Multiplayer and Collaborative Virtual Reality

Abstract—In today’s world, there are many applications on mobile devices which deal with 2 dimensional formats to complete a collaborative task among different users. A collaborative task is a common task which can be shared with many users and can be modified by many users in real time and even at the same time. Every user is supposed to complete a section of the collaborative task. If one user makes any change to the collaborative task then this change is visible to other users. In this fashion, every user can contribute to complete a section of the task and eventually the task will be completed. The main idea behind this project is to create a multiplayer and collaborative virtual reality 3D gaming application which can be played by many players using different platforms like Linux, Windows or Mac PCs and Android or iPhones.

I. INTRODUCTION

Different applications like Google docs and overleaf are available online for performing a collaborative 2D text task for many users. There are other applications like skype for communicating and collaborating in real world. However, there are very few environments where a non 2D non text task can be performed by many users with high efficiency. Hence, it becomes essential to look for such options into the collaborative virtual space. Here are some of the collaborative virtual and augmented reality environment related background works.

The paper by Leema et al. [1] proposes a new architecture for creating the virtual reality environment. For this purpose, the authors suggest creation of a new kind of API for virtual environment. The paper gives in detail an assembly of different hardware components like databases and machines which make use of software components like JAVA3D APIs [1]. This new API seems to enable the collaborative virtual reality environment effectively.

One more paper by Churchill et al. [2] talks about the issues related to such kind of collaborative virtual reality environments. The authors state that the environments which are capable of creating the virtual reality must be distributed applications and they must provide an ability to communicate so that the information can be shared which will enable any one using this environment to create collaborative virtual reality applications [2].

Another paper by Benford et al. [3] talks about the kind of challenges for such environments. The author mentions that scale becomes an issue when this virtual world is extended to many users and some of the other challenges may include choosing from various types of architecture styles like peer to peer or server and client architecture [3].

The paper by Monahan et al. [4] talks about the application of virtual reality in the education sector and proposes an online learning environment so that the students can collaboratively learn different courses. For this purpose, the authors propose the use of virtual reality along with online access and hardware devices like camera for collaboration and this system can support both mobile and computer platforms [4].

One more paper by Kan et al. [5] state one more application of the virtual reality as a better method to design a product. The author proposes a VR environment using VNet and VRML which will be efficient in designing low cost products for smaller companies [5]. Such collaborative environments are also available in augmented reality space. Some of them are discussed below.

Augmented reality allows us to insert virtual objects into the reality by using devices such as mobiles, computers etc [6]. Augmented reality makes things more interactive and is more useful for human computer interaction as well [7]. It also helps in giving better perception of the real world [8]. The authors state that this new VR environment is better that the other environments as the time taken to design products using this environment is lesser than other environments [8].

Collaborative augmented reality involves virtual reality objects being shared by many devices [8]. In collaborative augmented reality applications, the devices are synchronized and they are sharing the virtual objects [8]. Hence, any modifications done by any device at any point in time gets reflected on the other devices. A shared task is completed in collaboration with many users.

For this project, such shared task is a multiplayer game in which different players can share a common scene. A multiplayer gaming application will be developed for sharing and modifying a 3D room scene. Many players on different devices will be able to join the room. The collaborative task mainly involves destroying common targets. Any changes done by any of the devices to a common target must be simultaneously reflected on all the devices.

The following collaborative tasks can be performed.
1) The Players can collaboratively explore the game space together.
2) Players can collaboratively destroy special objects like specific types of Rocks
3) Players can form teams and collaboratively go against each other.
There are many 2 dimensional applications which can assist us to perform a collaborative task in virtual reality space.

A. Experimentation

For the purpose of understanding the concepts of collaborative tasks, two open source projects were downloaded and executed. The first project is a collaborative drawing applications developed by Puffelen et al. [9]. It is built using a Firebase database. Firebase is a real time database. Using firebase database, two or more Android devices can share any kind of data. It allows to create any object which can be stored in the real time database. Other Android devices can also access that database. All of the devices can make changes to that object and the same changes will be reflected on all the devices. This open source uses event Listeners for this purpose [9]. E.g. Any object such as white board can be created. Such a white board can be shared among many users for collaborative painting purpose. Such white boards can be synced and any changes to the board will be reflected on all the Android devices. As we can see that the word Hi from the Android Device with Black Background from figure 1 is reflected on the Android Device with White Background from figure 2. So this application allows any user to draw anything on the 2D canvas and the same can be shared with the other users. Many users can collaborate and complete a shared task of drawing or writing a common task using this application.

The second open source project is developed by Fernandez et al. [10]. This project which uses unity packages for rendering is applied to 3 dimensional models [10]. Here, the targeted 3 dimensional model is a sea creature. This project is an improvement over the earlier projects as this project is applicable to 3 dimensional objects. Moreover, in this application we can vary the size of the brush strokes [10]. Suppose, if we have applied red color to a specific section of this 3D model and after that we tried applying blue color to this model. Then the application puts the resultant color after mixing of red and blue color in that area. Moreover, we can observe that the size of the brush stroke has increased.

II. Initial Setup

Before moving to initial setup a high level description of system design is necessary. There are three devices namely Android phone, a Computer and an iPhone in figure 3. These devices are connected to the photon cloud and each device is running a unity gaming application with an AppID of 1. As all the devices are using the same AppID, these devices are capable of networking with each other as they are sharing a specific location in the photon cloud. Moreover, the gaming application running on each device has networking scripts and player control script which will be used to control the networking configuration of that device and the player movement of that device.

A. Unity

Unity3d is essential to create a virtual reality gaming application. Unity gives an option to store any newly created project on either the disk or unity’s own cloud. Unity has many versions like personal, pro etc. However, for this project’s requirements personal free version is sufficient. Every unity project will have four essential components which can be stored in four different sub folders for modularity purpose. These are materials, prefabs or models, scenes and scripts.

1) Materials: The kind of material that has to be used for different kind of prefabs or player models.
2) Prefabs: Prefabs is nothing but our 3D player model. Unity will look for Resources folder in order to find all the models, otherwise it will throw an error. Hence, to avoid that all models must be under Resources.

3) Scene: A scene is our static gaming environment which may include sand, palm trees, mills, different kinds of Rocks and bushes etc.

4) Scripts: For player movement purpose, a separate player control script is needed. For networking and communication purpose, network scripts are needed. The player control script is responsible for movement of the player prefab, the players orientation and its camera view along with its projectile shooting capability.

B. Photon Unity Network Plugin

The photon unity plugin network is essential for establishing network connection among different devices. Photon free package supports up to 20 multiplayers at a time. This plugin can be downloaded from the asset store of unity. For the expansion purpose of this project in the future, installing all the packages of this plugin is essential. Such import figure 4 and its API update image figure 5 will appear in unity environment. For this kind of project, mainly its view and network plugins packages are sufficient.

Some times unity might throw an error suggesting that there are some old APIs that needs to be updated as shown in figure 5. For any given device if it has to establish a network connection, then it must be able to access the photon cloud. In order to access the photon cloud, every single application that is created using a photon plugin will have its own independent Application ID or AppID. Every single device which will be using the same application can communicate with other devices as there is one common ID. Using this ID these devices access the photon cloud and hence, are able to communicate and complete a collaborative task together. For this purpose, a photon unity networking wizard will pop up. This wizard asks for an AppId if there is a need for a player to join an already established photon cloud app. However, For a beginner application this ID can be extracted after providing the email address for registering as a new user. Initially this Setup can be skipped. However, at a later stage we will need to have our own AppId in order to run our application using photon cloud so that we can access photons networking capabilities which will help different players to share their actions with others. Server configuration file of photon is the most essential file for networking purpose. Its location is shown in figure 6. This settings files deals with many remote procedure calls and allows to set or disable those RPCs.

Server Settings inspector is an informative window Inspector and this file can be observed as shown in figure 7. This inspector gives the option to setup AppID. It also provides different logging options which are set to Errors only option. It means in the console window we will be able to see messages only if there are errors in our connection. But for debugging purpose and to see if we can give console output messages while testing a function, we need to choose Full option for all kind of loggings. As we can see in the Inspector window,
clear RPCs option will remove all of the default remote procedure calls of the photon unity networking plugin. The photon networking package has many remote procedure calls. If required we can add those as on demand basis to our project.

III. PLAYER MODELS

The player model or the prefab is a three dimensional capsule shaped object which has a shooting arm and bullets associated with it. Unity has inbuilt capsule shaped object. However, this capsule shaped object is modified and painted as per the requirements of the project. The entire inspector section of our capsule player is shown in figure 9. This section indicates the scale of the capsule, its position and different scripts associated with it. Complete player shape is also shown in figure 8. The different components of our capsule player are discussed below.

1) Capsule: The scale of capsule object is fixed at 1 X 1 X 1 as shown in inspector section figure 9. Different colors can be given to different prefabs. For this capsule, a new material was created and it was added as a new component to the capsule. Color inspector of new material can be seen in the figure 10. The initial rotation is fixed at zero for all the three axes. The original position of the capsule is at (10.764, 1, 10.19) from the origin (0, 0, 0). It means when the game starts then the player will appear at this position unless the player control script specifies a different random location.

Fig. 7. Server Configuration file of Photon

clear RPCs option will remove all of the default remote procedure calls of the photon unity networking plugin. The photon networking package has many remote procedure calls. If required we can add those as on demand basis to our project.

III. PLAYER MODELS

The player model or the prefab is a three dimensional capsule shaped object which has a shooting arm and bullets associated with it. Unity has inbuilt capsule shaped object. However, this capsule shaped object is modified and painted as per the requirements of the project. The entire inspector section of our capsule player is shown in figure 9. This section indicates the scale of the capsule, its position and different scripts associated with it. Complete player shape is also shown in figure 8. The different components of our capsule player are discussed below.

1) Capsule: The scale of capsule object is fixed at 1 X 1 X 1 as shown in inspector section figure 9. Different colors can be given to different prefabs. For this capsule, a new material was created and it was added as a new component to the capsule. Color inspector of new material can be seen in the figure 10. The initial rotation is fixed at zero for all the three axes. The original position of the capsule is at (10.764, 1, 10.19) from the origin (0, 0, 0). It means when the game starts then the player will appear at this position unless the player control script specifies a different random location.

Fig. 8. Player Prefab

2) Bullet: A bullet prefab is a spherical 3D object as shown in figure 11. The spherical shape was painted using a new material.

3) Gun: Gun is a shooting arm of the player model which is located on the right hand side of the player. Gun is similar to a rectangular prism. Unity provides cylindrical 3D object which can be used as a shooting arm. However, some custom modifications to this cylindrical shape was needed. Hence, scale of the gun and its position from center of player is chosen in such a way that it acts as the right hand of the player model. A new material with blue color is created for the Gun.

4) Main Camera: Main camera is the camera which is associated with our player model. If the player moves, along with the player the camera also moves and it
maintains a specific distance from the player model. As the player moves, different locations can be observed in the game environment.

5) Spawn point for Bullet: The bullets must be spawned from the front end of the shooting arm. The spawn point of the bullet is a location just in front of the shooting arm. This is the location from which the new bullets will be spawning one after the another. When the player presses shooting keys, this position is located such that the bullets will start spawning from the front end of the shooting arm.

6) Health: For health indication a cylindrical horizontal shape is added to the player model. Its foreground is green and background is red in color. So if the player gets hit with bullets, then green color will reduce and red color will be visible.

IV. Scripts

There are two types of scripts which are used for this gaming application.

1) Networking Scripts
2) Player control script

A. Networking Scripts

There are total two networking scripts which are stored within scripts section. They are mentioned below.

1) MyNetworkManager
2) MyNetworkPlayer

MyNetworkManager: The MyNetworkManager file is used to connect the players device(Android or Computer) to the network. MyNetworkManager script manages the player Capsule, the game scene/room and the version number of the game. This script also takes care of spawning of players at different locations in the game scene. For networking purpose, this script uses photon.

This script has different functions which are used for different purposes. Some of the main functionalities of the script are discussed below.

1) The main purpose of this script is to connect the given player initially to the photon Network. For connection purpose, the photon utility function needs the version number in string format.
2) The script also allows additional activities like debugging after a player has joined the gaming room.
3) To join a room, unity utility function can create a new room and can also restrict how many players can join that room. It also gives an option whether the room should be visible to others or not. As we are developing a collaborative multiplayer application which will be run on different platforms, this option is set to true. Using this script, the player can either join an existing room or create a totally different room.
4) In case, if photon is simply not able to connect the player to the created room then in that case this script simply creates a new room only. It will not try to join a room in this case. The script can also print the cause and the error that occurred while connecting to cloud.
**MyNetworkPlayer:** The script MyNetworkPlayer has the functionality to control the camera. This script checks whether the view belongs to the player in the scene and if it is true then and then only it activates its camera. If we do not use this script then in any given scene if there are more than 1 player, and we try to use the player controls then every single player in the scene will move simultaneously. Hence, it becomes necessary to identify which player in the scene belongs to our device and the input coming from the keyboards of that device is used to perform movements of only that player in the given scene.

The major functionality of this script is to determine whether the current camera view belongs to our device. If it is true then the script will activate the camera of that player. It means the camera can move along with the player and we can observe different places. Now, the controls of that player in the scene must be activated. For that purpose, this script enables the control script of that player. However, if the view does not belong to us then the script will deactivate the camera and for every single player in the scene, it will deactivate every players Control script. If we dont do this, then all the players in the scene will move simultaneously as they all will process the same input from any device.

**MyPlayerControl:** MyPlayerControl is mainly used to control the behavior of the player object in the scene. This script is useful for controlling the movements of a player in the given scene. As the behavior of a player must be seen by other players as well, we are using network Behavior for this script. The script uses two objects to access the bullet and its spawning point. The Bullet object references to the bullet model discussed earlier and the Spawning point of it determines from which point the bullets will be spawning. This script contains the following functionalities and its usage is discussed below.

Whenever an input is given by keyboard of any of the devices, this script gets invoked. As discussed earlier, this script also determines whether the given view belongs to it or not. If this is true then it performs the following set of actions. So whenever any user gives input from its keyboard for the player movement purpose, it is usually in the terms of horizontal or vertical keys. To capture this input, unity provides different packages which can be used to access any kind of input from the keyboards of any type of platform like Android, Linux, MAC PCs etc. These horizontal and vertical movements are stored as x and z positions. X position is used for rotation movement of the shooting arm and z position is mainly used for the translation with respect to the current position. Using the combination of both rotation and translation, the player can move to any position in the given scene.

Hence, translation and rotation of the player object is possible using unity’s library functions. Unity is compatible with many platforms. hence, one of its package enables the program to take the inputs from all the platforms including Linux, Windows or Mac PCs and Android or iPhones. After this, the program checks whether the space key is pressed by the user on the device, if that is the case then the shooting algorithm is invoked and bullets are fired from that devices shooting arm. As unity’s mobile joystick is included for mobile platform, this functionality is mobile friendly as well.

Now the function shoot is of void type and it is used as a Remote procedure call. The shoot function initializes game object. This object gets instantiated using previously declared global variables bullet and bullet Spawning point. It uses position and rotation elements of bullet Spawning point to determine from which location the bullets should be spawning. As discussed earlier, it is the front end of the shooting arm. We assign a rigid body to the bullet and then its velocity is set by the script. We can give a multiplication factor of 6.0 so that bullets will be moving forward with a speed six time greater than the original assigned speed. As we have used rigid body, these bullets will bounce off of other players body. The bullets should not be in the scene for more than few seconds. They need to be destroyed after some seconds. So for this purpose, the script will destroy the bullets just after 3 seconds of its instantiation. The scene of this multiplayer game looks like in figure 13. If the bullets are hit to player then its health component reduces which is reflected in the health cylindrical shape above player’s head. It becomes partially red in color.

**V. Results and Evaluation**

The photon unity network along with unity has an advantage that it allows almost 20 players at the same time in a given scene. The movements of different players at any given points look smooth. This gaming application can be evaluated only using user experience. So, the survey for this gaming application includes various questions about latency and user interaction. Unity seems to be doing a good job when it comes to this kind of 3D game environments. However, there are many other platforms which are better at doing collaborative multiplayer gaming applications. However, these platforms may not be necessarily good when it comes to the virtual 3D applications. Moreover, unity has its own libraries which are useful when it comes to exporting the gaming application to any of the platforms including Linux, Windows or Mac PCs and Android or iPhones platforms. The game is pretty interactive. However, there could be a bit of a latency issue
for a player if he wants to join an existing game. Moreover, we can modify and restrict the number of players that can join a gaming room to a number between 0 to 20. The best part of this application is a player can use his Android device or iPhone device to join another players who could be using Mac, Linux or Windows platforms.

The users were presented with the interface shown in figure 14. The ratings of each question range from 1 to 4.

1) Rating 1 : Very Bad
2) Rating 2 : Bad
3) Rating 3 : Good
4) Rating 4: Very Good

The following questions were asked to users.

1) **Overall User Experience**: Overall User Experience is measured on a scale of 1 to 4. The users were asked to give their ratings for User Experience and their responses are recorded and plotted in a bar chart as shown in figure 15. Highest value for user experience is 4 and lowest is 1.

2) **Performance/Latency**: Performance is measured on a scale of 1 to 4. The users were asked to give their ratings for Latency and their responses are recorded and plotted in a bar chart as shown in figure 16. If there are no issues with latency then a max value of 4 can be given and with lowest being 1.

The experiment was carried out with total 5 groups. Every group had 2 players. There answers to these questions were converted into a bar chart which are shown in figure 15 and figure 16. The average overall user

VI. CONCLUSION AND FUTURE WORK

There are many Android applications which deal with collaborative tasks. Unity along with photon unity plugin is helpful to create a great multiplayer collaborative application which can be use by many users. Unity has a great advantage as it provides a special package which can be used to create a single project on Unity platform and the same project can be exported to different platforms. For computers, same project can work on Linux, Mac or Windows PCs and for phones it can work on iPhones and Android devices. Photon is excellent when it comes to networking. Photon makes it much easier to create a working networking application. All in all the combination of photon and unity is highly useful when it comes to producing a multiplayer and collaborative application. The future scope of the project is given below.

1) In future, the project must be able to support more than 150 players in the same gaming environment.
2) In future, the game must be hosted on a big server and be capable of handling the international traffic in real time as well.

REFERENCES