Multi Device Multi Player Digital Game with Virtual Reality

Riddhi Kasliwal | UID: 2013516917 | HKU - COMP4801 | 16-04-2017
Supervisor: Dr. Choi Loretta
Second Examiner: Dr. TW Chim
ABSTRACT

This report describes the design, implementation and evaluation of a project which explores new ways of multiple user interaction for a Virtual Reality application via a multiplayer game. It attempts to overcome the limitation of Virtual Reality (VR) applications which conventionally allow only one user wearing the headset to interact with the system. The game application aims to achieve a multiplayer effect by allowing one player to use a VR headset and other players to connect to the game via their mobile devices. The game design helps the users to get immersive experience based on the device they use to play the game. Therefore, mobile phone players would have different role in the game compared to the VR player.

This report discusses and analyses the game development process. There were many problems involved in different parts of the process. This report also attempts to explain the modifications which were needed from the original design to solve the problems and deliver on time.
ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Loretta Choi for helping me with this project’s idea and providing me with timely guidance. I am grateful for help received from Laurence Wong in setting up the project.

I would also like to thank all the people from various academic background for helping me test the game on regular basis and give me valuable feedback. Lastly, I would like to show my gratitude to Professor Ken Ho for guiding me in report writing.
TABLE OF CONTENTS

ABSTRACT ................................................................................................................................. 1
ACKNOWLEDGEMENTS ............................................................................................................ 2
LIST OF TABLES AND FIGURES ............................................................................................. 4
  TABLES ................................................................................................................................. 4
  FIGURES ............................................................................................................................... 4
INTRODUCTION ....................................................................................................................... 5
BACKGROUND ........................................................................................................................ 5
RELATED WORKS & MOTIVATION ......................................................................................... 5
OBJECTIVE ............................................................................................................................. 6
SCOPE ...................................................................................................................................... 6
METHODOLOGY ...................................................................................................................... 7
  DESIGN ................................................................................................................................ 7
  HARDWARE AND SOFTWARE .............................................................................................. 9
IMPLEMENTATION .................................................................................................................. 11
  UNITY GAME ENGINE ....................................................................................................... 36
SCHEDULE ............................................................................................................................... 38
  TIMELINE ............................................................................................................................ 38
FUTURE WORKS AND IMPROVEMENTS ............................................................................. 39
  IMPROVEMENTS ............................................................................................................. 39
  FUTURE ADDITIONS ........................................................................................................ 40
CONCLUSION .......................................................................................................................... 41
GAME MANUAL ....................................................................................................................... 42
  COMPATIBLE DEVICES .................................................................................................. 42
  GAME CONDITIONS ....................................................................................................... 42
  HOW TO PLAY .................................................................................................................. 42
REFERENCES .......................................................................................................................... 43
# LIST OF TABLES AND FIGURES

## TABLES

Table 1: Timeline For The Project..............................................................38

## FIGURES

figure 1: Vive Controller Description..........................................................10
Figure 2: Player Avatars. (Left) Mobile Player. (Right) Vr Player ..................12
Figure 3: First Person View For The Vr Player..............................................13
Figure 4: Humanoid Vr Player Rig In T-Pose .............................................15
Figure 5: Camera Attached To The Head Of The Player .................................16
Figure 6: Movement Of Hand And View Of The Avatar .................................17
Figure 7: Third Person View For The Mobile Phone Player .........................18
Figure 8: Bumps In The Scene.......................................................................20
Figure 9: Dna Samples - Female (Left), Male (Right) ....................................21
Figure 10: House For The Elves.................................................................22
Figure 11: Vr Player’s Space Ship...............................................................22
Figure 12: Photon Server Architecture ......................................................23
Figure 13: Usage Of Photon Unity Network ...............................................24
Figure 14: Waiting Room Screen For The Vr Player .....................................26
Figure 15: Mobile Player Functions And Ui ................................................28
Figure 16: Maths For Movement Button To Resemble Joystick ....................29
Figure 17: Circular Ring Around Bomb ......................................................30
Figure 18: Particle Effect On Picking Up Objects .........................................32
Figure 19: Ui Panel For The Vr Player.........................................................33
Figure 20: Scene With Less Objects To Test Network Lag .........................35
Figure 21: Network Lag..............................................................................35
INTRODUCTION

BACKGROUND

Virtual Reality is an experience of completely immersing yourself into a virtual world. People frequently achieve/do things that they cannot imagine about doing in the real world using Virtual Reality for immersion experience. There are various levels of immersion for VR. For example, devices such as Google Cardboard which use mobile phone as a source of VR do not detect hand movements, hence the applications are limited to using mobile phone sensors. The other devices such as HTC Vive, Oculus Rift are accompanied with controllers to detect hand movements of the users. This significantly increases the variety of applications that can be made for Virtual Reality.

For the VR applications developed for such devices, the major limitation is that, only people wearing the headset can see the VR world. People surrounding the person wearing headset cannot get the feel of what the VR user is experiencing. Addressing this issue, this project aims to develop a multiplayer game which allows users to interact with the Virtual World using their mobile devices. So, any number of people can be part of one experience. This experience will, however, depend on the type of device the user is using.

RELATED WORKS & MOTIVATION

Currently, there are a few multiplayer applications for Virtual Reality such as Redout, Think Space [1]. Redout is a multiplayer game which is desktop based with an optional VR headset – Vive or Oculus Rift support. It has online multiplayer compatibility which lets the players connect and play with anyone in the world. Think Space is only for VR headset – Vive and it also has online multiplayer compatibility. So, users from anywhere in the world who have Vive, can connect to the game and interact in the Virtual World.
The limitations of games such as Redout is that it is not designed to give experience to the user based on the type of device the player uses. Therefore, the desktop and the VR player have the same role which leads to questioning the role of VR support for this game. For games such as Think Space which are designed only for VR headsets, people without the headset cannot be part of the experience.

Considering these different type of multiplayer games, this project attempts to solve the issues mentioned above by introducing unique game design which assigns different roles to the users based on the device they are using. They can choose between VR headset and mobile device to join the game and get a different experience according to the device used. This project will, as a result, explore the interaction of the camera positions and algorithms for different devices to give immersive experience accordingly.

**OBJECTIVE**

The main objective of this project is to develop a working game application which explores a way for the mobile device users to interact with a player wearing a VR headset in the same virtual experience. It has different camera angle, and game logic for the different devices.

**SCOPE**

The features of this project for the first release will be:

- Different User Interface for mobile and desktop (VR).
- 2-5 player compatibility (1-4 Mobile players, 1 VR player).
- Free movement and rotation around the game scene.
- Potions, health, enemies, bombs for competition among the players.
- Sound effects to indicate different actions carried out by the players.
However, this version of the game would not include:

- Customization of player avatar or the scene.
- Chat functionality for the users. Therefore, they need to be in the same physical room to collaborate in the game.

The final deliverable is a working game application with immersive game design and a demo for people to experience the Virtual Reality game.

**METHODOLOGY**

This project was developed using one of the agile practices, Extreme Programming (XP). This method follows a disciplined approach to deliver high quality software continuously in stages. This is achieved by frequent feedback, continuous testing and planning to deliver a working application every 1-3 weeks.

The timeline for this project was set using XP and is described later in this report. Using the XP approach, every week, a list of tasks with number of hours required to finish it were estimated. The time estimation was adjusted based on the progress made every day. This approach therefore made sure that there was a deliverable every end of the week.

**DESIGN**

The original design for the project was to include compatibility for two VR players with 8 mobile device users. The mobile phone players join the team of one of the two VR players. The idea was inspired by some post graduate student’s game application, *Save The Pumpkin*, which explores networking between two HTC Vive headsets connected in the same room. That game is themed around Halloween where two players appear as giants in virtual environment. There is a wooden plank placed at a certain height over the city. The two players need to reach for the pumpkin which is placed in the middle of the plank. There is leap motion sensor attached over the headset which allows the game to track the hand
gestures whereas the two controllers are attached to the feet to detect movements. Different hand gestures initiate different abilities which are used to defeat the other player. The first player to touch the pumpkin wins the game.

However, the original idea for the final project which was to support 8 mobile devices and 2 VR headsets was dropped because it was not feasible to complete this project within the given time. Also, since there was only one developer working on this project, testing the game application would not have been ideal. Therefore, the scope for the project was reduced to 1 VR player and the game is extended to be compatible with multiple mobile devices (1-4). The mobile phone users will have different view of the game compared to the Vive user (VR). The mobile phone users can see the VR players’ avatar on their phone with a third person view.

The early development of the game used models from the game mentioned above – Save The Pumpkin. The VR player acts as a witch and she needs to find good pumpkins hidden by some minions, who are the mobile phone players, around the city. The witch cannot see all the minions at once but the minions can see the witch always since they would have a third person view of the city on their screens. The minions will have to predict the behavior of the witch and lay traps accordingly. This game design was discarded based on user feedback about the city model being too unreal for the user to get the VR experience. Also, the scene was rendered very slowly on the mobile devices as their processing power is not sufficient to render high poly model in real time without lag.

After some different types of scene render testing on both mobile and Vive, a forest asset from the Unity asset store was selected [2]. Some design feedbacks from the users and social issues exploration inspired the final version of the game - The Earth had a nuclear war due to an Alien race and the entire human race was wiped out. Some Elves were keeping various kinds of human DNA safe in a galaxy far away. They created a healthy Earth like forest environment on an asteroid in the far away galaxy. An Alien named Kiro, from planet Jaba learnt about the
Elves and sent one of their kind to investigate and send back the DNA to Jaba, if found. He was also given permission to kill anyone who comes in between.

Therefore, Kiro has a laser pointer in its eyes but he cannot estimate the strength of the Elves. The Elves however have prepared bombs for such alien invasion and they can kill anyone or anything in one hit. But they need to be careful about detonating the bomb too close to the DNA samples. Both the players have free movement around the scene.

**HARDWARE AND SOFTWARE**

**Hardware:**

This application will be best experienced if the following devices are used:

1. HTC Vive – HTC Vive is one of the best VR headsets in the market and it comes with 2 controllers and 2 base stations. The sensors need to be placed diagonally from each other with a maximum distance of 5m. The room where the Vive is set up needs to at least measure 2m x 1.5m [3].

   **Vive Controllers:**
Figure 1 shows a table of the functions of the interactive options on the controller. For example, the part of the controller marked as 1, is a menu button. For this project, the menu button (1), trigger (7) and trackpad (2, 3, 4, 5, 9) will be used. The player can press the trigger button and point their head in a direction he/she wants to shoot, and he/she will be able to fire towards that location. The trackpad would be used to move around the scene. It would work like a joystick and pressing the trackpad near the DNA samples will pick them up. For both the controllers, the tracking sensor gives information about the position of the controller in the room to the game engine. This is done with the assistance of the base stations which are set up in the room to track the positions of the controller.

2. Android/iOS mobile devices – Any Android device with Lollipop 5.x and above with 2GB RAM would give the best results for playing the game as this game is not optimized for render in real time for slow processing devices.

3. Windows desktop PC or laptop [4] – HTC Vive is compatible only with Windows at this stage and therefore, for this project, only Windows PC
would be well-suited. The absolute minimum requirement of Windows system is:
Graphics processor: Nvidia GeForce GTX970, or AMD Radeon R9 290 equivalent or greater.
CPU: Intel i5-4590 or AMD FX 8350 equivalent or greater RAM: At least 4GB
Video output: HDMI 1.4 or DisplayPort 1.2 or newer
USB port: One USB 2.0 or greater
Operating system: Windows 7 SP1 or newer

Software:

1. Unity 3D engine – Unity is a powerful engine to develop games and get different executable file for various type of devices. Therefore, this project is developed on Unity.
2. Unity Steam VR asset – This asset implements basic Vive related methods for new developers to use and modify. This asset is necessary for developing any application to run on Vive as it provides the basic setup for connecting the Unity Engine to HTC Vive.
3. Photon engine [5] – This is used for networking between the different type of devices as it is very powerful and has a lot of features like voice chatting for future development improvements of this project.
4. Instant VR Free – This asset helped the VR player’s avatar and its synchronization over the network. This report discusses the use of this asset in detail in the next section.

IMPLEMENTATION

This project has completed the first version of its development by producing the deliverables mentioned in the Introduction section. For discussion, the implementation of this project can be divided into various parts – Player Avatars,
Scene Setup (Lighting, Camera Angles), Functionalities, Networking and Testing. This section describes the development in detail in these areas:

**Player Avatars:**

The VR player and the mobile device player have different avatar in the game. Each mobile phone character has unique name tag on their head for everyone in the game to identify them.

*Figure 2: Player Avatars. (Left) Mobile Player. (Right) VR Player*

As seen in Figure 1, the left character is the mobile player. It has the tag - Roa. Aro, Kai, Dym and Roa are the 4 mobile phone characters. Kiro is the Alien character which is on the right in Figure 2. Each Mobile character have a name tag attached to its head. So, when the character spawns in the Virtual environment, VR player and other mobile players can see its name tag.

These characters are free to use on Unity Asset store and they were selected because they fit the storyline for the project. The sword in the hands of mobile phone character represents that they are always ready to protect the DNA samples.
Scene Setup:

For each type of player, their view of the game world is different. VR player gets a first person view whereas the Mobile phone player gets a third person view. This is in accordance to the best capabilities of the devices they are using to join in the game.

VR Player -

![Figure 3: First Person View for the VR Player](image)

Figure 3 shows the view for the VR Player. For Virtual Reality applications, the best experience is when the player feels that he is part of this alternate reality. Therefore, the first-person view is the ideal option for the VR Player.
From user feedbacks, the view from the first-person camera in this virtual environment feels real which justifies the purpose of using this scene and it also fits the storyline of this project.

For this first-person camera for HTC Vive, a Steam VR object is used. A player object avatar, alien’s avatar, is attached to the camera part of this object to show the avatar of the VR player. So, when the player moves, the avatar will move with it. But, this also implies that if the player is looking up, the VR player’s avatar’s entire body would swing up in the scene. So, from mobile device player’s perspective, that would feel unnatural. To tackle this issue, only the head part of the avatar should be attached to the camera whereas the rest of the body should be still and should not be in the air if the VR player is looking up.

For this project, an asset which was mentioned above in Software section, Instant VR, was used. The asset, uses humanoid T-pose avatar and uses the head part of the object for rotation and the rest of the body changes its posture based on the VR player’s physical height. So, if the player is physically sitting, the avatar would bend its legs in the scene.
Figure 4: Humanoid VR Player Rig in T-Pose

Figure 4 shows the rig model of the VR character. For a humanoid model, at least 15 bones should be identified from the rig. Instant VR asset identifies these bones and joints, and automatically triggers walking or sitting animation when the player walks or sits. It calibrates the height of the player on start of the game, so when the user’s height is below that height, it starts to crouch based on this rig. Therefore, when the player first joins the game, player should wear the headset and press tab to set up the height according to his physical height.
Figure 5: Camera attached to the head of the player

This asset, programmatically adds the camera to the head part of the rig so that when the user rotates his head, for example by looking up, only the head of the avatar looks up as seen in Figure 5.

This approach also tackles the issue of feeling nauseated in the Virtual World as reported by multiple users [6]. The players feel nauseated because of the disparity in both the realities. When the player moves physically, he expects to see a walking animation in the Virtual Reality. If there is an object that behaves like the physical behavior, the nausea can be overlooked. Some approaches therefore add a nose to camera so that the VR player has a constant object to relate to the physical reality.
In the approach taken for this project, the VR player can relate to the body and its movements. So, when the player looks down, he will get the view of his hands and legs as shown in Figure 6.

The nausea will be further reduced if the application supports the use of the models for the controllers. Because, they are sensed in the base stations and the position and rotation of the controllers reflect the physical position and rotation from the camera. But when using the asset – Instant VR, there were many problems with the configuration of the controller models. The free version of the asset does not come with a set up for the Vive controllers. As a result, when the player is spawned in the game, he has a Steam VR object with the controller support is attached to its prefab. But for this project, Vive controllers are needed to carry out the actions in the game. Therefore, an additional Steam VR object without camera is added to the scene which supports the controllers. But the models for the controllers are hidden because the position and rotation of this second SteamVR object cannot be made accurate in relation to the camera position and has huge discrepancy. To solve this problem, only one controller is
used by the alien and the actions taken by the alien are made to feel like response to stimuli rather than being controlled by a remote controller. For example, the laser beams shot by the alien are fired on the trigger press by the player on the controller but are simulated to feel like they are shot wherever the head camera is pointed to.

Also, the asset works perfectly only when the player is positioned at Y=0. But in this project, there are many bumps and therefore, the character kept bouncing up and down. This was very unpleasing as the camera moved up and down rapidly giving a nauseated feeling. The support team for this asset responded quickly and helped to fix this problem.

Mobile Player –

![Figure 7(a): Third Person view for the mobile phone player](image)

*Figure 7(a): Third Person view for the mobile phone player*
Figure 7 (a) shows the camera angle and view of the Mobile player. Figure 7 (b) shows the multiple player view. Each player has their name tag visible to other players to identify who is nearby. Since this project does not support online chat function, the players need to be physically present in the same room to play the game. And to identify each other, the name tags are useful.

User feedback about the camera angle led to the angle shown in Figure 7 (a). Most of the 3D games for mobile phone have this angle therefore the third person camera for the mobile players in this game help the players to adapt to the game mechanics easily.

Lighting & Terrain Map—

The scene in the game is made to get the most out of a virtual environment for the VR player. The Terrain map has bumps and the grass have an animation to make it look more natural. Also, the direction light is used to light up the scene.
and imitate Sun Light. From game story’s perspective, it is done in this way to make the DNA samples be preserved in the Earth-like condition. The light color can be changed to imitate night effect but it makes the experience dull. From user feedback, night effects demand the game story to be a little scary. Therefore, the direction light is imitating sun light in the game environment.

Figure 8: Bumps in the scene

Figure 8 shows the bumps in the scene. When some users first tried moving in the scene, the bumps in the scene gave them realistic feeling of a virtual environment. This is because, when the user walks on the bump, the camera follows the players height from the ground which increases and decreases according to the bump, giving a natural experience. A flat terrain on the other hand will not be able to give the same effect. Although some users felt nauseated but that made their experience real.
The objects that the VR player is supposed to collect look like the images in Figure 9. The left side image represents female DNA samples and right side image represent male DNA samples. There will be 3 containers for the female samples and 2 containers for the male samples. The reason for this is that the females play a major role in reproduction. These samples are randomly allocated in the start of the game by the first player who starts the game.

The other objects instantiated in the scene are some shooting effects, particle effects and bombs as required by the game design.

There are also some static objects in the scene like Spaceship and a house for the player’s reference.
The objects shown in figure 10 and 11 are important to the story of the game but their main purpose is to act as static objects which allow the Mobile player to
communicate with each other about where in the scene they are with respect to these objects.

**Networking:**

Since this game is a multiplayer game, networking is an essential part of the project. Photon Unity Network asset is used for the networking. This asset is powerful and easy to use.

![Photon Server Architecture](image)

*Figure 12: Photon Server Architecture*

The photon cloud server registers players’ basic information and adds him to the list of players. For the first player who joins the game, the server makes him the master client by connecting it to a master room and all the other player who join the game connect through this client. As seen in the Figure 12, all the actions taken by the players are first sent to the master room, and then the game state is updated on the client. This updated information is sent to all the players connected to the master room. The game server will be responsible only for the creation of master room. In this project, photon engine, a plugin for Unity is used to implement the game server. This engine is used because it can handle the
server logic quite well for desktop and mobile multiplayer games. It generates a unique code for ‘n’ number of devices. These codes are sent to the ‘n’ devices who want to connect to the game. It handles the actions performed by these devices by their unique codes to differentiate among different players.

Any component which needs to be networked and shown in all the users instance should have a photon view component on them.

![Photon View (Script)](image)

*Figure 13: Usage of Photon Unity Network*

Figure 13 shows how the Photon Unity Network is used for each object which needs to be instantiate across all the instances connected to a game. Photon View Script is compulsory for object instantiation across the network. Photon Transform View Script is needed for updating the transform components – position, rotation and scaling across the network. Therefore, objects like Player Avatars, DNA Samples, Bombs, Particle Effects all have a photon view component on them. This greatly reduces the number of objects shared across the network.
The important thing to take note when networking different kinds of players is that the components exclusive to the player who instantiated their character should not be active in other player’s instances. For example, when instantiating first mobile player – Aro, the player movements component, the UI component, and camera component should be inactive in all the other player’s instances. And should be programmatically set active in the player who instantiated Aro. The same applies to the camera component for the VR player. If the components are not inactive in other player’s components, Unity randomly picks one of the cameras from all the active cameras which produces undesirable results.

While networking, the scheduling of the update function from all the components of the game could be unreliable. Therefore, the logic about setting components active and not active should be kept in update function and be synchronized to get the expected behavior. For this project, some static variables were used to synchronize the behavior.

**Functions:**

**Start Screen –**

When the game starts, the player has the option to start the game, quit the game and mute the sounds in the game.

The player can see “connecting” text once he clicks play to connect to the master client and the room. He is assigned a unique player ID based on the sequence number of his connection. The game logic asks the player who connected first to be the VR player. Since any number of mobile phone players ranging from 1 to 4 can join the game. The limit set for maximum number of players in the room is set to 5. Therefore, if at the same time, more than 5 players connect, the 6th player would be the VR player in a different room.
Waiting Screen –

Once the player connects to the game server, he is taken to the waiting room. Where he can see the name of the players who joined the game.

*Figure 14 (a): Waiting room screen for the VR Player*

*Figure 14 (b): Waiting Room view for the first mobile player and VR player after the first mobile player joins the game.*
As seen in figure 14 (a), since the VR player will always be the first player, the waiting room will show just his name, Kiro. When the first mobile player joins in, the VR player would be able to see his name get updated on the screen and so on, as seen in Figure 14 (b). This screen makes sure that each player knows how many players have joined the game. Once any player clicks the Start button, the game for all the players would start simultaneously.

The functionalities of the VR player and Mobile player are different and are designed in a way that it is fair for both the players. For example, VR player would die with one bomb hit and Mobile player would die with 5 right hit. Also, the movements for both the type of players is through joystick like controls. The speed is also similar but since there can be only 1 VR player, he is given the option to shoot from distance. And as there can be multiple mobile players, each has only 3 bombs that they can use to deal fatal damage to the VR player. Also, the environment in the game allows the mobile player to place the bomb and hide behind the trees to avoid getting hit.

The fairness of the game was tested based on user feedback and user tests. The number of times mobile player won and the number of times the VR player won was counted and it was concluded that the game is fair according to the number of functionalities for each.
Mobile Player –

![Game Interface](image)

**Figure 15: Mobile Player functions and UI**

1. Joystick Movements and Rotation: Mobile phone user has joystick like buttons for movement and rotation in the scene as shown in Figure 15. The left-hand side button is for movements and the right-hand side button is for rotation of the players view angle in the scene. These buttons with their animations to imitate joystick is taken from an asset called Ultimate Joystick from the unity asset store [7].
Figure 16: Maths for movement button to resemble joystick

The 4 directions are marked by the four co-ordinate points as seen in Figure 16. When the joystick is pressed in one direction, the acceleration vector multiplied by the magnitude of the co-ordinate points is applied to the position of the player. Similarly, the rotation of the player is calculated.

2. UI: The aim of the mobile players is to protect the VR player from taking the DNA samples. But if they place a bomb too close to the Sample, the sample will be blown away too. Therefore, they need to strategically place the bomb. As seen in Figure 15, they can see the number of samples left in the scene. They can see the number of bombs left with them and their health. Their health is affected by 2 things, standing too close to the bomb which is detonating or getting hit by the bullets fired by the VR player. While implementing the logic for the UI elements, one important decision that needed to be made is if it is appropriate to make the UI Canvas be the child object of the player or as a static element in the scene. Because, if it is static, everyone has exactly one copy of the UI Canvas. But if it is a child object, each player would get a copy of the element for all the players even if the element is disabled. By making the UI canvas static, when hit, it showed unpredictable behavior. For example, if Aro was hit, health points of all the mobile players in the scene would get affected. Therefore, the better design strategy is to use the Object-Oriented Programming and
make the UI canvas element, a child object of the player itself and upon hit, the child element can be referenced easily.

3. Bomb placing and detonation: The Place and Detonate Buttons on the UI are used for placing and detonating the bombs as seen in figure 15. Once the player clicks place, a bomb is placed at his location and there is a sparkling circle around the bomb to warn other players that there is a bomb placed as shown in figure 17.

![Figure 17: Circular ring around bomb](image)

The player then can go around 2 units radius away from the bomb to detonate it. He should detonate the bomb within 30 seconds otherwise the bomb will get destroyed from that location. When the player presses the detonate button, all the player within 1.5 units from the bomb get hurt. If the VR player is within 1.5 units, he will get a fatal hit. Each player gets 3 bombs to use, therefore they need to collaborate and play the game to kill the VR player.

There is a script attached to the bomb object which checks for any player near it and gives them damage accordingly using the Object-Oriented Programming strategy.
VR Player –

1. Movements: The VR player will use only one controller in this game to do any type of action in the game. In this game, the trackpad is used for the movement. The logic behind the movement works like in Figure 16. The player rotated and changes its movement direction based on the camera view angle. This approach makes the walking smooth and natural. A few users who have used HTC Vive HMD before complained that this type of movement is nauseated and it does not give good experience whereas some users who never experienced any application in HTC Vive enjoyed the smoothness and natural way of walking in the environment using the trackpad. For this project, the trackpad movement was picked simply because some group of users preferred it and it is fairer for the game logic. If alternate movements like the teleportation is used, it would be nearly impossible to kill the VR player in the scene. Since, as soon as the VR player sees a bomb, he can teleport to another location within fraction of seconds before the bomb detonates. The trackpad movement is in sync with the joystick movement of the mobile player and therefore the game seems fair.

2. Shooting: The trigger button on the controller shoots laser beams from the head target in the direction of the head target. This is to simulate the effect of being able to shoot through your head in the virtual world. Shooting where the controller points is traditional and does not work best the with Alien character as he is not holding any object. The laser beam when collided with an object creates particle effect. If there is no object it collides with, within 3 second, it destroys automatically. The Mobile player when hit with this beam, dies with 5-10 successful hits. To add fairness to the game, the laser beam will reduce in its length after first successful hit. This is to avoid mobile player to die easily by 5-10 consecutive successful hit. This approach make sure that the mobile player would get 1 successful hit in 7-8 hits which gives them enough time to run away from the VR
player. The beam detects the object it collides with using ray casting method in Unity. There is a method attached to each player which checks for incoming collisions. When the laser beam hits the collider of the player, the method checks for collision and reduces the health of the player accordingly.

3. Picking objects: The trackpad on the controller can also be pressed. If the VR player presses the trackpad within 1.5 units of the DNA samples, the particle effect as in figure 18 is triggered and the DNA samples is destroyed from the game scene.

![Figure 18: Particle effect on picking up objects](image)

This kind of particle effect depicts that the Alien is picking up the objects and sending it back to its home system.

4. UI: The menu button on the controller opens a static panel with stats about how many samples are left to be collected as shown in the figure 19. It disappears upon pressing the menu button again. This is to help the VR player keep track of how many object he has collected and how many are
Figure 19: UI panel for the VR player.

Game End – The game ends under following conditions:

- VR player is killed with the bomb hit.
- VR player collects all the Samples
- Mobile player dies, in which case, only this player is removed from the room.

Then the players are taken to another scene which ends the game.

Audio –

The sounds in this game were taken from online sources. There is a background music for each different scene of the game. Other than that, the bomb placing, explosion, picking of objects also trigger an audio clip of the respective activity.

There is a sound on and sound off button on the main screen. If the user chooses to select turn off the sound, he will not be able to hear sound further in the game. When the main environment of the game starts, there is an audio clip of a robot-like character explaining the story of the game. Sound effects in the game add focus in the game and the player can connect to the events happening in the scene because of these audio clues. Also, he becomes aware about what is happening in the rest of the environment which is not in his view. For example, if the VR player is shooting at 2nd mobile player, the first mobile player can listen to the shootings.
and try to locate the players to place a bomb while the VR player is distracted in shooting this other player.

**Testing:**

As seen in the various sections of the report, every decision for this project was based on user feedback. Using XP strategy, for each deliverable a set of students from different background were selected and asked to comment on the deliverable. Before the final testing, when the VR and mobile devices were connected, both were not simultaneously tested. Once the design and implementation were finalized, a few students were asked to test the entire application. When the mobile player connected to the game, the VR player experienced huge networking lag. There appeared a blue screen with loading sign and when the scene came back, the game broke and did not work as expected.

Some online forums complained about similar issue with VR related multiplayer games [8]. They suggested it is because of the WiFi speed and its inability to communicate with network at the speed of updates in the Vive which leads to the lag in the game. This greatly reduces the experience and is the major short coming of this project which was discovered towards the end of this project’s timeline.

The objects in the scene were reduced to bare minimum as seen in figure 20, but it was not able get rid of the network lag.
Some users suggested that this reduced graphics scene feels nice and warm in the Virtual Environment but the lag persists. Therefore, it was concluded that the lag is not because of the object networking but because of the Internet speed to update the game data every frame.
The Network lag can be seen in figure 21. It takes some time to update the players position as indicated by the red line. The player reached the green vertical line positions later than it is supposed to be. Therefore, the game starts breaking, giving an unpleasing experience.

The main use case for this project was tested with several players who reported the lag but agreed upon the main functionalities of the project. A successful use case is described below:

1. User joins the game as VR player and shoots at objects, picks up the objects, roam around the scene.
2. Mobile Player connects the game and goes near the VR player and puts a bomb near him and detonates it.
3. VR player dies and the game ends.

This use case makes sure that the functionalities described above works and therefore the deliverable for the first version of the game is ready.

After taking into consideration a few online suggested solutions, the mobile player and the VR player were connected to different WiFi network and the network lag was completely gone. This suggests that the network used for earlier testing of the game was congested and was not ideal to play the game. It is recommended therefore, that the game would be best played in a network which is not congested. However, this does not explain why people sitting in different locations in the World experience lag with other VR multiplayer games.

**UNITY GAME ENGINE**

This project is developed on Unity 3D platform. The way this game engine works is very intuitive and easy to develop games. It supports Physics to make the game be more realistic.
Each Game Object in the game can be assigned a tag and be referenced from the script using the tag. Each object has various components like Transform, Mesh Renderer etc, which helps build up the object. Some Physics components like Gravity can also be added to the objects. And if the objects are supposed to behave in a certain way, C# or JavaScript scripts can be added. Unity schedules an update for each script on all the objects in the current scene, every frame.

This project uses C# for its development because the scripts from Steam VR, which is a required component for any HTC Vive related development, are developed in C#. Unity also provides options for adding colliders to objects and therefore, when two colliders collide, it triggers a function and the behavior of those objects can be modified accordingly. In this project, this approach is used for detecting collision between the laser beam and the Mobile player.


## SCHEDULE

### TIMELINE

The timeline adopted by this project is described in this section.

*Table 1: Timeline for the project*

<table>
<thead>
<tr>
<th>Month</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>September</td>
<td>Game story &amp; Mechanics</td>
</tr>
<tr>
<td>October</td>
<td>Feedback and Improvements in Scope for the game by changing a two-person VR – VR to 1 VR person VR.</td>
</tr>
<tr>
<td>November</td>
<td>Testing Networking capabilities by connecting multiple users and spawning them at random locations</td>
</tr>
<tr>
<td>December</td>
<td>Movements for the players using Joystick like buttons – touch screen for mobile and trackpad for VR player</td>
</tr>
<tr>
<td>January</td>
<td>Mid Term presentation and Report showing networking capabilities and movements of the players</td>
</tr>
<tr>
<td>February</td>
<td>Improvements in the game scene and story based on user feedbacks.</td>
</tr>
<tr>
<td></td>
<td>Change the player Avatars</td>
</tr>
<tr>
<td>March</td>
<td>Player Game Logic implementation – shooting, bombs, health points, death, end scene</td>
</tr>
<tr>
<td>April</td>
<td>UI for both the players</td>
</tr>
<tr>
<td></td>
<td>Networking to show updates about the game user interface</td>
</tr>
<tr>
<td></td>
<td>Camera Angles for the different type of players</td>
</tr>
<tr>
<td></td>
<td>Game Story about alien and Elves</td>
</tr>
<tr>
<td></td>
<td>Testing by asking users from different background to play the game and then do the necessary improvement.</td>
</tr>
<tr>
<td></td>
<td>Final Report and Final Presentation</td>
</tr>
<tr>
<td></td>
<td>Poster Exhibition</td>
</tr>
</tbody>
</table>
Table 1 shows the schedule for this project. At the end of each week, there was a deliverable produced. For each deliverable, there was a working game application with minor improvements or changes from the previous week’s deliverable.

For example, in September, a Unity project was set up with a sample scene and the characters from the master’s students game were imported. This is in accordance with XP principles to produce deliverables rapidly.

**FUTURE WORKS AND IMPROVEMENTS**

**IMPROVEMENTS**

There was only one developer working on this game, therefore, there were many small details which were overlooked during development to deliver within the given timeline.

To make this game complete, the following improvements are necessary in this game:

1. Animation synchronization across the network as currently, the animation works only in the local device.

2. Mobile Players Avatar rotating automatically in the direction it is moving. Currently, the mobile player looks in the direction the view angle is set when he is moving from one location to the other in local as well as over the network. The avatar should be able to change the direction automatically in the direction he is moving.

3. Health Bar on each player just like the name tag for each other player to understand the health stats. Because some VR users complained that they do not know how the health of the mobile player is affected when they shoot as they do not have visual cues about the same.

4. Some visual feedbacks about getting hit and the danger radius around the bomb to get a better experience.
5. Tutorial for the players to get used to the VR environment and get themselves familiar with the controller actions.

6. Let the players select how they want to join the game as VR player or as mobile player instead of making it manual which is the current implementation - first player who joins the game will be the VR player.

**FUTURE ADDITIONS**

After the above-mentioned improvements are completed, this game should add the following features for a great experience:

1. It should be extended to be compatible with just 1 VR player if mobile phone users are not available. This can be done by adding an AI. The Artificial Intelligence will be based on path finding algorithms like A*, Depth First Search, Breadth First Search. These algorithms take the origin and destination into account and gives the shortest path from the origin to the destination. A* gives more flexibility to the developer in implementing this algorithm as it considers cost of finding the shortest path and cost of every alternative path taken. There will always be 4 Elves in the scene. Therefore, if only one mobile player joins the game, remaining 3 Elves will be controlled by the AI. If there is no mobile phone player, the AI will control all the Elves in the scene and try to defeat the VR player. However, the difficulty laid by the AI will depend on the number of Elves controlled by it. This will also guarantee fairness to the players and both the types will have equal chances of winning in the game.

2. It should allow users to customize their avatars and names so that they feel more connected to the game and get a realistic virtual feeling.

3. The game’s graphics can be worked on more. Currently, the Mobile Player has a sword in the game but he does not use that sword anywhere in the game. Also, the game needs to be optimized for the mobile devices.

4. Investigate network lag and make it compatible for congested networks.
CONCLUSION

This project delivers a multiplayer game application which can be played with different types of devices like mobile phones and HTC Vive. The aim of the project was to give user a multiplayer experience through different types of devices. User testing and feedback demonstrates that the game was successful in giving them a new and natural experience. There is a huge scope for improvement in this game to make it a complete game, but it delivers the basic version to get the idea across the players about the capabilities of Virtual Reality. This game allows the user to explore the game at their own pace and therefore gives an immersive experience. It uses the basic game objects like bombs, explosions, laser beams so that the user is not over whelmed by the number of different things he/she experiences in the game.

This game also pays significant attention to the functions of the game for both the different types of players to make sure that the game is fair. It restricts the number of actions a player can take like triggering a bomb but at the same time does not allow the opponent to shoot consecutively.

This game uses user centric approach like XP to deliver a basic version on time. User centric approaches always have higher chance of success compared to the functionalities centric approach.
GAME MANUAL

COMPATIBLE DEVICES

- Any desktop computer which can run HTC Vive and Internet connection
- Any android device preferably ones which have at least 2GB RAM.

GAME CONDITIONS

- There is a desktop file which should be played on a device which has Steam VR installed with HTC Vive connection.
- At least one VR and one mobile player should join the game. The first player should be the VR player who connects to the game.
- If you experience network lag, change the WiFi network for VR player and the mobile player as your WiFi network might be congested or use a different source for the Internet Connection.

HOW TO PLAY

- VR Player:
  - The VR Player when connected should check which one of the controllers allow him to move using trackpad.
  - The trigger button is for shooting, aim your headset at the mobile player and then press trigger to shoot.
  - To pick up objects, press the trackpad button.
  - To view the number of objects collected, press the menu button.

- Mobile Player:
  - Use the right joystick to change the view direction
  - Use the left joystick to move
  - Use the place button to place a bomb at your location.
  - Use the detonate button to detonate the bomb at the location it is placed.
REFERENCES