Multi-User VR Cooperative Puzzle Game

Lucas Calabrese, Andrew Flangas, Frederick C. Harris, Jr.
Department of Computer Science and Engineering
University of Nevada, Reno
{lcalabrese, andrewflangas}@nevada.unr.edu, fred.harris@cse.unr.edu

***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

I. 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 [1] 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 II-B, 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 II. Gameplay is presented in Section III, and Conclusions and Future Work are covered in Section IV.

II. The Creation Process

A. 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 [2]. 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 [3].

The next step was to create the animations for the witches. One of the better animation papers was written by Narang, Best, and Manocha [4]. 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. 1.

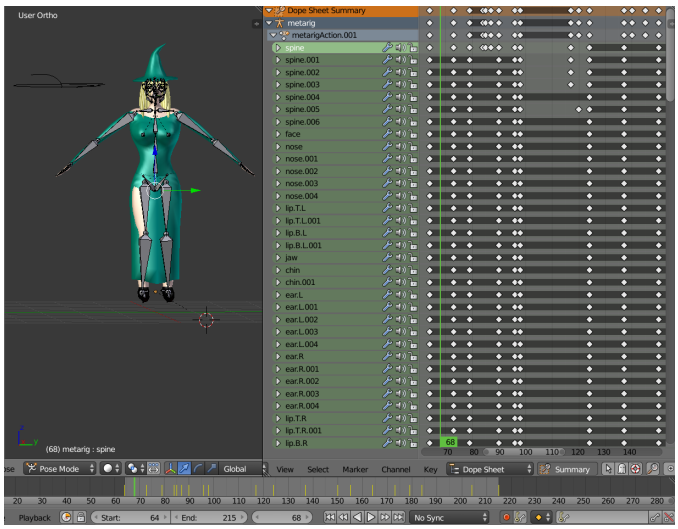

Fig. 1: A Screenshot in Blender which shows how to setup animations using the human rig for the witch model.

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 2

Fig. 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.

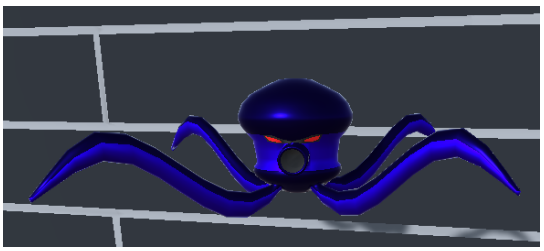

Fig. 3: The Octopus model

B. 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.

C. 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. 4-8.

Swap Power: The swap power (Fig. 4) is used to instantaneously switch the position of the players with other GameObjects in the scene.

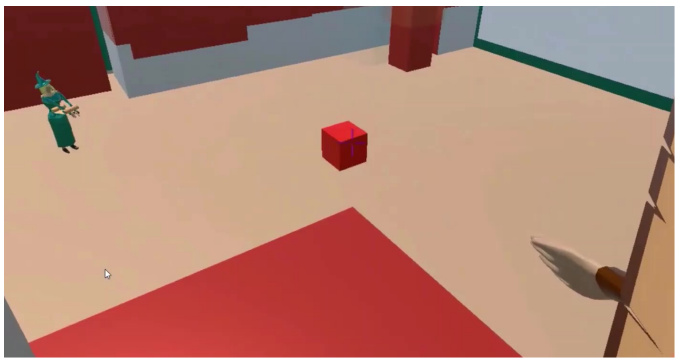

Fig. 4: A first person view showing the object that the swap power applies to has its material changed to red. After the swap power is applied, the player will switch places with the cube.

Shrink Power: The shrink power (Fig. 5) is designed for the players to fit through small tunnels or other similar obstacles by making the player significantly smaller.

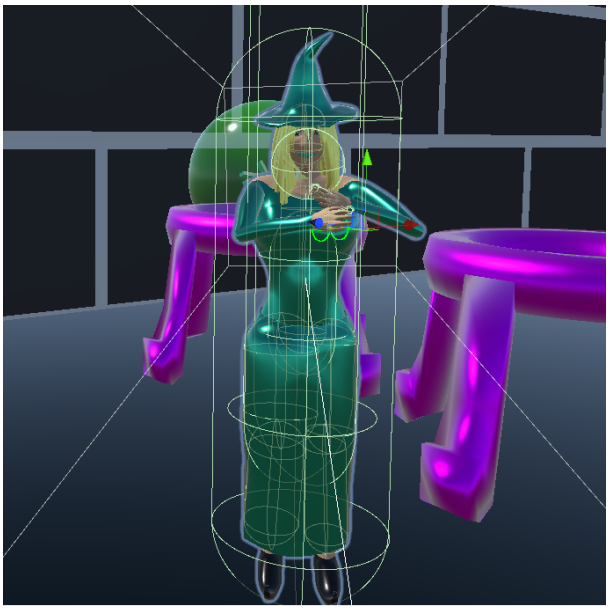

Fig. 5: The shrink power scales the player to a considerably smaller size. Notice the crystal ball stands next to the player.

Freeze Power: The freeze power (Fig. 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.

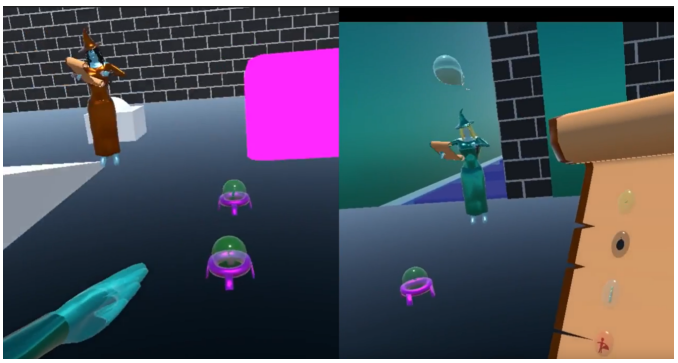

Fig. 6: A split screen (two images from different player's screens). When the ice power is selected and activated it changes the material of your witch's skin into a light blue color. The ice power is bringing down the balloon seen in the right window.

Bomb Power: The bomb power (seen in Fig. 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.

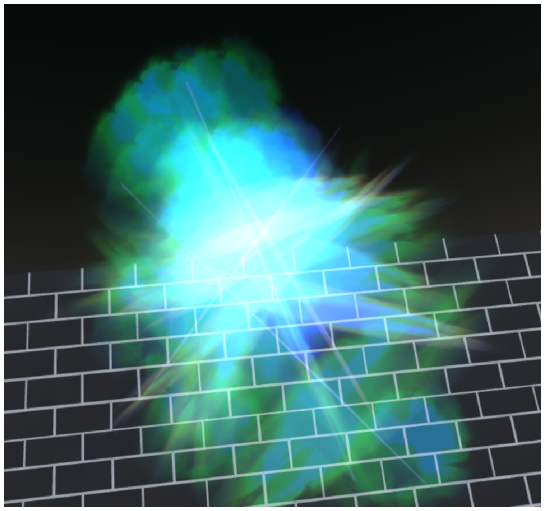

Fig. 7: The fireball is used for the bomb explosion power.

Fire Power: Lastly, the fire power (Fig. 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.

Fig. 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.

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. GameObjects 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 meaningful and useful way [5] 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 [6]. 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.

D. 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. 9.

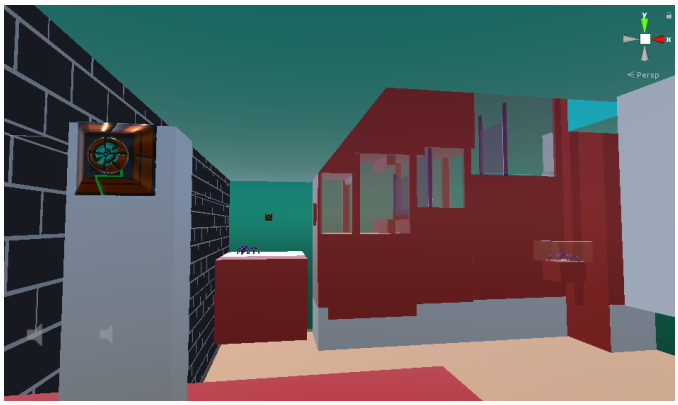

Fig. 9: The puzzle level with the barriers, as well as the buttons the players use to open them.

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 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.

E. 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 [7]. 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 GameObjects 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. 6 and Fig. 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 [7].

III. Gameplay

A. Locomotion

The locomotion method used included both room-scale and the controller to move. It was similar to glide locomotion [8]. 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.

B. 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 [7]. The feature is a necessity since players will need to communicate their ideas to solve puzzles.

IV. Conclusions and Future Work

A. 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.

B. 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 [8], 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 [9], 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 [10], Dark Rift 2 [11], and Forge Networking Remastered [12]. All of which are available on the Unity Asset Store [13].

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 GameObjects 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.

References

[1] 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 2018.

[2] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2019. http://www.blender.org Last Accessed 1/7/2020.

[3] D. Helgason, J. Ante, and N. Francis. *Unity - Video Game Engine*. Unity Technologies, Unity Technologies, San Francisco, 2019. http://www.unity3d.com Last Accessed 1/7/2020.

[4] 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.

[5] J. LaViola, JR., E. Kruijff, R. McMahan, D. Bowman, and I. Poupyrev. *General Principles of Human-Computer Interaction*. Addison Wesley, 2nd edition, 2017.

[6] D. Mazzoni. *Audacity(R): Free Audio Editor and Recorder [Computer program]*. Audacity, 2020. https://www.audacityteam.org/download/ Last Accessed 1/7/2020.

[7] Alexander Novotny, Rowan Gudmundsson, and Frederick C. Jr. Harris. A unity framework for multi-user VR experiences. In *Proceedings of the 35th International Conference on Computers and Their Applications (CATA 2020)*. ISCA, 2020.

[8] J. Linowes. *Unity Virtual Reality Projects*, chapter 7: Locomotion and Comfort, pages 201–235. Packt Publishing, Birmingham, UK, 2018.

[9] 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, March 2018.

[10] *Photon Unity Networking 2*. https://doc-api.photonengine.com/en/pun/v2/index.html, Last Accessed 1/7/2020.

[11] DarkRift Networking. https://darkriftnetworking.com/DarkRift2/Docs/2.3.1/html/944c4100-5c17-449f-8a8e-c9fbfdaedaee.htm, Last Accessed 12/30/2019.

[12] BeardedManStudios. Beardedmanstudios/forgenetworkingremastered. https://github.com/BeardedManStudios/ForgeNetworkingRemastered, Last Accessed 1/7/2020.

[13] Unity asset store - the best assets for game making. https://assetstore.unity.com/, Last Accessed 12/30/2019.