Homework 4

Yasson Haddish

Server

And

Francis Quang

(Client)

San Francisco State University

College of Science and Engineering

CSC 631 Multiplayer Game Development

Spring 2023

Contents

Project Overview. 3

Setting up 3

Connecting to Master Server 3

Creating a Lobby. 4

Creating Rooms 5

Room Browsing 5

Joining Room 5

Listing players and updating the list. 6

Players Nickname 6

Switching Masters 7

Project Overview.

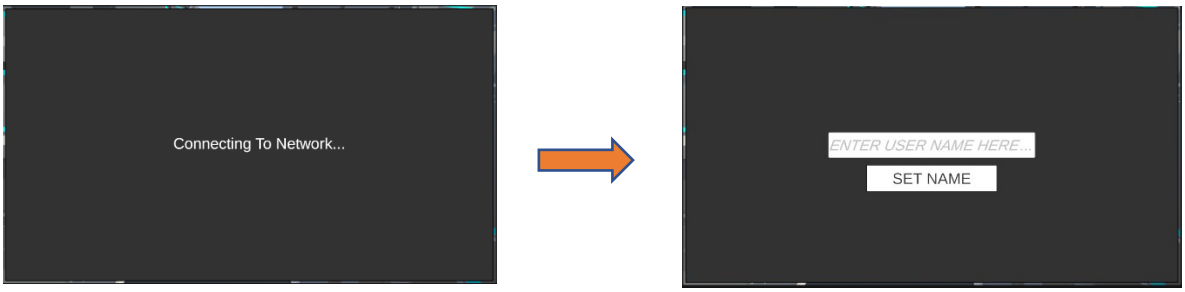
Our team is building an online battle game that allows players intersect and combat on the same room. The first thing to do is to make the basic control elements of the game. As a starting point for networking. Our players can move, rotate, jump, run, buttle and collide with our environment.

Setting up

For this game project we're testing our game using Photon Unity Network which is worldwide used and kind of straight forward to use and can allow users up to 20 players in the same room for free. Photon will generate unique App ID for us so we can use it to connect our Unity app with the server.

Connecting to Master Server

After successfully connecting our game with unity, the first thing to do is create a system that allows players to connect into our game. We've created a menu system that allows players to connect to the different rooms on the server so our players can play the game together. As soon as the game starts the player sees a loading screen that indicates they are connecting to the network. After successfully connecting to the server the loading screen will close and the other menu section will open.



We've created a new Launcher.cs script so that our game can be connected to the Photon server. When we work with photon network element, there is a function in addition to the normal default unity one other than start and update. For example, on connect to master which tells us when we connect to a server. The MonoBehaviourPunCallbacks will allow us to access those different default functions working with default on network.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using TMPro;
using Photon.Realtime;

public class Launcher : MonoBehaviourPunCallbacks
{
    public static Launcher instance;
    private void Awake()
    {
        instance = this;
    }
```

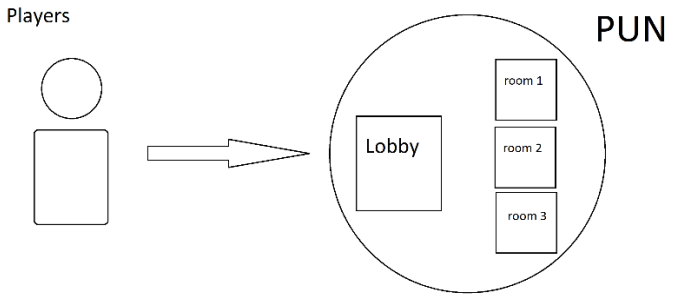
```
public void StartGame()
{
    PhotonNetwork.LoadLevel(levelToPlay);
}


public override void OnMasterClientSwitched(Player newMasterClient)
{
    if(PhotonNetwork.IsMasterClient)
    {
        startButton.SetActive(true);
    } else
    {
        startButton.SetActive(false);
    }
}
```

This allowed us to connect to the master server of the network.

Creating a Lobby.

Once we successfully connected to the master server, we need to connect it to our lobby. The lobby will allow us to find any rooms on the server, and in each room is where the match is taking place. So, when we connect to the server, we should choose on what lobby that finds us the rooms we want to join.




```
public override void OnConnectedToMaster()
{

    PhotonNetwork.JoinLobby();

    PhotonNetwork.AutomaticallySyncScene = true;

    loadingText.text = "Joining Lobby...";
}
```

```
public override void OnJoinedLobby()
{
    CloseMenus();
    menuButtons.SetActive(true);

    PhotonNetwork.NickName = Random.Range(0, 1000).ToString();

    if(!hasSetNick)
    {
        CloseMenus();
        nameInputScreen.SetActive(true);

        if(PlayerPrefs.HasKey("playerName"))
        {
            nameInput.text = PlayerPrefs.GetString("playerName");
        }
    }
    else
    {
        PhotonNetwork.NickName = PlayerPrefs.GetString("playerName");
    }
}
```

Creating Rooms

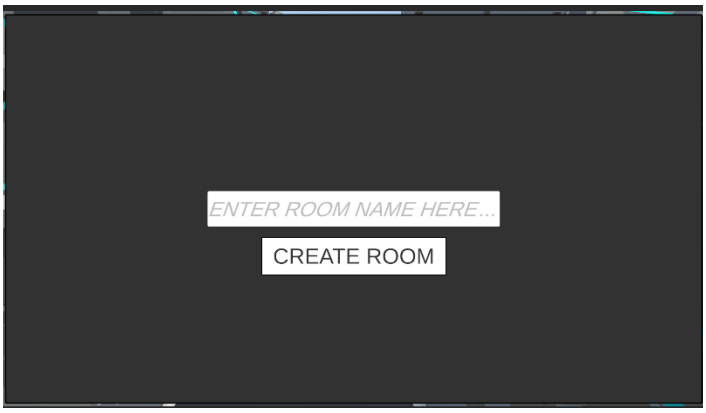
The first thing to consider on creating a room is our client must put a name and make sure our string is not empty. After we make sure that, we tell the photon network to create room and make sure the created room is the user input room name.

```
public void CreateRoom()
{
    if(!string.IsNullOrEmpty(roomNameInput.text))
    {
        RoomOptions options = new RoomOptions();
        options.MaxPlayers = 8;

        PhotonNetwork.CreateRoom(roomNameInput.text, options);

        CloseMenus();
        loadingText.text = "Creating Room...";
        loadingScreen.SetActive(true);


    }
}
```



Room Browsing

Once a room is created, Photon will automatically try to join you a room, so what we need is instead on just joining a room, we wanted a list what room are available in our game. We also need to consider all the time if there is a change to the list of rooms in the lobby and refresh in the background.

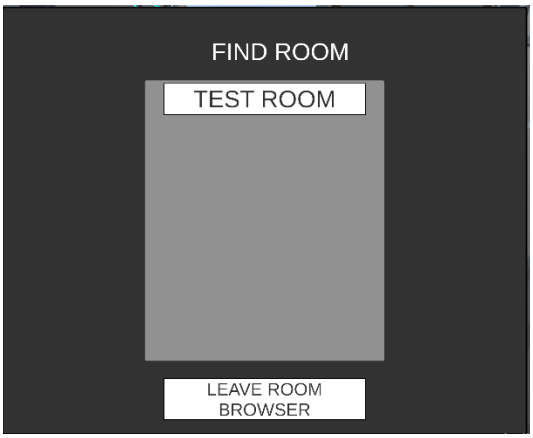
```
public override void OnRoomListUpdate(List<RoomInfo> roomList)
{
    foreach (RoomButton rb in allRoomButtons)
    {
        Destroy(rb.gameObject);
    }
    allRoomButtons.Clear();

    theRoomButton.gameObject.SetActive(false);

    for(int i = 0; i < roomList.Count; i++)
    {
        if(roomList[i].PlayerCount != roomList[i].MaxPlayers && !roomList[i].RemovedFromList)
        {
            RoomButton newButton = Instantiate(theRoomButton, theRoomButton.transform.parent);
            newButton.SetButtonsDetails(roomList[i]);
            newButton.gameObject.SetActive(true);

            allRoomButtons.Add(newButton);

        }
    }
}
```



Joining Room

On join room, we also need to connect it to our photon network and need to tell the name of the room we want to join. We handled most of the join room work when the room was created.

```
public void JoinRoom(RoomInfo inputInfo)
{
    PhotonNetwork.JoinRoom(inputInfo.Name);

    CloseMenus();
    loadingText.text = "Joining Room";
    loadingScreen.SetActive(true);
}
```

Listing players and updating the list.

When our player successfully joins our room, we need to list all the available players on the room and update the list every time if someone joins or leaves the room created. We handle the update list just when someone leaves the room destroy the list and make it again.

```
private void ListAllPlayers()
{
    foreach(TMP_Text player in allPlayerNames)
    {
        Destroy(player.gameObject);
    }
    allPlayerNames.Clear();

    Player[] players = PhotonNetwork.PlayerList;
    for(int i = 0; i < players.Length; i++)
    {
        TMP_Text newPlayerLebel = Instantiate(playerNameLabel, playerNameLabel.transform.parent);
        newPlayerLebel.text = players[i].NickName;
        newPlayerLebel.gameObject.SetActive(true);

        allPlayerNames.Add(newPlayerLebel);
    }
}
```

```
public override void OnPlayerEnteredRoom(Player newPlayer)
{
    TMP_Text newPlayerLebel = Instantiate(playerNameLabel, playerNameLabel.transform.parent);
        newPlayerLebel.text = newPlayer.NickName;
        newPlayerLebel.gameObject.SetActive(true);

        allPlayerNames.Add(newPlayerLebel);
}

public override void OnPlayerLeftRoom(Player otherPlayer)
{
  ListAllPlayers();
}
```

Players Nickname

To store our nickname, we can store it in our system using PlayerPreefs since we're just storing a simple string of text for the name of the player. And make sure we're storing the right thing and regularly telling the network to set our name.

```
public void SetNickname()
{
    if(!string.IsNullOrEmpty(nameInput.text))
    {
        PhotonNetwork.NickName = nameInput.text;

        PlayerPrefs.SetString("playerName", nameInput.text);

        CloseMenus();
        menuButtons.SetActive(true);

        hasSetNick = true;

    }
}
```

```
public override void OnJoinedLobby()
{
    CloseMenus();
    menuButtons.SetActive(true);

    PhotonNetwork.NickName = Random.Range(0, 1000).ToString();

    if(!hasSetNick)
    {
        CloseMenus();
        nameInputScreen.SetActive(true);

        if(PlayerPrefs.HasKey("playerName"))
        {
            nameInput.text = PlayerPrefs.GetString("playerName");
        }
    }
    else
    {
        PhotonNetwork.NickName = PlayerPrefs.GetString("playerName");
    }
}
```

Switching Masters

One of the biggest things to consider is what happens if the master player leaves the game. What we wanted to do is when the master leaves, someone new from the room will be the master and have access to start the game. Photon have a special function that allows us to detect when the master has switched.

```
public override void OnMasterClientSwitched(Player newMasterClient)
{
    if(PhotonNetwork.IsMasterClient)
    {
        startButton.SetActive(true);
    } else
    {
        startButton.SetActive(false);
    }
}
```