Let's VR: A Multiplayer Framework for Virtual Reality

Alex Hansen, Kurt Andersen, Brittany Sievert
Jalal Kiswani, Sergiu M. Dascalu, Frederick C. Harris, Jr.

{ahansen2, kandersen, bsievert, jalal}@nevada.unr.edu
{dascalus, fred.harris}@cse.unr.edu

Department of Computer Science and Engineering
University of Nevada, Reno
Reno, Nevada, 89557

Abstract

Virtual reality is an ever-growing market that provides an immersive and interactive experience. With the predicted growth of this market, we wanted to meet future demands by creating a multiplayer framework for virtual reality devices, specifically for the HTC Vive. The main objective of this project was to create a framework for other game developers that facilitates the simple creation of a multiplayer virtual reality environment. Setup in a plug and play style, this framework allows engineers to focus more on the actual mechanics of their application rather than tedious networking details. To demonstrate the use of the framework, we developed a virtual reality multiplayer game that involves player-to-player interaction. The game is setup like a first-person shooter, where the cooperating players fight off multiple waves of enemies. As the game progresses, the number of enemies increases as well as the difficulty of the enemies. Score is tracked individually and is tracked on a scoreboard. The game can be played in a single player experience, but emphasis is put on the multiplayer aspect to demonstrate the framework as well as user interactions.

keywords: Networked Multiplayer, Peer to Peer, HCI, Software Engineering, HTC Vive, VR

1 Introduction

Let's VR is a multiplayer, virtual reality, first person shooter game which has been designed and implemented with Unity. The key point of Let's VR is to create a modular multiplayer framework for other software engineers to use for virtual reality projects.

In this game, users are placed into an open environment where they are pitted against waves of enemies. The user is armed with a weapon to protect themselves from oncoming enemies. In between each wave of enemies, the user is able to teleport around the environment to collect enemy boxes.

In the design process, we created a modular framework for the implementation of multiplayer virtual reality applications. It is a plug and play styled framework with Unity. The user is import the required scripts and game objects into Unity, then they will place the appropriate game objects into the specified parameters.

By creating a modular framework, this allows future developers to have an easier time working on application features, rather than the back end network communication. This will speed up the process for other developers as well making it easier for newer developers to create complex applications. Overall, the inclusion of this modular framework allows for flexibility and variability. This framework is not restricted to games, but rather, can be utilized by Unity applications.

The rest of this paper is structured as follows: Section 2 covers previous research put into Let's VR and statistics for the chosen virtual reality headset. Section 3 will discuss the software engineering aspect of Let's VR including: functional requirements, non-functional requirements, design diagrams, and detailed use cases. Section 4 covers the implementation and how Let's VR was constructed. Section 6 discusses the conclusion and future work for Let's VR.

2 Background and Literature Review

Virtual Reality is a fresh new market that deals with state of the art display systems as well as input systems.

The displays are comprised of two small screens, one in front of each eye, which gives the illusion of depth in what is being displayed. The input peripherals are tracked by senor systems that feed the location and button presses to the computer. These sensor systems track the headset as well giving fluid motion within the virtual environment[2].

In order to create a basic multiplayer framework, we first looked into which virtual reality headset would be the most appropriate to work with. According to online sales trends, the Oculus Rift sold 250,000 units in 2016, and 850,000 units in 2017; whereas the HTC Vive sold 400,000 units in 2016, and 950,000 units in 2017[14]. Aside from sales, Unity had a library available, "SteamVR," that enabled simple interaction with the peripherals. SteamVR allows use for multiple VR systems, but we found the HTC Vive interaction was simple enough. It was also easy for us to choose this because the research lab we worked in already had two HTC Vives.

Once the hardware was decided on, we shifted our research focus to the ratio of multiplayer to single player games that are already available on the public market. As of 2018, near the completion of the project, there are a total of 2351 total single-player games being sold on the Steam Marketplace that are strictly designed for virtual reality, while there are a mere 238 multiplayer games that are being sold on the Steam Marketplace that are designed for the use of virtual reality[7].

There are a number of games that we used to design our product. Some virtual reality game samples, which are all off of the Steam Marketplace, include but are not limited to: Space Pirate Trainer, The Lab, Job Simulator, Rec Room, and Spell Fighter VR[7]. Each sample provided different aspects of implementing aesthetically-pleasing and intuitive interactions to give us a solid foundation for our project development.

To provide a hands-on and interactive demonstration of the multiplayer functionality, we created a survival shooting game similar to Call of Duty's Zombie Mode[1] or Gears of War's Horde Mode[15]. As the project progressed, the demonstration was used by us to provide a visualization of progress for investors. The game was developed in Unity[16] and the implementation of the game's components was split into multiple sections that are outlined in the following section.

3 Software Engineering

3.1 Requirements Specification

The requirements detailed in the following subsections include descriptions of the functional and non-functional requirements used by the system for this project. The functional requirements detail how the system should behave, while the non-functional requirements describe the constraints on the features offered by the system [13].The elicitation process was given to us at the conception of the project by our advisor as well as other stakeholders that had a hand in the project.

3.1.1 Functional

1. **Main Menu:** A user shall be able to use a menu to interact with the application and navigate through various settings.
2. **Game Environment:** The system shall have an environment for the user to interact with during the game.
3. **Player Movement:** A user shall be able to move within the environment through the use of teleportation.
4. **Shooting Functionality:** A user shall be able to shoot a gun to defend themselves from oncoming enemies.
5. **Enemy Spawning:** The system shall spawn enemies randomly throughout the environment for the user to interact with.
6. **Enemy AI Pathfinding:** The system shall implement enemy AI pathfinding, where enemies will adapt and move around obstacles to reach their intended goal.
7. **Player Heads Up Display (HUD):** A user shall have an interactive HUD available, allowing them to keep track of ammo capacity, reserve ammo, total health, the current enemy wave, and their score.
8. **Enemy Waves and Increased Difficulty:** The system shall implement enemy waves that increase in number and difficulty as the game progresses.
9. **Weapon Accuracy:** A user shall experience realistic weapon accuracy throughout the game, implementing functionality such as bullet drop creating a more realistic experience.
10. **Item Collection:** A user shall be able to collect items that randomly drop from enemies. These items will give the user additional health, ammo, or power-ups that give the user a temporary advantage.
11. **Networking Functionality:** The system shall implement multiplayer functionality, allowing users to interact and play the game with others.

3.1.2 Non-Functional

1. **Unity Game Engine:** The system uses Unity, as the entire project was built using this engine.

2. **Performance Requirements:** The system shall operate using the HTC Vive. Users will need an Intel Core i5-4590 or AMD FX 8350 equivalent or better processor, an NVIDIA GeForce GTX 970 or AMD Radeon RX 480 equivalent or better graphics card, and at least 4 GB RAM [19].
3. **Platform Constraints:** The system shall operate on Unity in conjunction with the Steam platform to get the VR capabilities from the HTC Vive.
4. **Reliability:** The system shall operate such that the hardware and software that the user interacts with must be robust and intuitive throughout the entire experience.
5. **Security:** The system shall operate such that the user does not come into any physical harm while playing the game. This is ensured by limiting user actions and creating visuals in the game to prevent any harm from coming to the user.

3.2 Detailed Use Cases

The Use Cases describe the interaction between users and the system as seen in Figure 1. The numbering order for each Use Case shown in Figure 1 are described in further detail in this section.

1. **Choose Role:** The player will be able to choose an in-game role that will provide bonuses to the player.
2. **Choose Mode:** The player has different choices on how the game can be initiated. Once the choices have been made, the main portion of the Lets VR application begins.
3. **Single or Multiplayer Mode:** A player has the choice of initiating the game through a local instance or by playing over a network.
4. **Manage Game:** This gives the player options to configure the game correctly to their system for a streamlined experience.
5. **Start Game:** This allows the player to resume playing the game if it has been paused or in another state.
6. **Pause Game:** While the player is playing the game, the play has the ability to stop the game and take a break.
7. **End Game:** When a player is done playing the single player or multiplayer game, the player can select this option to return to the main menu.
8. **Play Game:** This allows the player to start the game after finalizing all options. All player interactions shift from the menu to the main game.
9. **Shoot Enemies:** The player is able to defend themselves from oncoming enemies in the game. The player will have limited ammo.

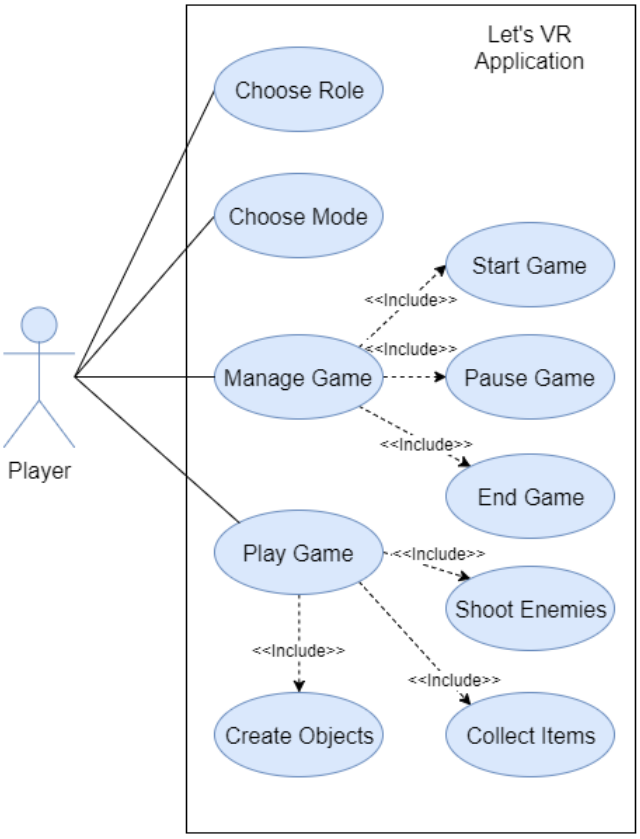

Figure 1: Use case diagram for Let's VR demonstrating the flow of interactions within the system.

10. **Collect Items:** As enemies are eliminated by the player, they drop items that can be collected by the player at the end of the enemy wave.
11. **Create Objects:** During intermediate rounds, the player will be able to create objects that obstructs enemy pathing.

4 Implementation

At the initial stages of the project's development, we utilized the documentation outlined in Section 3 to create an agile development process that would adapt to future development hurdles as well as reinforce team oriented coding practices.

To ensure these goals, software components that needed to be developed were assigned to individuals based off of interest, rather than programming skill. In regards to project and team understanding, we utilized weekly meetings that involved all team members to analyze progress and possible roadblocks of the project. Lastly, pair programming[10, 13] was utilized as much as possible to ensure the project's modules worked in a cohesive manner and to ensure each team member

understanding of the developed system.

Once team dynamics were outlined and addressed, we began developing a single player environment. This direction would allow us to ensure all functionalities would be implemented and perfected before multiplayer functionality was incorporated. Once single player was optimized, we began to merge the system into multiplayer modules. We utilized test driven development strategies comprised of unit, component, and integration tests to ensure the migration was successful and met the project's specifications[10, 13].

4.1 Unity Integration

4.1.1 Game Menu

To ensure a player can navigate the game effectively, we created a game menu that is loaded and displayed at the start of the application. Figure 2 displays a snapshot of the menu players interact with. This menu allows the player to initiate a single player or multiplayer game, see their high score, initiate a tutorial, or change in game settings. Additionally, when a player is in the game, there is a menu the player can activate to exit the game.

Figure 2: When the application is initialized, the main menu is loaded for player usability.

4.1.2 Game Environment

To ensure an immersive experience, we took the initiative to incorporate as many features as possible to mimic the real world. In game features such as lighting, wind, sounds, and interactive objects were incorporated and optimized to the best of our ability. Although many of these features are not a necessity, these small details play a big role in creating an immersive experience.

4.1.3 Artificial Intelligence (AI)

To provide a goal for the players taking part in the game, robot enemies were implemented with the goal of eliminating the player. There are a total of four different enemy types, two of which utilize melee attacks and the other two utilize ranged attacks. The two melee enemies are shown in Figure 3. To ensure the system acts in a realistic manner, we utilized rigs that were fully animated to provided walking, attacking, and death animations. Additionally, we used NavMesh Agents[17] and NavMesh Baking[18] to allow the AI to navigate the world to reach the player. Both of these systems were crucial to provide a realistic experience for players when autonomous system mimic the real world.

Figure 3: Two of the enemy types that players will encounter when playing the game.

4.1.4 Networking Interface

When implementing networking in Unity, we utilized `C#` scripts as well as multiple game object settings that interacted with the Unity's development environment. Additionally, we utilized a Peer-to-peer networking architecture[9] when implementing the multi player framework. This architecture was chosen due to the resources we had as well as our architecture analysis.

4.2 Libraries Used

We make use of a few pre-made libraries within Let's VR. We use these libraries because we did not want to rewrite preexisting code that already functioned well. This allowed us to focus more on the construction of the framework rather than the functionality of the game.

MultiuserVive[3]: When developing the multiplayer system, we utilized open source code that helped integrate with some of Unity's multiplayer components. Unity incorporates multiplayer capabilities and this library enabled us to better understand and implement our goals. An in game multiplayer avatar was provided

Figure 4: When in mulitplayer mode, the players are displayed as the white figure.

in the library for players to see and interact with eachother. This multiplayer avatar is shown in Figure 4 as a white circle.

Player Teleportation[3]: During the initial development of the project, we utilized this library to get a better understanding of how to develop a single player experience.

4.3 Assets Used

The assests that we use in Let's VR add to the aesthetic of the game. It makes the overall product look nicer, and made development of the game easier. This allowed us to set the majority of our focus on the construction of the framework for Let's VR.

Enemy Models[5]: These models were selected due to them being rigged with animations. The use of the animations provide an immersive experience when utilized with other in game assets.

Skybox[11]: With the game taking place in an outdoors setting, a skybox was used to represent an outdoor theme.

Curved Menu[6]: Through multiple VR menu tests, an in game asset was needed to provide a menu feature. Utilizing an in game object as well, rather than attached to the player's screen, as a curved screen provided intuitive player understanding.

Nature Environment[12]: To ensure a consistent outdoors theme, nature assets were used provide a diverse and organic environment for players.

Grass Models[21]: This asset was used due to the original grass game object not fitting the theme the game was trying to achieve.

SteamVR Plugin[8]: To interact our Virtual Reality (VR) hardware with Unity[16], we incorporated this asset to bridge that gap.

M16 Gun Model[20]: To provide an intuitive understanding by the player, a simple gun model was utilized which made it easier to point and shoot towards the enemy robots.

Game Sounds[4]: To provide an immersive experience, a wide range of sounds was used to ensure actions that occurred in the environment or by players would be representative of real world interactions.

5 Discussion

Let's VR was showcased at a yearly event called Innovation Day hosted by the University of Nevada, Reno. This event usually has close to five thousand guests in attendance. Amongst the guests, approximately 150 guests interacted with Let's VR. These interactions told us about the usability of our system.

Let's VR was setup as an individual booth with a singular setup to allow guests to play the game. Throughout the entirety of the event, the game was in constant use. Because of the framework we created, Let's VR did not run into any issues. The robust design of the framework ensured a fulfilling test of Let's VR. The only issue that ever arose is a minimal amount of latency between each user.

Nearly every person that used Let's VR at Innovation Day gave positive feedback. Users enjoyed the game and were not affected by motion sickness. Some of the feedback received was:"The game was fun and difficult," "The game ran smoothly and the controls were intuitive," and "The friendly player seemed to be lagging slightly in their movements." There was some latency issues due to network setup, but there was never a loss of connection between the players.

6 Conclusion and Future Work

The majority of features from the project's conception were implemented through the current lifetime of this project. There were minor features that were not added, but the majority of these were more for gameplay quality rather than for networking performance. Some of the gameplay features that were not implemented include but are not limited to: individual player roles, polished player models, aesthetically-pleasing user interface, and interactive tutorial.

6.1 Conclusion

The novelty factor of this project is the networking framework being geared towards modularity rather than being custom built for this project specifically. Another novelty factor is that this is a multiplayer virtual reality game, where as noted earlier in Section 2, the number of multiplayer virtual reality games is extremely lacking when compared to the number of single-player games.

By having a modular networking framework, other game developers will be able to create their own instances of multiplayer virtual reality games or applications with ease.

Given more time to work on this application, we plan to address the issues stated in the Section **??**. We also wish to add voice communication between the users within the application. Including this will add another layer of immersion as well as making it easier for the players to communicate between each other. As of right now they can only gesture at each other.

Another major functionality that we wish to change is the way the network is set up. The network is set up in a peer-to-peer style. We would like to change to a server-client setup for ease of communication between all users. This will be especially useful when more than two users will be using the application at once.

Acknowledgement

This material is based in part upon work supported by the National Science Foundation under grant numbers IIA1329469 and IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Activision. In *Zombie Mode*, 2018. [https://www.callofduty.com/wwii/zombies n.d. Last Accessed May 10, 2018].

[2] Christoph Anthes, Rubén Jesús García-Hernández, Markus Wiedemann, and Dieter Kranzlmüller. State of the art of virtual reality technology. In *Aerospace Conference, 2016 IEEE*, pages 1–19. IEEE, 2016.

[3] Pascal Auberson. In *Multiuser Vive*, 2016. [https://github.com/pauberson/MultiuserVive n.d. Last Accessed May 5, 2018].

[4] Audioblocks. In *Storyblocks*, 2018. [https://www.audioblocks.com/ n.d. Last Accessed May 10, 2018].

[5] POLYGON BLACKSMITH. In *Battle Droids Pack*, 2018. [https://assetstore.unity.com/packages/3d/characters/humanoids/battle-droids-pack-74088 n.d. Last Accessed May 10, 2018].

[6] CHISELY. In *Curved UI*, 2018. [https://assetstore.unity.com/packages/tools/gui/curved-ui-vr-ready-solution-to-bend-warp-your-canvas-5325 n.d. Last Accessed May 10, 2018].

[7] VALVE CORPORATION, 2018. [https://www.store.steampowered.com/ n.d. Last Accessed May 13, 2018].

[8] VALVE CORPORATION. In *SteamVR Plugin*, 2018. [https://assetstore.unity.com/packages/templates/systems/steamvr-plugin-32647 n.d. Last Accessed May 10, 2018].

[9] PC Magazine. In *Encyclopedia*, 2018. [https://www.pcmag.com/encyclopedia/term/49056/peer-to-peer-network n.d. Last Accessed May 10, 2018].

[10] Robert C. Martin. A code of conduct for professional programmers. In *The Clean Coder*. Prentice Hall, 2011.

[11] MGSVEVO. In *Classic Skybox*, 2018. [https://assetstore.unity.com/packages/2d/textures-materials/sky/classic-skybox-24923 n.d. Last Accessed May 10, 2018].

[12] SHAPES. In *Nature Starter Kit 2*, 2018. [https://assetstore.unity.com/packages/3d/environments/nature-starter-kit-2-52977 n.d. Last Accessed May 10, 2018].

[13] Ian Sommerville. In *Software Engineering*. Pearson, 2016.

[14] Statista. In *Worldwide virtual reality (VR) headset unit sales by brand in 2016 and 2017 (in millions)*, 2018. [https://www.statista.com/statistics/752110/global-vr-headset-sales-by-brand/ n.d. Last Accesed May 13, 2018].

[15] Microsoft Studios. In *Horde Mode*, 2018. [http://gearsofwar.wikia.com/wiki/Horde n.d. Last Accessed May 10, 2018].

[16] Unity. In *Unity Game Engine*, 2018. [https://unity3d.com/ n.d. Last Accessed May 10, 2018].

[17] Unity. In *NevMesh Agent*, 2018. [https://unity3d.com/learn/tutorials/topics/navigation/navmesh-baking n.d. Last Accessed May 10, 2018].

[18] Unity. In *NevMesh Baking*, 2018. [https://docs.unity3d.com/Manual/class-NavMeshAgent.html n.d. Last Accessed May 10, 2018].

[19] VIVE. 2018. [https://www.vive.com/us/ready/ n.d. Last Accessed May 13, 2018].

[20] Warhead3D. In *M16A1 Assault Rifle*, 2018. [https://www.turbosquid.com/3d-models/m16a1-assault-rifle-m16-3ds-free/523322 n.d. Last Accessed May 10, 2018].

[21] PATRYK ZATYLNY. In *Hand Painted Forest Envrionment Free Sample*, 2018. [https://assetstore.unity.com/packages/3d/environments/hand-painted-forest-environment-free-sample-35361 n.d. Last Accessed May 10, 2018].