**Multiplayer FPS Game Project**

By Vivek Sharma

TYBSc. Computer Science

Roll no.:482

**Introduction:**

The main goal of this project is to make a FPS(First Person Shooter) game and implement Multiplayer functionality using the Unity Engine. In order to achieve the multiplayer functionality we will have to use an asset from the unity asset store named Photon PUN. It is a very convenient way of implementing multiplayer networking. Photon makes it easier to implement networking and host client communication enabling indie- gamedevs to make multiplayer games effortlessly and its also free to use for smaller games.

**Photon PUN:**

It can accommodate up to 20 players in one server. We make an account on Photonengine.com and then we can create our own multiplayer projects. Every Project has a unique App-id that can be used to make the server in unity by entering our own unique app-id after importing Photon pun asset.

Now that the Photon PUN package has been imported, we set up the server to our closest location so that we get the least ping difference. Photon has a list of fixed region servers that can be assigned on its website. We go to Resources folder in the photon package and then in Fixed region assign ‘in’ as its the code for the Indian region. The photon server for our region is located in Chennai.

**Working of Game in Unity Engine:**

For the project to work in unity we first design the menus required to navigate through the project. The code responsible for the working of the UI and menus will be given below. Once the menu management is done we can start designing the test playground to implement and test the multiplayer connection and make sure both host and client are connected to the same server. We do this by using the Room (lobby) system in Photon.

Once the connection part is sorted out we can program the mechanics of the game to be given to the player which will be the same for every single player in the game. The player mechanics such as running, jumping, equipping items, etc are programmed / written in the PlayerController.cs script. Code for same will also be given below.

**Scripting:**

There are many more scripts that have been made for different small functionalities that are in the game some of them are responsible for the mechanics some for the network and some for the menu management, But even though these all scripts are different they work in conjunction with each other for the smooth functioning of the game.

Every single script in Unity is written in C#, since Unity only supports C# scripting. They have a “.cs” extension.

**Finishing the Game:**

Once we Successfully Implement the Multiplayer Functiionality we will be able to play the game using the build in the games files. Making multiple players and making them join the same game server on the same map.

Once all of this functionality works smoothly we can say we have successfully implemented and made a game that has Multiplayer functionality using Unity Engine & Photon PUN.

**Scripts:**

All the scripts used in the game are written/given below,

**MenuManager.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MenuManager : MonoBehaviour

{

public static MenuManager Instance;

[SerializeField] Menu[] menus;

private void Awake()

{

Instance = this;

}

public void OpenMenu(string menuName)

{

for (int i = 0; i < menus.Length; i++)

{

if (menus[i].menuName == menuName)

{

menus[i].Open();

}

else if (menus[i].open)

{

CloseMenu(menus[i]);

}

}

}

public void OpenMenu(Menu menu)

{

for (int i = 0; i < menus.Length; i++)

{

if (menus[i].open)

{

CloseMenu(menus[i]);

}

}

menu.Open();

}

public void CloseMenu(Menu menu)

{

menu.Close();

}

}

**Menu.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Menu : MonoBehaviour

{

public string menuName;

public bool open;

public void Open()

{

open = true;

gameObject.SetActive(true);

}

public void Close()

{

open = false;

gameObject.SetActive(false);

}

}

**PlayerController.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using Photon.Pun;

using Photon.Realtime;

using Hashtable = ExitGames.Client.Photon.Hashtable;

public class PlayerController : MonoBehaviourPunCallbacks

{

[SerializeField] GameObject cameraHolder;

[SerializeField] float mouseSensitivity, sprintSpeed, walkSpeed, jumpForce, smoothTime;

[SerializeField] item[] items;

int itemindex;

int previousItemIndex=-1;

float verticalLookRotation;

bool grounded;

Vector3 smoothMoveVelocity;

Vector3 moveAmount;

Rigidbody rb;

PhotonView PV;

void Awake()

{

rb = GetComponent<Rigidbody>();

PV = GetComponent<PhotonView>();

//playerManager = PhotonView.Find((int)PV.InstantiationData[0]).GetComponent<PlayerManager>();

}

void Start()

{

if (PV.IsMine)

{

EquipItem(0);

}

else

{

Destroy(GetComponentInChildren<Camera>().gameObject);

Destroy(rb);

}

}

void Update()

{

if (!PV.IsMine)

return;

Look();

Move();

Jump();

for (int i = 0; i < items.Length; i++)

{

if (Input.GetKeyDown((i + 1).ToString()))

{

EquipItem(i);

break;

}

}

}

void Look()

{

transform.Rotate(Vector3.up \* Input.GetAxisRaw("Mouse X") \* mouseSensitivity);

verticalLookRotation += Input.GetAxisRaw("Mouse Y") \* mouseSensitivity;

verticalLookRotation = Mathf.Clamp(verticalLookRotation, -90f, 90f);

cameraHolder.transform.localEulerAngles = Vector3.left \* verticalLookRotation;

}

void Move()

{

Vector3 moveDir = new Vector3(Input.GetAxisRaw("Horizontal"), 0, Input.GetAxisRaw("Vertical")).normalized;

moveAmount = Vector3.SmoothDamp(moveAmount, moveDir \* (Input.GetKey(KeyCode.LeftShift) ? sprintSpeed : walkSpeed), ref smoothMoveVelocity, smoothTime);

}

void Jump()

{

if ((Input.GetAxisRaw("Mouse ScrollWheel") < 0f|| Input.GetKeyDown(KeyCode.Space)) && grounded)

{

rb.AddForce(transform.up \* jumpForce);

}

}

void EquipItem(int \_index)

{

if (\_index == previousItemIndex)

return;

itemindex = \_index;

items[itemindex].itemgameobject.SetActive(true);

if (previousItemIndex != -1)

{

items[previousItemIndex].itemgameobject.SetActive(false);

}

previousItemIndex = itemindex;

if (PV.IsMine)

{

Hashtable hash = new Hashtable();

hash.Add("itemIndex", itemindex);

PhotonNetwork.LocalPlayer.SetCustomProperties(hash);

}

}

public override void OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedProps)

{

if (changedProps.ContainsKey("itemIndex") && !PV.IsMine && targetPlayer == PV.Owner)

{

EquipItem((int)changedProps["itemIndex"]);

}

}

public void SetGroundedState(bool \_grounded)

{

grounded = \_grounded;

}

void FixedUpdate()

{

if (!PV.IsMine)

return;

rb.MovePosition(rb.position + transform.TransformDirection(moveAmount) \* Time.fixedDeltaTime);

}

}

**PlayerGroundCheck:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerGroundCheck : MonoBehaviour

{

PlayerController playerController;

void Awake()

{

playerController = GetComponentInParent<PlayerController>();

}

void OnTriggerEnter(Collider other)

{

if (other.gameObject == playerController.gameObject)

return;

playerController.SetGroundedState(true);

}

void OnTriggerExit(Collider other)

{

if (other.gameObject == playerController.gameObject)

return;

playerController.SetGroundedState(false);

}

void OnTriggerStay(Collider other)

{

if (other.gameObject == playerController.gameObject)

return;

playerController.SetGroundedState(true);

}

private void OnCollisionEnter(Collision collision)

{

if (collision.gameObject == playerController.gameObject)

return;

playerController.SetGroundedState(true);

}

private void OnCollisionExit(Collision collision)

{

if (collision.gameObject == playerController.gameObject)

return;

playerController.SetGroundedState(false);

}

private void OnCollisionStay(Collision collision)

{

if (collision.gameObject == playerController.gameObject)

return;

playerController.SetGroundedState(true);

}

}

**Playerlistitem.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using Photon.Realtime;

using Photon.Pun;

using TMPro;

public class PlayerlistItem : MonoBehaviourPunCallbacks

{

[SerializeField] TMP\_Text text;

Player player;

public void SetUp(Player \_player)

{

player = \_player;

text.text = \_player.NickName;

}

public override void OnPlayerLeftRoom(Player otherPlayer)

{

if (player == otherPlayer)

{

Destroy(gameObject);

}

}

public override void OnLeftRoom()

{

Destroy(gameObject);

}

}

**PlayerManager.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System.IO;

using Photon.Pun;

public class PlayerManager : MonoBehaviour

{

PhotonView PV;

private void Awake()

{

PV = GetComponent<PhotonView>();

}

// Start is called before the first frame update

void Start()

{

if (PV.IsMine)

{

CreateController();

}

}

public void CreateController()

{

Debug.Log("instantiated player controller");

PhotonNetwork.Instantiate(Path.Combine("PhotonPrefabs", "PlayerController"), Vector3.zero, Quaternion.identity);

}

// Update is called once per frame

void Update()

{

}

}

**Roomlistitem.cs:**

using Photon.Realtime;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

public class RoomlistItem : MonoBehaviour

{

[SerializeField] TMP\_Text text;

public RoomInfo info;

public void SetUp(RoomInfo \_info)

{

info = \_info;

text.text = \_info.Name;

}

public void OnClick() {

Launcher.Instance.JoinRoom(info);

}

}

**RoomManager.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using Photon.Pun;

using System.IO;

public class RoomManager : MonoBehaviourPunCallbacks

{

public static RoomManager Instance;

void Awake()

{

if (Instance)

{

Destroy(gameObject);

return;

}

DontDestroyOnLoad(gameObject);

Instance = this;

}

public override void OnEnable()

{

base.OnEnable();

SceneManager.sceneLoaded += OnSceneLoaded;

}

public override void OnDisable()

{

base.OnDisable();

SceneManager.sceneLoaded -= OnSceneLoaded;

}

void OnSceneLoaded(Scene scene, LoadSceneMode loadSceneMode)

{

if (scene.buildIndex == 1) // We're in the game scene

{

PhotonNetwork.Instantiate(Path.Combine("PhotonPrefabs", "PlayerManager"), Vector3.zero, Quaternion.identity);

}

}

}

**Iteminfo.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Iteminfo : ScriptableObject

{

public string itemname;

}

**Item.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class item : MonoBehaviour

{

public Iteminfo iteminfo;

public GameObject itemgameobject;

}

**Launcher.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using Photon.Pun;

using TMPro;

using Photon.Realtime;

using System.Linq;

public class Launcher : MonoBehaviourPunCallbacks

{

public static Launcher Instance;

[SerializeField] TMP\_InputField roomnameInputField;

[SerializeField] TMP\_Text errorText;

[SerializeField] TMP\_Text roomnameText;

[SerializeField] Transform roomlistContent;

[SerializeField] GameObject roomlistitemprefab;

[SerializeField] Transform PlayerlistContent;

[SerializeField] GameObject Playerlistitemprefab;

[SerializeField] GameObject startgamebutton;

private void Awake()

{

Instance = this;

}

void Start()

{

Debug.Log("connecting to master");

PhotonNetwork.ConnectUsingSettings();

}

public override void OnConnectedToMaster()

{

Debug.Log("connected to master");

PhotonNetwork.JoinLobby();

PhotonNetwork.AutomaticallySyncScene = true;

}

public override void OnJoinedLobby()

{

MenuManager.Instance.OpenMenu("title");

Debug.Log("joined lobby ");

PhotonNetwork.NickName = "Player" + Random.Range(0, 1000).ToString("0000");

}

public void CreateRoom()

{

if (string.IsNullOrEmpty(roomnameInputField.text))

{

return;

}

PhotonNetwork.CreateRoom(roomnameInputField.text);

MenuManager.Instance.OpenMenu("loading");

}

public override void OnJoinedRoom()

{

MenuManager.Instance.OpenMenu("room");

roomnameText.text = PhotonNetwork.CurrentRoom.Name;

Player[] players = PhotonNetwork.PlayerList;

foreach (Transform child in PlayerlistContent)

{

Destroy(child.gameObject);

}

for (int i = 0; i < players.Count(); i++)

{

Instantiate(Playerlistitemprefab, PlayerlistContent).GetComponent<PlayerlistItem>().SetUp(players[i]);

}

startgamebutton.SetActive(PhotonNetwork.IsMasterClient);

}

public override void OnMasterClientSwitched(Player newMasterClient)

{

startgamebutton.SetActive(PhotonNetwork.IsMasterClient);

}

public override void OnCreateRoomFailed(short returnCode, string message)

{

errorText.text = "room creation failed : " + message;

MenuManager.Instance.OpenMenu("error");

}

public void StartGame()

{

PhotonNetwork.LoadLevel(1);

}

public void LeaveRoom()

{

PhotonNetwork.LeaveRoom();

MenuManager.Instance.OpenMenu("loading");

}

public void JoinRoom(RoomInfo info)

{

PhotonNetwork.JoinRoom(info.Name);

MenuManager.Instance.OpenMenu("loading");

}

public override void OnLeftRoom()

{

MenuManager.Instance.OpenMenu("title");

}

public override void OnRoomListUpdate(List<RoomInfo> roomList)

{

foreach (Transform trans in roomlistContent) {

Destroy(trans.gameObject);

}

for (int i = 0; i < roomList.Count; i++)

{

if(roomList[i].RemovedFromList)

continue;

Instantiate(roomlistitemprefab, roomlistContent).GetComponent<RoomlistItem>().SetUp(roomList[i]);

}

}

public override void OnPlayerEnteredRoom(Player newPlayer)

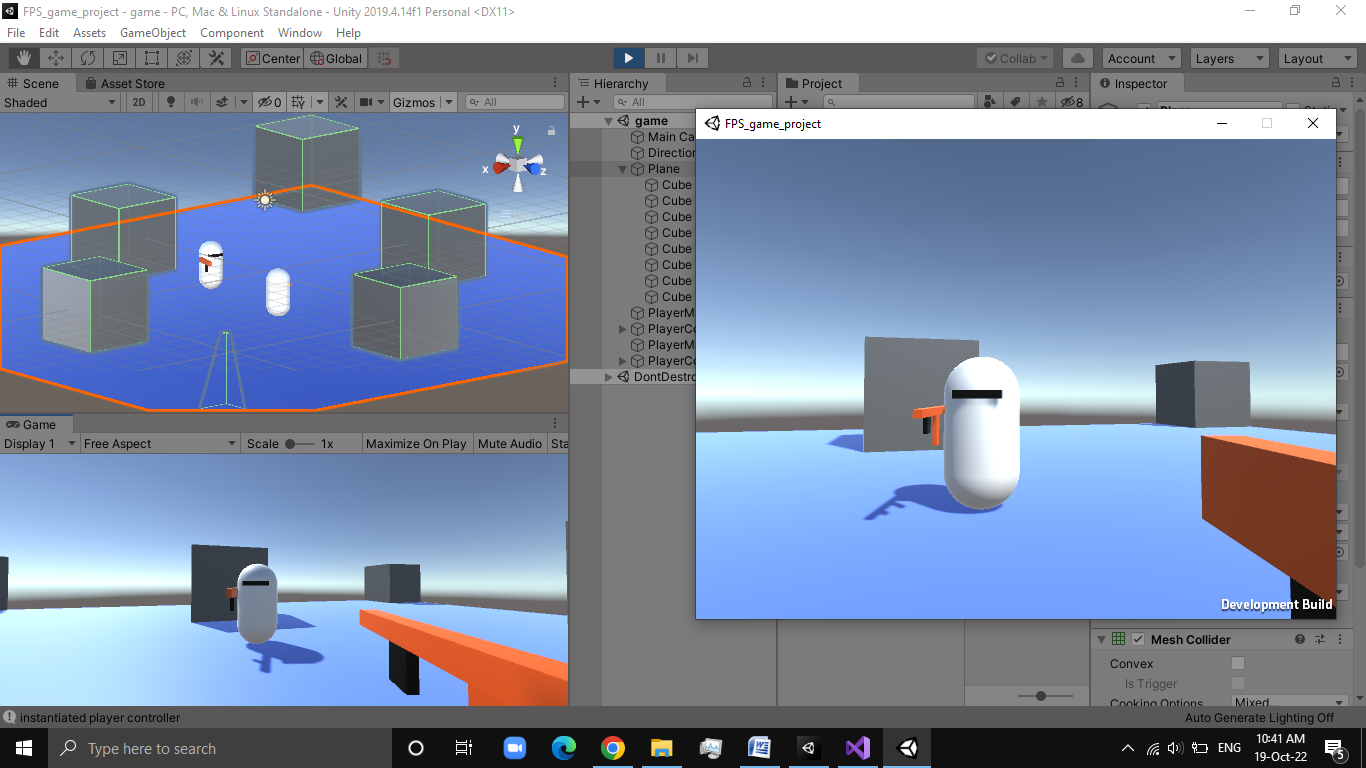
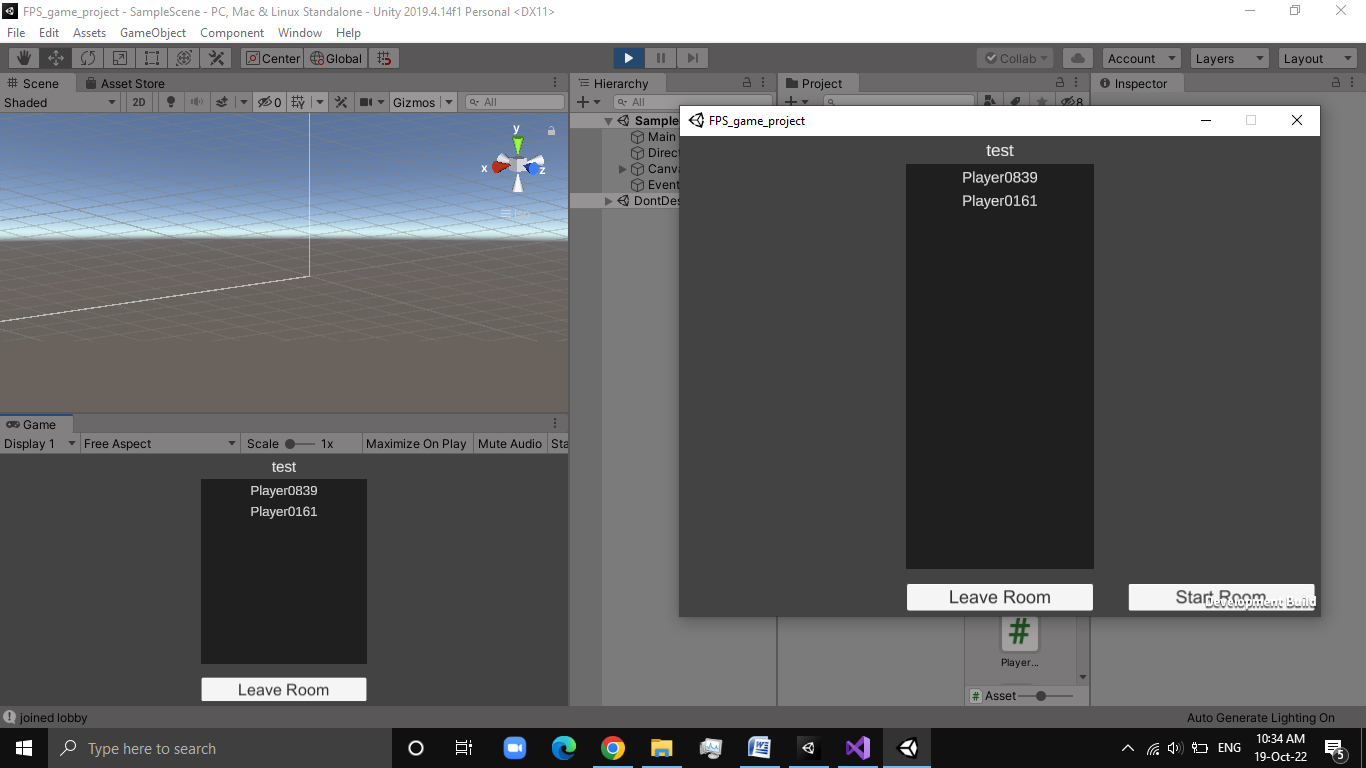
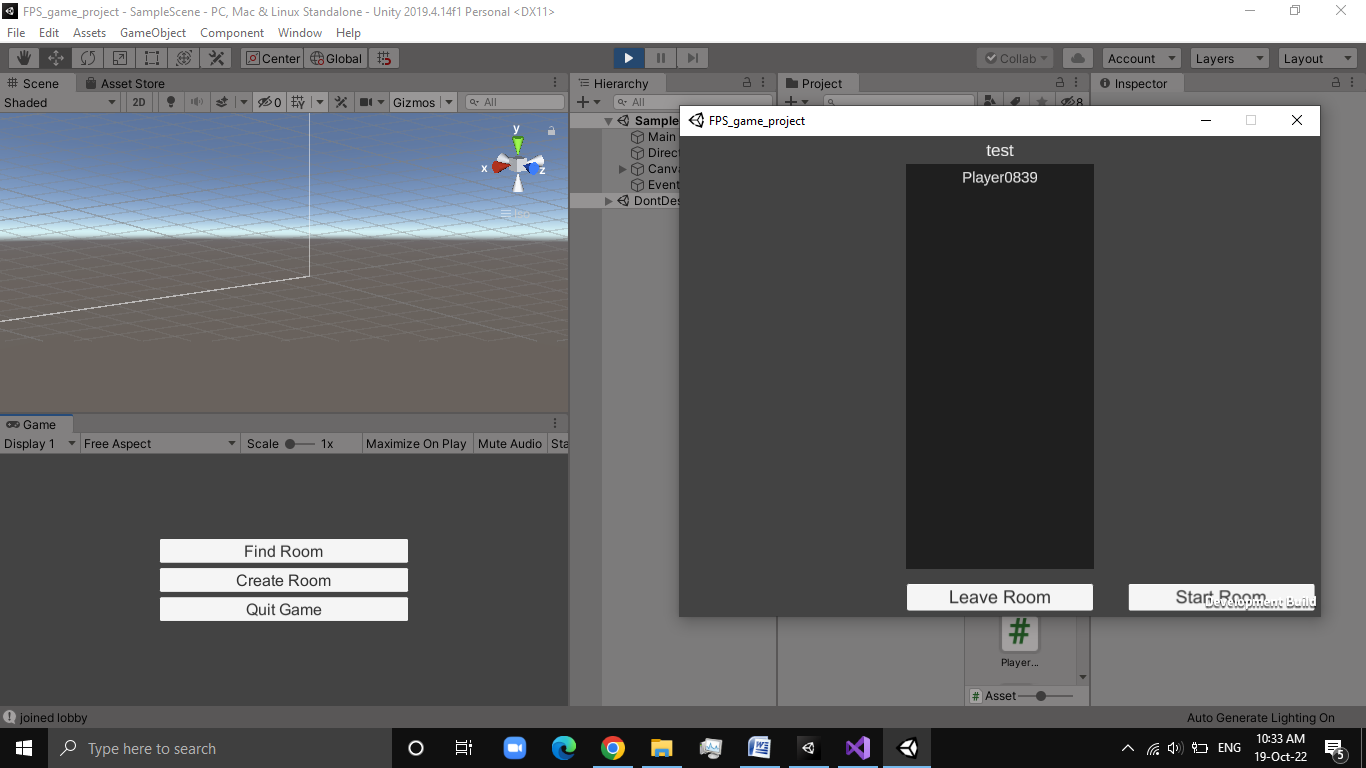
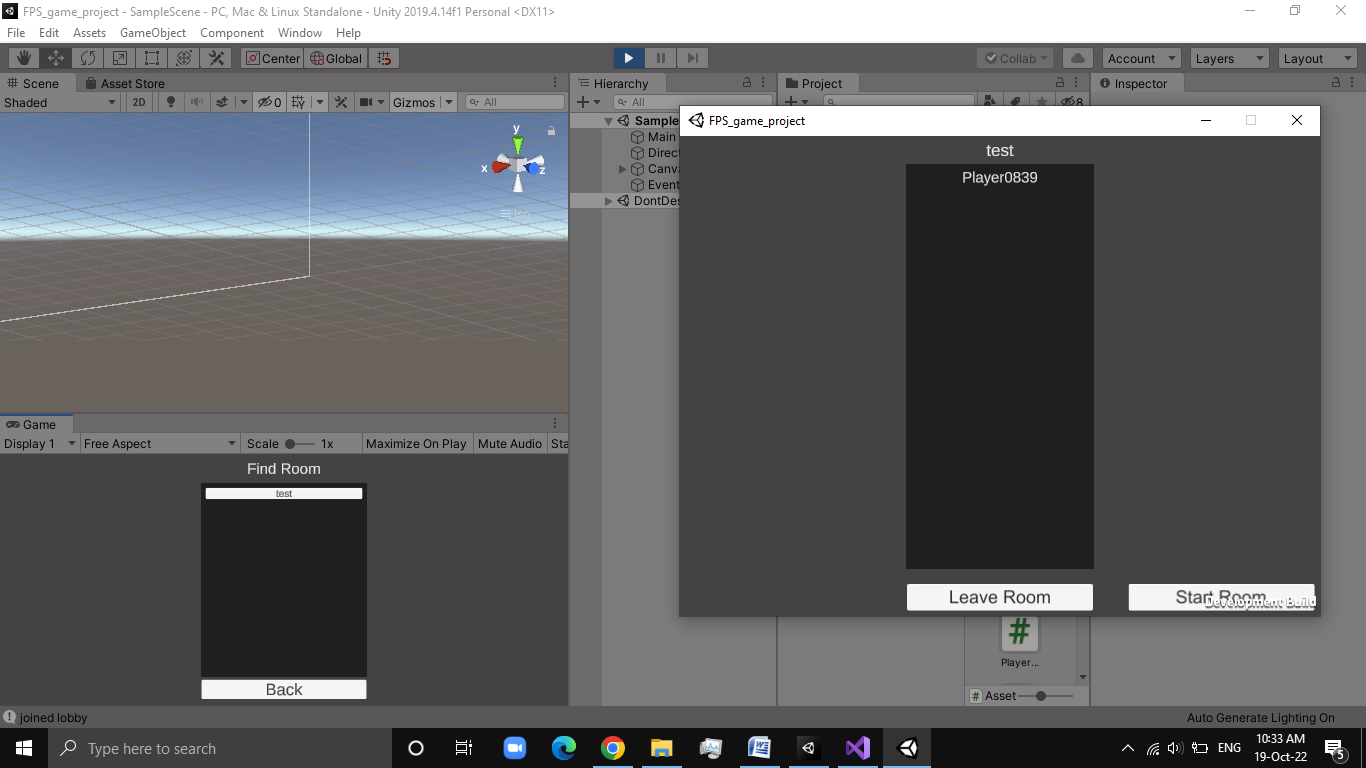
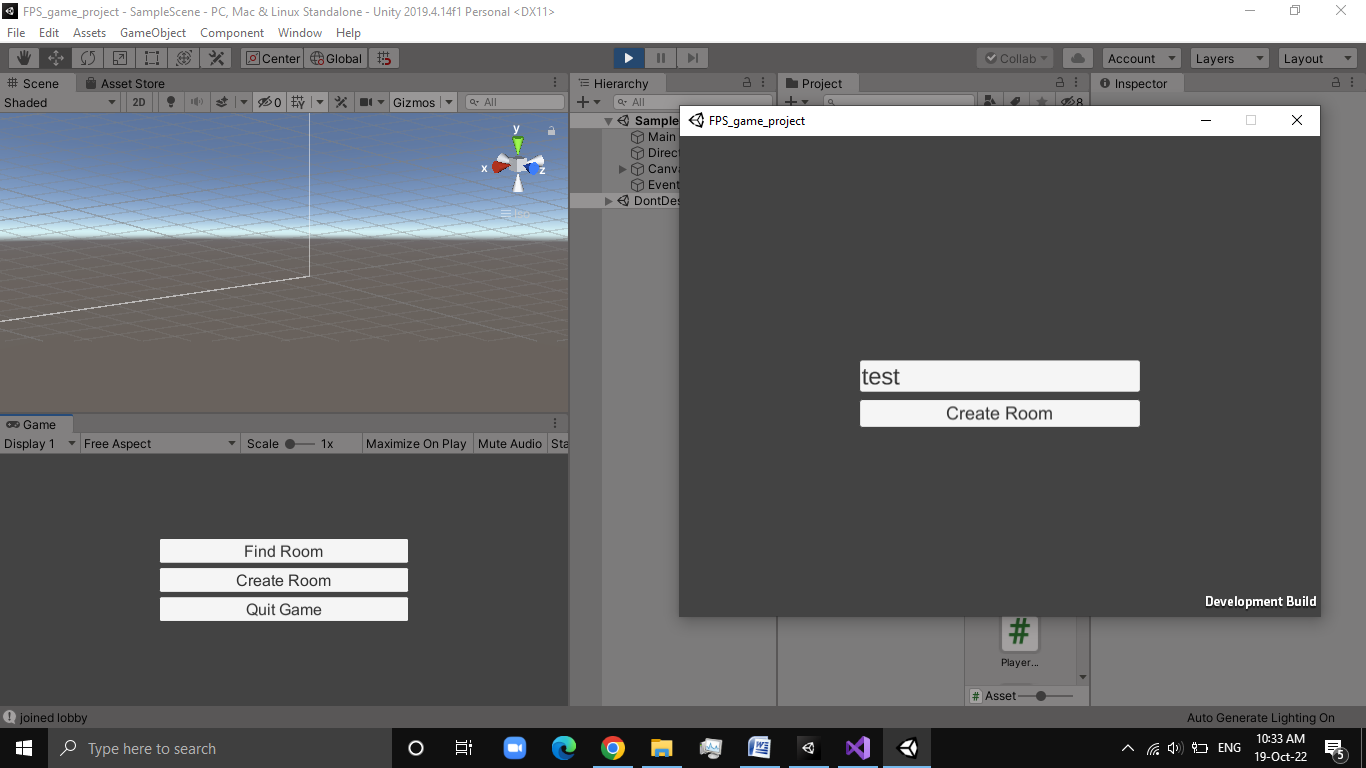
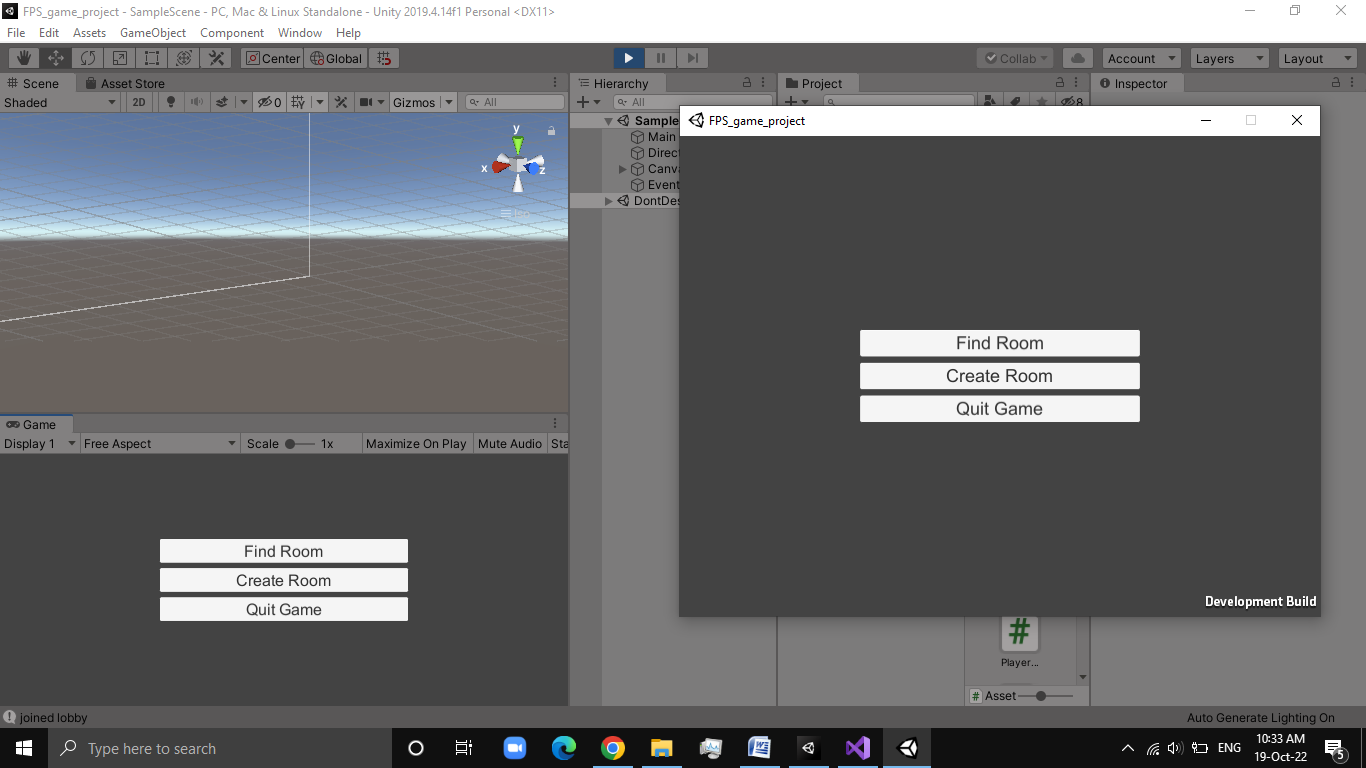
{

Instantiate(Playerlistitemprefab, PlayerlistContent).GetComponent<PlayerlistItem>().SetUp(newPlayer);

}

}

**Execution Demo Screenshots:**



**Conclusion:**

Here we have successfully created a Base for a FPS Multiplayer game with responsiveness from both the host and the client.

It is the most convenient way of implementing Multiplayer functionality in a game using Unity.