were several patient examination rooms used. Two were for the groups that did not use VR, and two were for the VR groups. For the VR groups we had each room set up with an hTC Vive Pro Eye [31]. The two non-VR rooms were monitored and recorded by an assistant in the control room. This is where the the third computer was set up with the instructor's software running. It was able to view the participants and hear voice chat audio and recorded all of this as well as the participants as they were in the VR patient examination room. For each VR room we had at least one person helping the participants begin the study. They gave instructions, watched to make sure the users did not hurt themselves, disinfected controllers, made sure that the software was used properly, and helped participants with the calibration process for eye tracking. Each round of two participants were

about 20 minutes apart. Each set of rounds took about an entire day. After the interaction/hand-off was complete, users went to another room to debrief and fill out a survey.

5.3.2 Issues and Fixes

One major issue with the first part of the user study was that we made too many assumptions on what the users were comfortable with. A locomotion technique where the user slides across the ground may be simple for us to understand, but not necessarily for people who may have never used VR before and who will probably only use the application once or twice. We noticed that users would have some trouble understanding the locomotion technique since we did not have much time to explain it and since is was not really the main focus of the application.

Originally the customization selection had users walk up to a wall and press buttons. This meant that users would need to use the locomotion method. In order to touch the buttons they would use the model's hands that represents them to collide with the buttons. On the entrance of the collision, the user's color or model choice would have been made. But because of this, users would touch multiple buttons on accident while backing away from the button. Some have kept one of their hands outwards and would forget about it and accidentally press a button. Some would use the touch pad locomotion and not back up far away enough and would touch a button on accident. This issue may have been fixed if there was a way to ensure the user selected an option by pressing a button as their hand is colliding to ensure that they do not accidentally run their hands through the button. Treating the buttons like actually buttons one would use at an elevator could also have helped to ensure the user had made the choice they intended. Another option could have been to just select customization before entering the virtual environment.

Our solution to alleviate the issues with customization was to use a laser pointer for selection. Laser pointers are more convenient since we can allow users to choose options at a distance. It can be argued that it is can allow the use of smaller buttons

since laser points can be small. That's not to say that we cannot animate a models hand to point it's index finger, attach a capsule collider for collision on the finger and use that for selection. The laser pointer was made to turn green when the trigger button on the controller was pressed. It would turn back to blue when a menu option was selected. This was done to prevent accidental selection, and the user would need to press the trigger button down again to turn the laser green. If there were any negatives to using the laser pointer, it may be that they may be less immersive since they require the press of a button on the controller and since there was not a reason for a laser to exist in the type of setting for a clinical training scenario. It should also be noted that the farther away an object is, the more effort needs to be made to have the laser collide with the object since the object appears smaller from a distance.

Some smaller issues just involved making the minimum text on the user's clipboards smaller so that more text could fit on them and moving the buttons away from the mirror so that users could see them better. Another issue was that since the model needed users to place their hands lower than the could to have them straight and to their sides, their avatars would look like they had their arms bent and they were touching their hips. In order to fix this, the `GameObject`s that represent the users hands would need to have been lower compared to the model. We have not finished fixing this.

All of these issues could have been dealt with if we had been able to do a pilot study (as you normally would). However, due to covid-19 and the lockdown and shelter-in-place order by the Governor, this was not possible. Coding for this user study basically had to be finished in a 2 week windo and it was much more difficult to perform testing, since we need multiple people for this. We should have done testing with users outside the lab to help confirm what worked in the application and what would not.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

We developed a template for developing multi-user VR applications. We found issues with the first attempt at a framework that used FizzySteamyMirror[54] and decided to use Photon[18] instead. It uses an on premise server to allow us to connect and share data between at least two computers. Photon Voice 2's code allows us to synchronize `GameObject`s and utilize voice chat. The template allows users to record video and audio which is a useful feature for those intending to do training and research. The template will undergo some changes to improve it, and then we plan to make it available in the future.

We have shown the practicality of the template through two applications. The first was a simple ping pong game. This was attempted to evaluate the networking, voice chat, and authority of objects. The second was the Doctor Nurse patient hand-off application. This tested avatars, menus, and scenario recording.

6.2 Future Work

There are many things that we would like to work on to improve this template and make it better to use.

6.2.1 Avatars

Some improvements to the full-body model avatar representation could be made. For example, experimentation with the HTC Vive Trackers [27] could be used to synchronize the feet, and to tell which direction the user's torso is facing. Another improvement would be code for facial animations, even if it is simple, it could make the avatars appear less static. Such a method is even possible for the head and hands only representation of a user. Models with facial animations would need to be modeled with a mouth. Experimentation with users of different arm lengths, heights, etc. may be useful to ensure that the method is comfortable for people of different body types. It is possible to display the user avatar on the other clients' ends and not display it on the owner of the avatar's end as well, which may be an option for future work.

Whichever method, head and hands or full-body model, should be used may depend on the context it is being used in. Some people may prefer the head and hands, while some may prefer to be a human. The audience that the application is being made for may have an affect on this as well. People in VRChat[77] may choose to represent themselves as very short characters, but perhaps some may find that such an avatar would break immersion, or perhaps they will feel the avatar does not represent them properly. Leg tracking may not even be necessary in many cases, since people probably do not look at their feet very much and since they may only care about other people seeing their feet if perhaps they are dancing or playing a game like soccer. Another use may perhaps be for training, like making sure people don't step on land-mines, or other dangerous situations. So functionality of the avatar and the audience should probably be taken into consideration when deciding on how to represent the player.

6.2.2 Bots

AI Bots may be of use in a Multi-VR application. They can replace other players who left if necessary, and they can be used if other users are not available. Bots

may also be usable for representing a person that would normally be present during training, like a human. Animation states can be easily applied to models, but they may need some custom animations. Custom animations are not necessarily difficult to make as it just involves rotating bones to a pose. Models will move from one pose to the next. If we want the bot to point out certain objects, we can always use Inverse Kinematics and use the object of interest as a target for the IK. Having a bot respond to user's speech may be a difficult task, as it may require text to speech. There may be methods of machine learning that can be applied.

6.2.3 Applications

We hope that any mistakes noted within these thesis help us to improve our research and help others to consider work around for any issues we may have found. For example, in the puzzle game mentioned in Chapter 3 we mentioned that we were unsure if users would be comfortable with jumping. A work around may be to just have players control other objects or creatures in order to jump for them. We also noted some mistakes when it came to UI. We should make sure to think about how the audience we will be testing with will react to the UI and whether or not the UI will be easy to navigate.

There are many possibilities for new multi-vr applications. Perhaps they could be for training, social purposes, exercise, entertainment, and more. For example, we may want to use the template to develop a training application that requires the teamwork of multiple users. Perhaps a fire fighting application would fill this role. Another possibility could be a game where users construct something, which could possibly allow them to learn teamwork.

6.2.4 Medical School

There can be many improvements made to the current application for training medical students. Two of course, are bots and improvements to the models as previously mentioned. Another may include object interaction. Perhaps the UI for choosing

options can be improved. Pressing a button to teleport into a room is not really realistic, perhaps it would be better to start users in a room that connects to a hallway. The hallway could connect to other rooms. This may even be usable for creating and joining rooms based on which door in the hallway is opened.

6.2.5 Possible Improvements of Template

There may be some flaws in the template that are noticed over time. Any adjustments to it will help other developers in the future. Also, more locomotion samples would be beneficial, this way other developers will have more options to choose from. User studies on portions of the template, for example, for the body models may be beneficial. A menu system that is easily editable may beneficial for developers. This may help to reduce the amount of time that may be needed to develop UI for other students. Perhaps it may be beneficial to include Prefabs that use different hardware like VR gloves. New hardware can open up even more possibilities for VR and Multi-User VR applications.

6.2.6 New Applications

There are many applications that can be produced using this template. Given enough hardware we could test applications with many people. The ability to provide immersion and motion controls allows us to build unique experiences. One idea for an application is one for academic purposes, a classroom. Users may be able to be in a classroom remotely. VR provides another advantage which is that users will be able to do things that they would not normally be able to do in real life. We do not have to be limited to the setting of a classroom. Instead of having to go on a field trip, people using an application could just teleport there or just load a new scene. Quizzing system could also be implemented into VR. The gameification of the classroom setting may also make learning more enjoyable. In some cases like during a pandemic, or if most users are remote at a certain time like the summer. Given that labs, for Physics or Chemistry for example, may be difficult to perform when

everyone is remote, perhaps VR could help simulate real life lab scenarios as well.

Some other applications include cooperative training. One could be firefighting, another could be police training, we could also potentially see a construction application where users work together to build something. Entertainment can also be looked into. How would a first person shooter function in VR? Would players use body-models or just head and hands? Would it be unfair to allow players to make themselves smaller targets by lowing their camera? How do we make sure locomotion is comfortable and fair at the same time in a competitive scenario? We could also see some cooperative games as well. There are also other hardware that can be used. As we have seen treadmills, gloves, and more. But we could also take advantage of sense that normally are not used in video games like taste and smell. Hardware that makes interesting use of touch may be of benefit as well. The possibilities for Multi-VR games are endless. The template is intended to make building Multi-VR games easier, but developers are still given a blank canvas.

Bibliography

[1] M. Al Zayer, P. MacNeilage, and E. Folmer. Virtual locomotion: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 26(6):2315–2334, 2020. DOI: 10.1109/TVCG.2018.2887379.

[2] Majed Al Zayer, Isayas B. Adhanom, Paul MacNeilage, and Eelke Folmer. The effect of field-of-view restriction on sex bias in vr sickness and spatial navigation performance. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, 354:1–354:12, Glasgow, Scotland UK. ACM, 2019. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300584. URL: http://doi.acm.org/10.1145/3290605.3300584.

[3] Steven J. Anbro, Alison J. Szarko, Ramona A. Houmanfar, Amber M. Maraccini, Laura H. Crosswell, Frederick C. Harris, Michelle Rebaleati, and Luka Starmer. Using virtual simulations to assess situational awareness and communication in medical and nursing education: a technical feasibility study. *Journal of Organizational Behavior Management*, 40(1-2):129–139, 2020. DOI: 10.1080/01608061.2020.1746474. eprint: https://doi.org/10.1080/01608061.2020.1746474. URL: https://doi.org/10.1080/01608061.2020.1746474.

[4] Kurt Andersen, Lucas Calabrese, Andrew Flangas, Sergiu Dascalu, and Frederick C. Harris. A comparison between a natural and an inorganic locomotion technique. In Shahram Latifi, editor, *17th International Conference on Information Technology–New Generations (ITNG 2020)*, pages 317–323, Cham. Springer International Publishing, 2020. ISBN: 978-3-030-43020-7.

[5] Kurt Andersen, Simone José Gaab, Javad Sattarvand, and Frederick C. Harris, Jr. METS VR: mining evacuation training simulator in virtual reality for underground mines. In Shahram Latifi, editor, *17th International Conference on Information Technology–New Generations (ITNG 2020)*, pages 325–332, Cham. Springer International Publishing, 2020. ISBN: 978-3-030-43020-7.

[6] Kurt T Anderson. *A Comparison of Effectiveness and Immersion of Different Gait Techniques in Virtual Reality*. Master's thesis, University of Nevada, Reno, Department of Computer Science and Engineering, December 2019. Advisor: Frederick C Harris, Jr.

[7] AT_studio Pro Models_3D. 3d four medicals equipment (2) model. URL: https://www.turbosquid.com/3d-models/3d-real-medical-equipment-model-1432398 (visited on 07/14/2020).

[8] Bearded Man Studios, Inc. Forge networking remastered, March 20, 2019. (Visited on 07/14/2020). https://assetstore.unity.com/packages/tools/network/forge-networking-remastered-38344.

[9] Blender Foundation. Blender - a 3d modelling and rendering package, Buikslotermeerplein 161, 1025 ET Amsterdam, the Netherlands, 2020. URL: http://www.blender.org (visited on 07/14/2020).

[10] M. Boldt, M. Bonfert, I. Lehne, M. Cahnbley, K. Korschinq, L. Bikas, S. Finke, M. Hanci, V. Kraft, B. Liu, T. Nguyen, A. Panova, R. Singh, A. Steenbergen, R. Malaka, and J. Jan Smeddinck. You shall not pass: non-intrusive feedback for virtual walls in vr environments with room-scale mapping. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 143–150, Los Alamitos, CA, USA. IEEE Computer Society, March 2018. DOI: 10.1109/VR.2018.8446177. URL: https://doi.ieeecomputersociety.org/10.1109/VR.2018.8446177.

[11] Evren Bozgeyikli, Andrew Raij, Srinivas Katkoori, and Rajiv Dubey. Point & teleport locomotion technique for virtual reality. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '16, 205–216, Austin, Texas, USA. Association for Computing Machinery, 2016. ISBN: 9781450344562. DOI: 10.1145/2967934.2968105. URL: https://doi.org/10.1145/2967934.2968105.

[12] Lucas Calabrese, Andrew Flangas, and Frederick C. Harris, Jr. *Multi-User VR cooperative puzzle game*. In *Proceedings of the 17th International Conference on Information Technology : New Generations (ITNG 2020)*. Shahram Latifi, editor. Volume 1134. Advances in Intelligent Systems and Computing. DOI: https://doi.org/10.1007/978-3-03-43020-7_39. Springer International Publishing, Las Vegas, NV, April 2020. Chapter 39, pages 293–299. ISBN: 978-3-030-43020-7.

[13] Polona Caserman, Augusto Garcia-Agundez, Robert Konrad, Stefan Göbel, and Ralf Steinmetz. Real-time body tracking in virtual reality using a vive tracker. *Virtual Reality*, 23(2):155–168, June 2019. ISSN: 1434-9957. DOI: 10.1007/s10055-018-0374-z. URL: https://doi.org/10.1007/s10055-018-0374-z.

[14] DarkRift Networking. Darkrift networking 2. URL: https://assetstore.unity.com/packages/tools/network/darkrift-networking-2-95309 (visited on 07/15/2020).

[15] Epic Games. Networking and multiplayer. URL: https://docs.unrealengine.com/en-US/Gameplay/Networking/index.html (visited on 07/15/2020). Unreal Engine 4 Documentation.

[16] Epic Games. Unreal engine 4. URL: https://www.unrealengine.com/en-US/ (visited on 07/15/2020).

[17] Epic Games. Virtual reality development. URL: https://docs.unrealengine.com/en-US/Platforms/VR/index.html (visited on 07/15/2020). Unreal Engine 4 Documentation.

[18] Exit Games. Photon unity networking 2. URL: https://doc-api.photonengine.com/en/pun/v2/index.html (visited on 07/14/2020).

[19] Exit Games. Photon voice 2, July 9, 2020. URL: https://assetstore.unity.com/packages/tools/audio/photon-voice-2-130518 (visited on 07/15/2020).

[20] Exit Games. We have the fitting plan! URL: https://www.photonengine.com/en-US/Voice/pricing (visited on 07/14/2020).

[21] Facebook Research. AR/VR creating the future of personal and shared reality, Facebook. URL: https://research.fb.com/category/augmented-reality-virtual-reality/ (visited on 10/24/2019).

[22] Facebook Technologies. Oculus. URL: https://www.oculus.com/ (visited on 07/15/2020).

[23] Gears For Breakfast. A hat in time, October 5, 2017. URL: https://store.steampowered.com/app/253230/A_Hat_in_Time/ (visited on 07/14/2020).

[24] Don Glover. Unet deprecation faq, June 3, 2020. URL: https://support.unity3d.com/hc/en-us/articles/360001252086-UNet-Deprecation-FAQ (visited on 07/14/2020).

[25] Google. Google AR & VR. URL: https://vr.google.com/ (visited on 07/14/2020).

[26] Alex Hansen, Kurt Andersen, Brittany Sievert, Jalal Kiswani, Sergiu M. Dascalu, and Frederick C. Harris, Jr. Let's vr: a multiplayer framework for virtual reality. In *Proceedings of the ISCA 27th International Conference on Software Engineering and Data Engineering (SEDE 2018)*, October 2018.

[27] hTC. Htc vive tracker (2018) - european version. URL: https://www.amazon.com/HTC-Vive-Tracker-European-Version/dp/B07BYVB3RW/ref=olp_product_details?_encoding=UTF8&me=&qid=1570916982&sr=8-6 (visited on 07/14/2020).

[28] hTC. Htc vive virtual reality system. URL: https://www.amazon.com/HTC-Vive-Virtual-Reality-System-PC/dp/B00VF5NT4I?SubscriptionId=AKIAILSHYYTFIVPWUY6Q&tag=duckduckgo-d-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B00VF5NT4I&th=1 (visited on 07/14/2020).

[29] hTC. Vive. URL: https://www.vive.com/us/ (visited on 07/14/2020).

[30] hTC. Vive pro. URL: https://www.vive.com/us/vive-pro-vr/ (visited on 07/15/2020).

[31] hTC. Vive pro eye. URL: https://www.vive.com/eu/product/vive-pro-eye (visited on 07/15/2020).

[32] H. Iwata, H. Yano, and F. Nakaizumi. Gait master: a versatile locomotion interface for uneven virtual terrain. In *Proceedings IEEE Virtual Reality 2001*, pages 131–137, March 2001. DOI: 10.1109/VR.2001.913779.

[33] Dongsik Jo, Kangsoo Kim, Gregory F. Welch, Woojin Jeon, Yongwan Kim, Ki-Hong Kim, and Gerard Jounghyun Kim. The impact of avatar-owner visual similarity on body ownership in immersive virtual reality. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, VRST '17, Gothenburg, Sweden. Association for Computing Machinery, 2017. ISBN: 9781450355483. DOI: 10.1145/3139131.3141214. URL: https://doi.org/10.1145/3139131.3141214.

[34] Joohee Jun, Myeongul Jung, So-Yeon Kim, and Kwanguk (Kenny) Kim. Full-body ownership illusion can change our emotion. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, Montreal QC, Canada. Association for Computing Machinery, 2018. ISBN: 9781450356206. DOI: 10.1145/3173574.3174175. URL: https://doi.org/10.1145/3173574.3174175.

[35] Meeri Kim. Cybersickness: why people experience motion sickness during virtual reality, August 14, 2019. URL: https://www.insidescience.org/news/cybersickness-why-people-experience-motion-sickness-during-virtual-reality (visited on 07/14/2020).

[36] Eric Krokos, Catherine Plaisant, and Amitabh Varshney. Virtual memory palaces: immersion aids recall. *Virtual Reality*, 23(1):1–15, March 2019. ISSN: 1434-9957. DOI: 10.1007/s10055-018-0346-3. URL: https://doi.org/10.1007/s10055-018-0346-3.

[37] James F. Kurose and Keith W. Ross. *Computer networking: A Top-Down Approach*. Pearson India Education Services Pvt., 2017.

[38] Lava. Lava, Laboratory for Advanced Visualization and Applications, University of Hawai'i at Mānoa. URL: https://www.lavaflow.info/ (visited on 07/14/2020).

[39] Joeseph J LaViola, Ernst Kruijff, Ryan P McMahan, Doug A Bowman, and Ivan Poupyrev. *3D User Interfaces: Theory and Practice*. Addison-Wesley, 2nd edition, 2017. ISBN: 978-0134034324.

[40] Brian Lindenhof. Hi5 gloves in climbey - update 12/15/2017. URL: https://www.youtube.com/watch?v=cPvgTIh_I7s&t=193s (visited on 07/14/2020).

[41] Jonathan Linowes. *Unity Virtual Reality Projects: Learn Virtual Reality by Developing More than 10 Engaging Projects with Unity 2018*. Packt Publishing, Birmingham, UK, 2018.

[42] Jonathan Linowes. *Unity virtual reality projects: learn virtual reality by developing more than 10 engaging projects with unity 2018*. In Packt Publishing, Birmingham, UK, 2018. Chapter 7: Locomotion and Comfort, pages 201–235.

[43] J. Lugrin, J. Latt, and M. E. Latoschik. Avatar anthropomorphism and illusion of body ownership in vr. In *2015 IEEE Virtual Reality (VR)*, pages 229–230, March 2015. DOI: 10.1109/VR.2015.7223379.

[44] D. Mazzoni. Audacity(r): free audio editor and recorder, Audacity. URL: http://audacity.sourceforge.net/ (visited on 07/15/2020). Project has moved to https://www.audacityteam.org/ (Last visited 07/15/2020).

[45] Megacity: a collaborative virtual reality environment for emergency response, training, and decision making. URL: http://www.cs.bowiestate.edu/sharad/vrlab/MegaCity.html (visited on 10/28/2019).

[46] Mirror Networking. Mirror networking, 2020. URL: https://mirror-networking.com/ (visited on 07/14/2020).

[47] Moon Studios. Ori and the blind forest, March 11, 2015. URL: https://store.steampowered.com/app/261570/Ori_and_the_Blind_Forest/ (visited on 07/15/2020).

[48] Andrew E. Munoz, Zach Young, Sergiu Dascalu, and Frederick C. Harris, Jr. Tdvr: tower defense in virtual reality: a multiplayer strategy simulation. In Shahram Latifi, editor, *17th International Conference on Information Technology–New Generations (ITNG 2020)*, pages 301–307, Cham. Springer International Publishing, 2020. ISBN: 978-3-030-43020-7.

[49] S. Narang, A. Best, and D. Manocha. Simulating movement interactions between avatars agents in virtual worlds using human motion constraints. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 9–16, March 2018. DOI: 10.1109/VR.2018.8446152.

[50] Nitom. Steamvr game-climbey. URL: https://hi5vrglove.com/showcase/steamvr-game-climbey (visited on 07/14/2020).

[51] Noitom. Hi5 vr glove, 2020. URL: https://hi5vrglove.com/ (visited on 07/14/2020).

[52] Alexander Novotny, Rowan Gudmundsson, and Frederick Harris. A unity framework for multi-user VR experiences. EPiC Series in Computing, 69:13–21, 2020. Gordon Lee and Ying Jin, editors. ISSN: 2398-7340. DOI: 10.29007/r1q2. URL: https://easychair.org/publications/paper/59bX.

[53] Hirav Parekh. *Optimizing Natural Walking Usage in VR using Redirected Teleportation*. Master's thesis, University of Nevada Reno, 2017.

[54] Raystorms. Fizzysteamymirror. (Visited on 07/14/2020). https://github.com/Raystorms/FizzySteamyMirror Current version now on https://github.com/Chykary/FizzySteamworks.

[55] Sharif Razzaque, Zachariah Kohn, and Mary C. Whitton. Redirected Walking. In *Eurographics 2001 - Short Presentations*, volume 9, pages 105–106. Eurographics Association, 2001. DOI: 10.2312/egs.20011036. URL: http://www.cs.unc.edu/techreports/01-007.pdf (visited on 07/15/2020).

[56] RockVR. Video capture, November 28, 2017. URL: https://assetstore.unity.com/packages/tools/video/video-capture-75653 (visited on 07/14/2020).

[57] Graham Sellers, Richard S. Wright, Jr., and Nicholas Haemel. *OpenGL SuperBible Comprehensive Tutorial and Reference*. Pearson Education, Inc., second edition, 2016.

[58] Sensoryx. Sensoryx - vrfree® glove - intuitive vr interaction, August 3, 2017. URL: https://www.sensoryx.com/ (visited on 07/18/2020).

[59] Sharad Sharma. Game-theme based instructional (gti) or virtual reality instructional (vri) modules, Virtual Reality Laboratory, Bowie State University. URL: http://www.cs.bowiestate.edu/sharad/vrlab/course.html (visited on 07/14/2020).

[60] Sharad Sharma. Virtual reality laboratory. URL: http://www.cs.bowiestate.edu/sharad/vrlab/index.html (visited on 07/14/2020).

[61] Mary Shaw. Writing good software engineering research papers: minitutorial. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, 726–736, Portland, Oregon. IEEE Computer Society, 2003. ISBN: 076951877X.

[62] Bill Sherman. Freevr: virtual reality integration library. URL: http://www.freevr.org/ (visited on 07/14/2020).

[63] Mel Slater and Sylvia Wilbur. A framework for immersive virtual environments (five): speculations on the role of presence in virtual environments. *Presence: Teleoperators & Virtual Environments*, 6(6):603–616. ISSN: 1054-7460. DOI: 10.1162/pres.1997.6.6.603. URL: https://doi.org/10.1162/pres.1997.6.6.603.

[64] Placeholder Software. Dissonance voice chat, June 4, 2020. URL: https://assetstore.unity.com/packages/tools/audio/dissonance-voice-chat-70078 (visited on 07/15/2020).

[65] Ian Sommerville. *Software Engineering*. Pearson, 2016.

[66] Hyungki Son, Hyunjae Gil, Sangkyu Byeon, Sang-Youn Kim, and Jin Ryong Kim. Realwalk: feeling ground surfaces while walking in virtual reality. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, D400:1–D400:4, Montreal QC, Canada. ACM, 2018. ISBN: 978-1-4503-5621-3. DOI: 10.1145/3170427.3186474. URL: http://doi.acm.org/10.1145/3170427.3186474.

[67] Walker Spurgeon. *Exploring Hands-Free Alternatives for Teleportation in VR*. Master's thesis, University of Nevada Reno, 2018.

[68] A. Steed, Y. Pan, F. Zisch, and W. Steptoe. The impact of a self-avatar on cognitive load in immersive virtual reality. In *2016 IEEE Virtual Reality (VR)*, pages 67–76, March 2016. DOI: 10.1109/VR.2016.7504689.

[69] Rui Sun, Yenchun Jim Wu, and Qian Cai. The effect of a virtual reality learning environment on learners' spatial ability. *Virtual Reality*, 23(4):385–398, December 2019. ISSN: 1434-9957. DOI: 10.1007/s10055-018-0355-2. URL: https://doi.org/10.1007/s10055-018-0355-2.

[70] Tobii. Photon unity networking 2. URL: https://vr.tobii.com/sdk/develop/unity/documentation/usage-examples/ (visited on 07/15/2020).

[71] Tobii. Tobii xr sdk, 2020. URL: https://vr.tobii.com/sdk (visited on 07/14/2020).

[72] Turbosquid. 3d models for professionals, 2020. URL: https://www.turbosquid.com/ (visited on 07/14/2020).

[73] Unity Technologies. Unity - video game engine, 2020. URL: http://www.unity3d.com (visited on 07/14/2020).

[74] Unity Technologies. Unity asset store - the best assets for game making. URL: https://assetstore.unity.com/ (visited on 07/15/2020).

[75] Valve Corporation. SteamVR plugin, November 6, 2019. (Visited on 07/14/2020). URL: https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647.

[76] Virtuix. Omni by virtuix - the leading and most popular vr motion platform. URL: https://www.virtuix.com/ (visited on 07/15/2020).

[77] VRChat Inc. Vrchat. URL: https://www.vrchat.com/ (visited on 07/14/2020).

[78] Ansley Watson. Experience virtual reality at UA little rock, September 30, 2019. URL: https://katv.com/community/good-afternoon-ark/whats-on-good-afternoon-arkansas/experience-virtual-reality-at-ua-little-rock (visited on 07/15/2020). KATV, ABC 7, Little Rock.

[79] Séamas Weech, Sophie Kenny, and Michael Barnett-Cowan. Presence and cybersickness in virtual reality are negatively related: a review. *Frontiers in Psychology*, 10:158, 2019. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2019.00158. URL: https://www.frontiersin.org/article/10.3389/fpsyg.2019.00158.

[80] Jason Weimann. Unity baseball bat physics, April 11, 2016. URL: https://unity3d.college/2016/04/11/baseball-bat-physics-unity/ (visited on 07/15/2020).

[81] T. Weißker, A. Kunert, B. Fröhlich, and A. Kulik. Spatial updating and simulator sickness during steering and jumping in immersive virtual environments. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 97–104, March 18, 2018. DOI: 10.1109/VR.2018.8446620.

[82] C. Wienrich, K. Schindler, N. Döllinqer, S. Kock, and O. Traupe. Social presence and cooperation in large-scale multi-user virtual reality - the relevance of social interdependence for location-based environments. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 207–214, Reutlingen, Germany, March 18, 2018. DOI: 10.1109/VR.2018.8446575.

[83] Preston Tunnell Wilson, William Kalescky, Ansel MacLaughlin, and Betsy Williams. Vr locomotion: walking > walking in place > arm swinging. In *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*, VRCAI '16, 243–249, Zhuhai, China. Association for Computing Machinery, 2016. ISBN: 9781450346924. DOI: 10.1145/3013971.3014010. URL: https://doi.org/10.1145/3013971.3014010.

[84] Shunki Yamashita, Ryota Ishida, Arihide Takahashi, Hsueh-Han Wu, Hironori Mitake, and Shoichi Hasegawa. Gum-gum shooting: inducing a sense of arm elongation via forearm skin-stretch and the change in the center of gravity. In *ACM SIGGRAPH 2018 Emerging Technologies*, SIGGRAPH '18, 8:1–8:2, Vancouver, British Columbia, Canada. ACM, 2018. ISBN: 978-1-4503-5810-1. DOI: 10.1145/3214907.3214909. URL: http://doi.acm.org/10.1145/3214907.3214909.