



photon

A cross platform multiplayer game development framework

Photon Realtime

Realtime Intro

Get Started

Photon Realtime is a fully managed service (SaaS) of Photon on-premises servers running in regions worldwide, ready for low latency multiplayer gaming around the globe, anytime.

Use the LoadBalancing API to match players to a shared game session (called "room") and transfer messages synchronously, in real-time, between connected players across platforms. All client SDKs can interact with each other, no matter if iOS, Android, web, console or standalone.

Region	Hosted in	Token
Asia	Singapore	asia
Australia	Melbourne	au
Canada, East	Montreal	cae
Chinese Mainland (See Instructions)	Shanghai	cn
Europe	Amsterdam	eu
India	Chennai	in
Japan	Tokyo	jp
Russia	Moscow	ru
Russia, East	Khabarovsk	rue
South America	Sao Paulo	sa
South Korea	Seoul	kr
USA, East	Washington D.C.	us
USA, West	San José	usw

Protocols

The Photon Server core supports the following protocols

- **reliable UDP** (based on eNET) and specially tuned for Client-2-Server architectures
- **Binary TCP**
- **Web Sockets**
- **HTTP**

The transfer protocol is very lean and slim. Photon wraps up the networking layer of each client platform for you. Have your game clients communicate cross-platform and across protocols. Put your data in hashtables and just send it. Photon does the de-/serialization, you dont!

PUN – Photon Unity Networking

PUN's Structure

Usually, you don't have to mind the structure of the PUN package but just for the overview, here's how things stack up. The PUN package wraps up three layers of APIs:

- The highest level is the PUN code, which implements Unity-specific features like networked objects, RPCs and so on.
- The second level contains the logic to work with Photon servers, do matchmaking, callbacks and such. This is the Realtime API. This can be used on it's own already. You will notice a lot of overlap of topics between PUN and the Realtime API (a.k.a. LoadBalancing API) but that's fine.
- The lowest level is made up of DLL files, which contain the de/serialization, protocols and such.

Don't try to reinvent the wheel, at least when you're still in the initial phases of a topic.

Both the LoadBalancing API (Photon Realtime) and PUN can be used to implement the networked multiplayer system. However, implementing the former is a little more complicated compared to latter.

In PUN players are connected to a common sever and so don't need to connect one to one while in Photon Realtime, communication between players is established by direct connection.

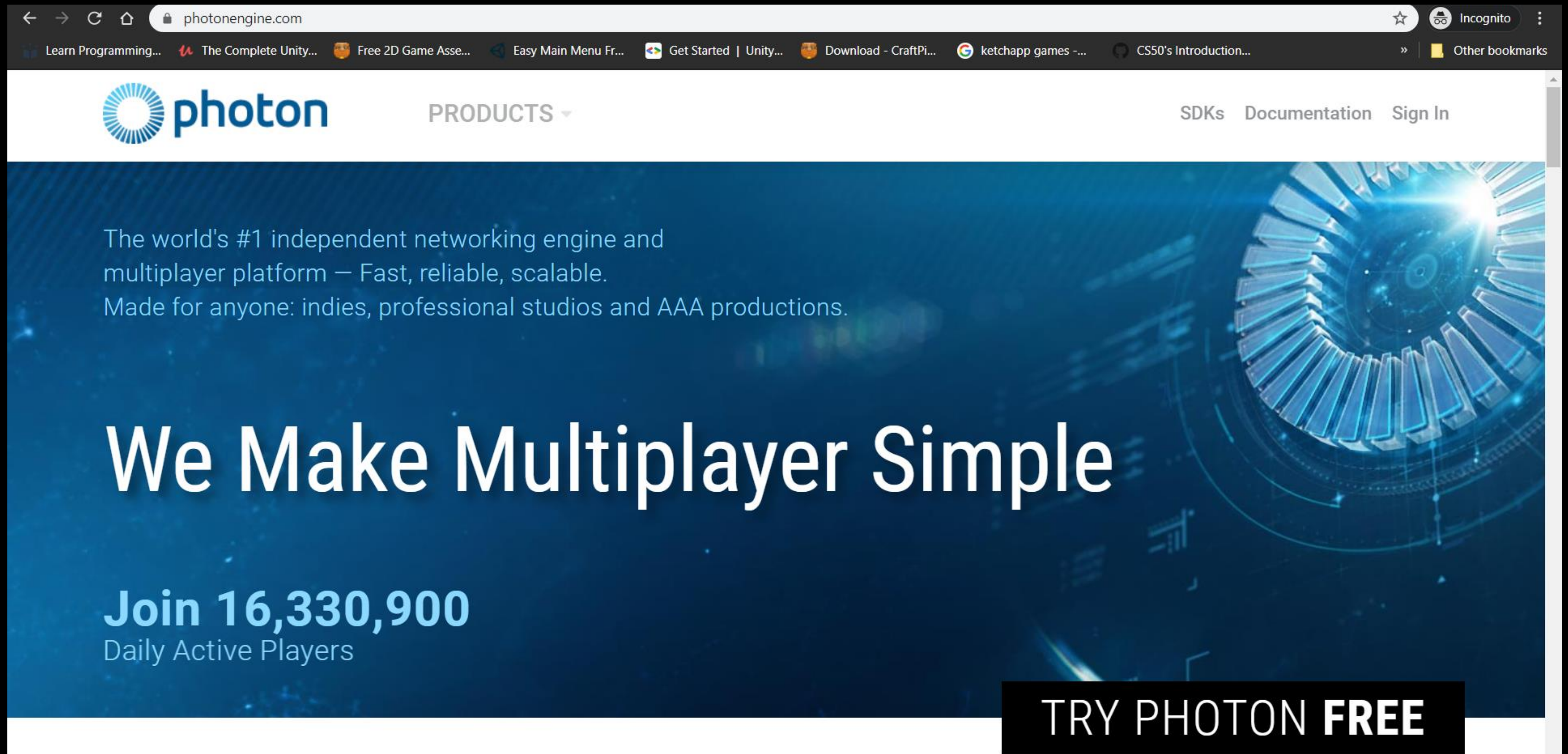
As expected, both of these support connection between players across different platforms.

Room

Lobby

Enough of theory.
Let's get real.

www.photonengine.com -> *“TRY PHOTON FREE”*



The screenshot shows the Photon Engine website in a web browser. The browser's address bar displays 'photonengine.com'. The website's header features the Photon logo on the left, a 'PRODUCTS' dropdown menu in the center, and links for 'SDKs', 'Documentation', and 'Sign In' on the right. The main content area has a dark blue background with a glowing, futuristic circular graphic on the right. The text on the page reads: 'The world's #1 independent networking engine and multiplayer platform — Fast, reliable, scalable. Made for anyone: indies, professional studios and AAA productions.' Below this, a large white headline states 'We Make Multiplayer Simple'. Further down, it says 'Join 16,330,900 Daily Active Players'. At the bottom right, a black button with white text says 'TRY PHOTON FREE'.

photonengine.com

Learn Programming... The Complete Unity... Free 2D Game Asse... Easy Main Menu Fr... Get Started | Unity... Download - CraftPi... ketchapp games -... CS50's Introduction...

Other bookmarks

photon PRODUCTS

SDKs Documentation Sign In

The world's #1 independent networking engine and multiplayer platform — Fast, reliable, scalable.
Made for anyone: indies, professional studios and AAA productions.

We Make Multiplayer Simple

Join **16,330,900**
Daily Active Players

TRY PHOTON FREE

Create new app.

The screenshot shows the Photon Engine dashboard at dashboard.photonengine.com/en-US/publiccloud. The page title is "Your Photon Cloud Applications". Below the title, there are four filters: "Show" (set to "All Apps"), "in Status" (set to "Active"), "Sort by" (set to "Peak CCU"), and "Order" (set to "Descending"). A red rectangular box highlights a button labeled "CREATE A NEW APP". Below the filters, there are two application cards. The first card is for a "PUN" application named "Shoot Out" with an App ID of "149157c5-...". It shows "Peak CCU" as 0 and "Traffic used" as 0%. The second card is for a "VOICE" application named "Fountane Ville" with an App ID of "c2a688c3-...". It also shows "Peak CCU" as 0 and "Traffic used" as 0%. Both cards have buttons for "ANALYZE", "MANAGE", "-/+ CCU", and "ADD COUPON".

dashboard.photonengine.com/en-US/publiccloud

Apps Learn Programming... The Complete Unity... Free 2D Game Asse... Easy Main Menu Fr... Get Started | Unity... Download - CraftPi... ketchapp games -... CS50's Introduction... Other bookmarks

photon PRODUCTS SDKs Documentation

Your Photon Cloud Applications

Show in Status Sort by Order

All Apps Active Peak CCU Descending

CREATE A NEW APP

PUN 20 CCU	VOICE 20 CCU
Shoot Out App ID: 149157c5-... Peak CCU: 0 Traffic used: 0%	Fountane Ville App ID: c2a688c3-... Peak CCU: 0 Traffic used: 0%
ANALYZE MANAGE -/+ CCU ADD COUPON	ANALYZE MANAGE -/+ CCU ADD COUPON

Photon Type = Photon PUN.
Copy AppID from dashboard.

Create a New Application

The application defaults to the **Free Plan**.
You can change the plan at any time.

Photon Type *

Photon PUN

Name *

Your application's name

Description

Short description, 1024 chars max.

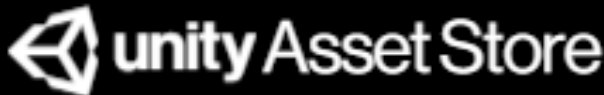
Url

http://enter.your-url.here/






CREATE

or [go back](#) to the application list.

Import PUN into your Unity project




Search for assets



Home > Tools > Network > PUN 2 - FREE

You downloaded this item on Oct 4, 2019. [Write a Review](#)



EXIT GAMES

PUN 2 - FREE

FREE

★★★★☆ 93 user reviews

[Download](#)

So far so good

★★★★★

Very easy to use and understand Haven't had a problem so far Nice work!

trin-o on previous version 2.15 a month ago

[Read more reviews](#)

Popular Tags

Add a new tag right now?

[Add tags](#)

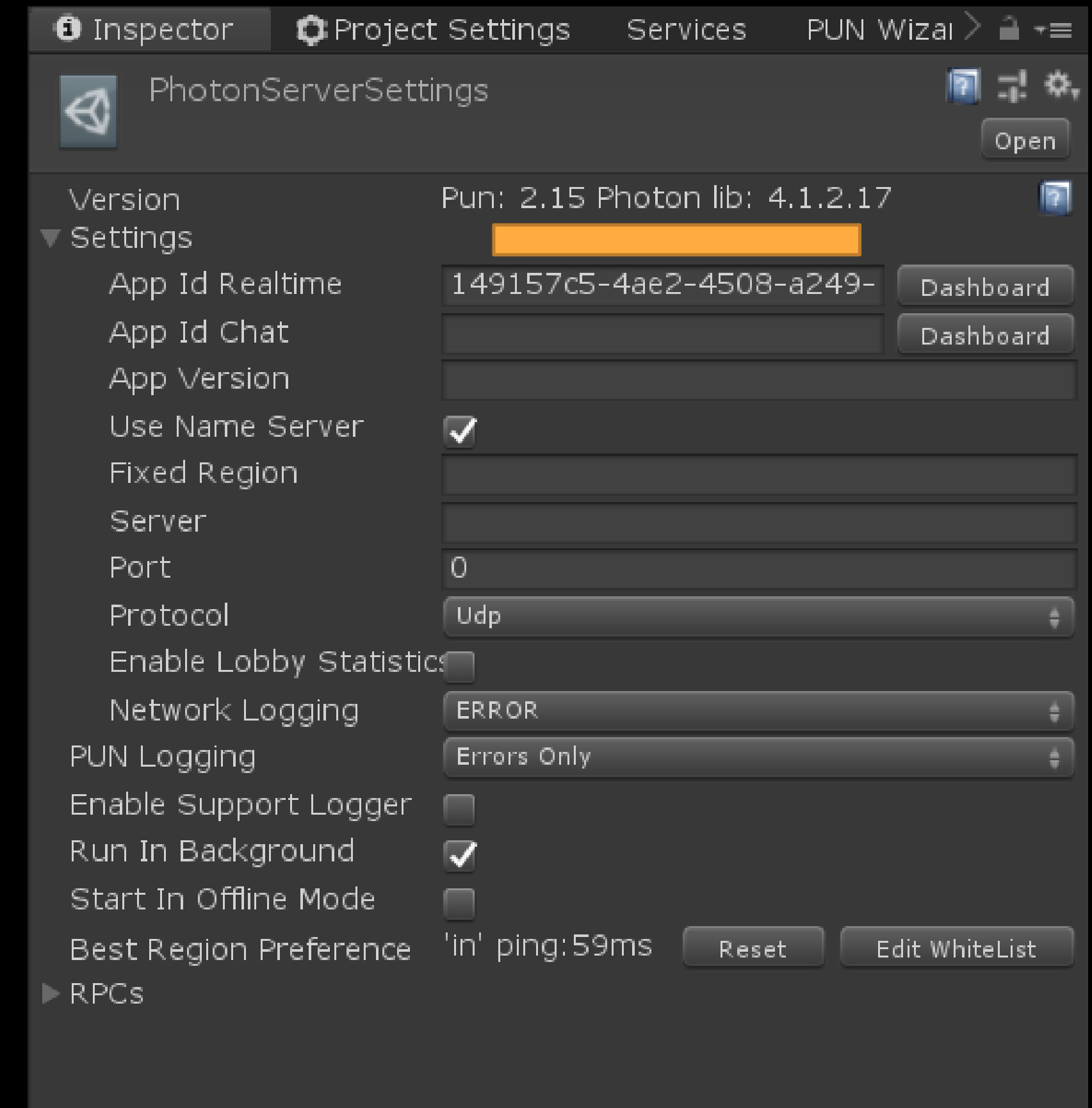
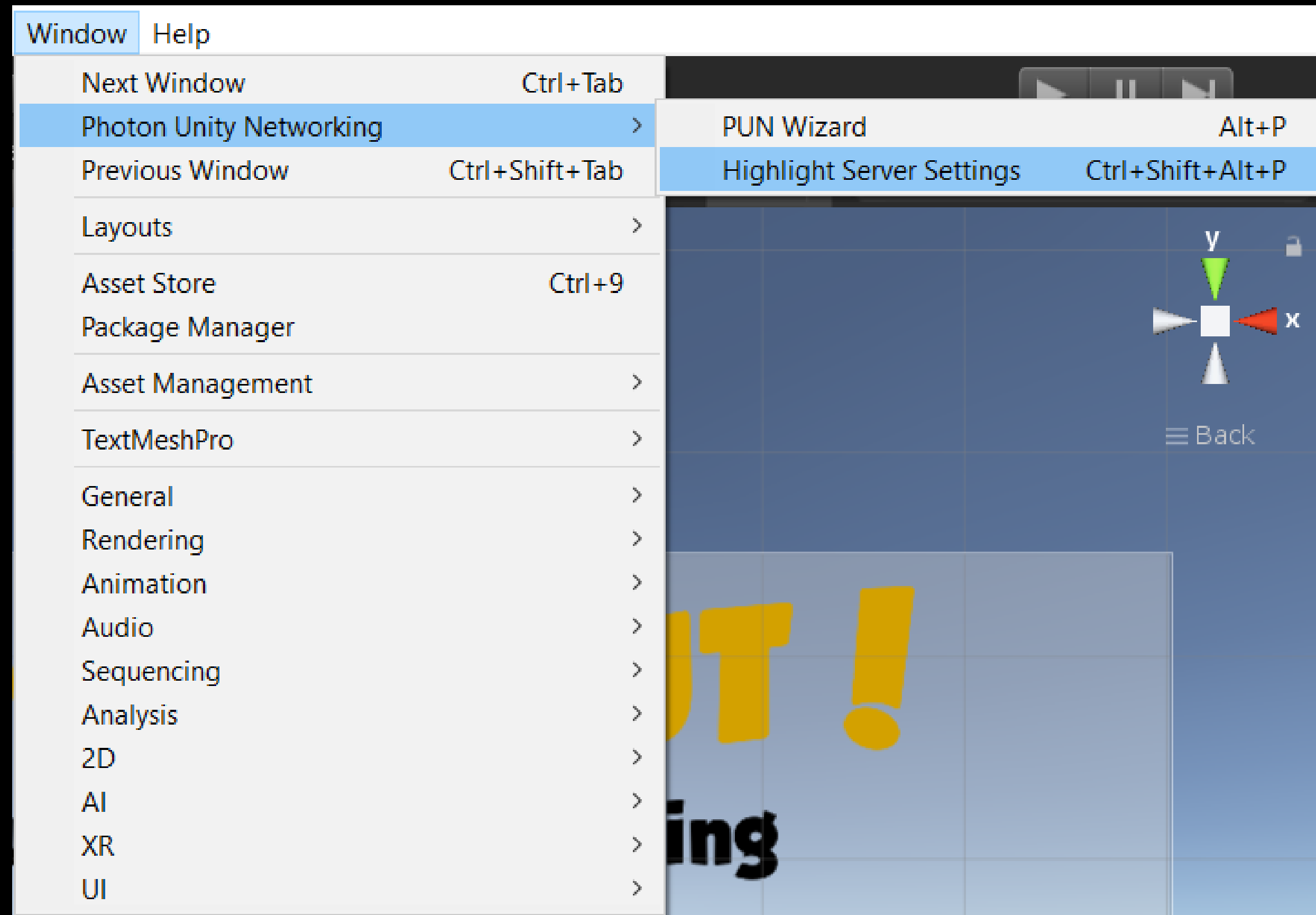
Note:

Contains breaking changes to existing PUN Classic (v1) projects. Please read our [PUN 2 migration notes](#). In doubt: keep [PUN Classic](#). PUN is tested and supported for official Unity

Package contents

14.0 MB

Link up the server you want to connect to, either Photon's or self-hosted.



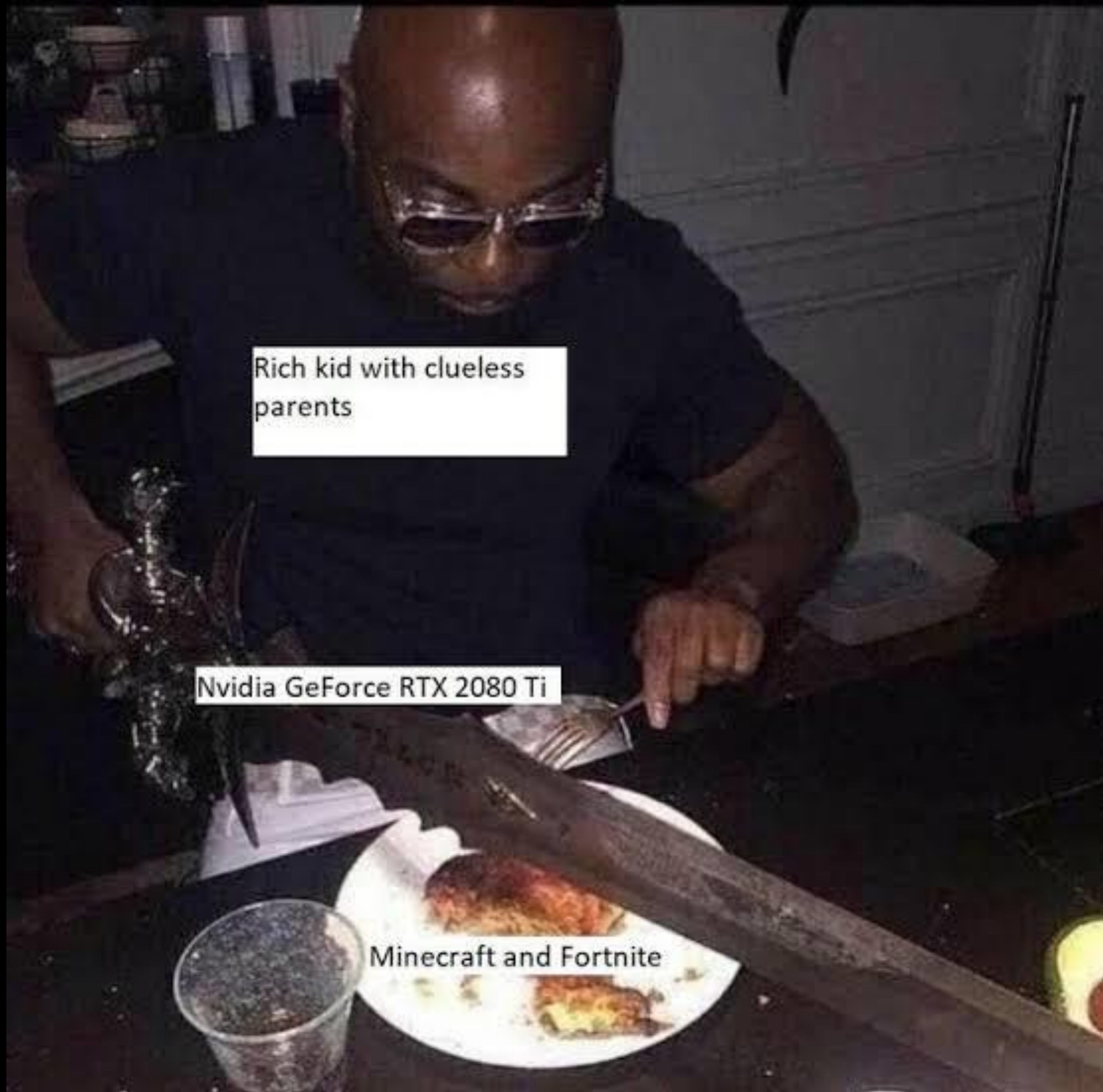
Establish network connection.

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

0 references
public class NetworkManager : MonoBehaviourPunCallbacks
{
    0 references
    private void Start()
    {
        PhotonNetwork.ConnectUsingSettings();
    }

    11 references
    public override void OnConnectedToMaster()
    {
        Debug.Log("Connected to " + PhotonNetwork.CloudRegion + " server!");
    }
}
```

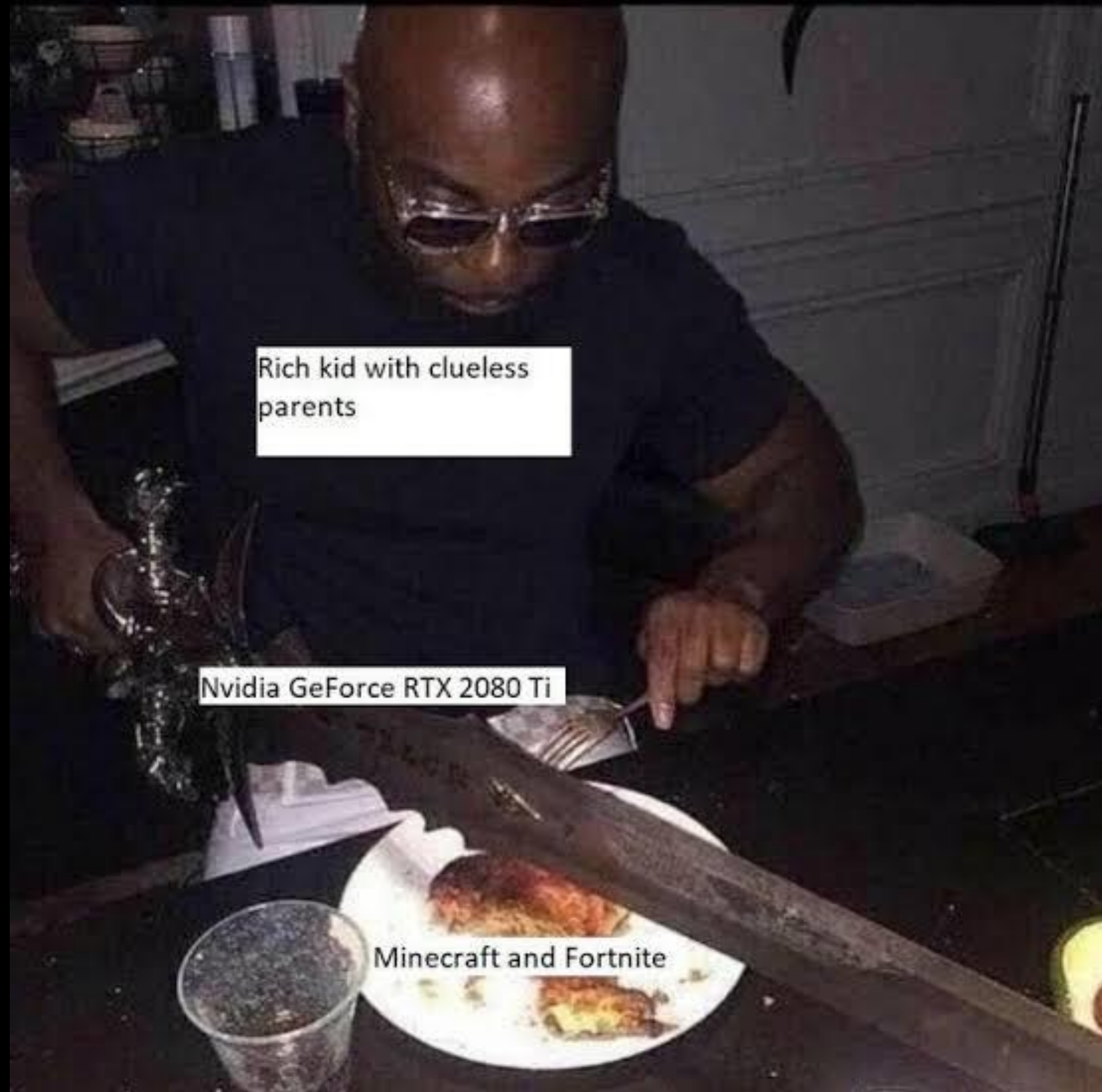
Quick Start Matchmaking



Rich kid with clueless
parents

Nvidia GeForce RTX 2080 Ti

Minecraft and Fortnite



Rich kid with clueless
parents

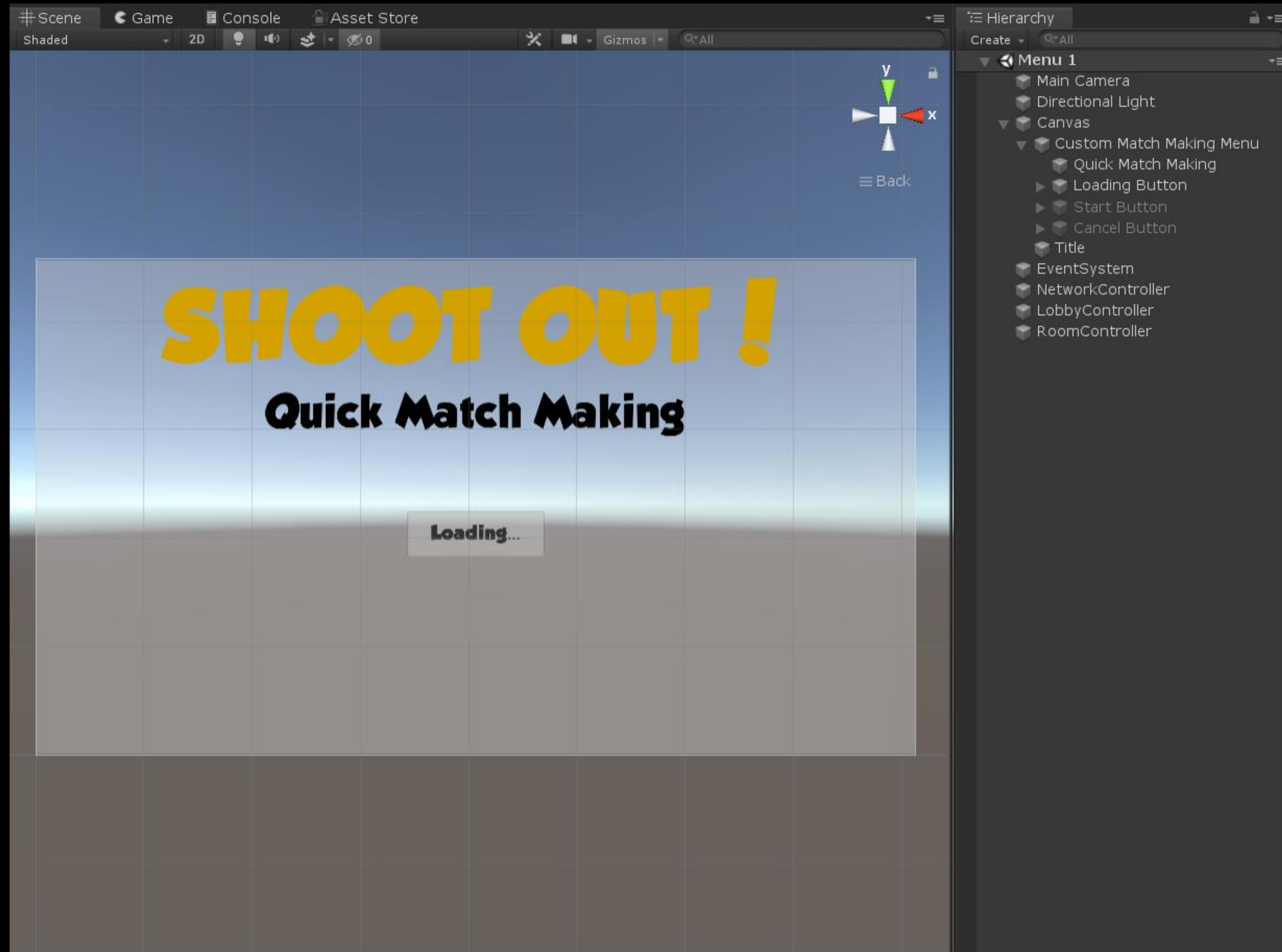
Nvidia GeForce RTX 2080 Ti

Minecraft and Fortnite

In QM (Quick Start Matchmaking), players directly enter rooms and play against each other without joining a lobby. So players are connected to each other only as long as the game is live and running. Also, the room is randomly assigned and player does not have any control over that.

This is not always bad. When you want to pair two global players just for a match, this is a good option.

Setup the menu environment.



Lobby Controller, creating and joining room.

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

0 references
public class LobbyControllerQM : MonoBehaviourPunCallbacks
{
    public GameObject loadingButton;
    public GameObject startButton;
    public GameObject cancelButton;

    12 references
    public override void OnConnectedToMaster()
    {
        PhotonNetwork.AutomaticallySyncScene = true;
        loadingButton.SetActive(false);
        startButton.SetActive(true);
    }

    0 references
    public void StartGame()
    {
        startButton.SetActive(false);
        cancelButton.SetActive(true);
        PhotonNetwork.JoinRandomRoom();
    }
}
```

```
public override void OnJoinRandomFailed(short returnCode, string message)
{
    CreateRoom();
}

2 references
private void CreateRoom()
{
    string roomName = "Room: " + Random.Range(1, 10000).ToString();
    RoomOptions roomOptions = new RoomOptions()
    {
        IsOpen = true,
        IsVisible = true,
        MaxPlayers = (byte)2
    };
    PhotonNetwork.CreateRoom(roomName, roomOptions);
}

10 references
public override void OnCreateRoomFailed(short returnCode, string message)
{
    CreateRoom();
}

0 references
public void CancelGame()
{
    cancelButton.SetActive(false);
    startButton.SetActive(true);
    PhotonNetwork.LeaveRoom();
}
}
```

Starting the game.

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

0 references
public class RoomControllerQM : MonoBehaviourPunCallbacks
{
    1 reference
    public override void OnEnable()
    {
        PhotonNetwork.AddCallbackTarget(this);
    }

    3 references
    public override void OnDisable()
    {
        PhotonNetwork.RemoveCallbackTarget(this);
    }

    14 references
    public override void OnJoinedRoom()
    {
        StartGame();
    }

    1 reference
    private void StartGame()
    {
        if(PhotonNetwork.IsMasterClient)
        {
            PhotonNetwork.LoadLevel(1);
        }
    }
}
```


Delay Start Matchmaking

NOPE



STILL WAITING

STILL WAITING

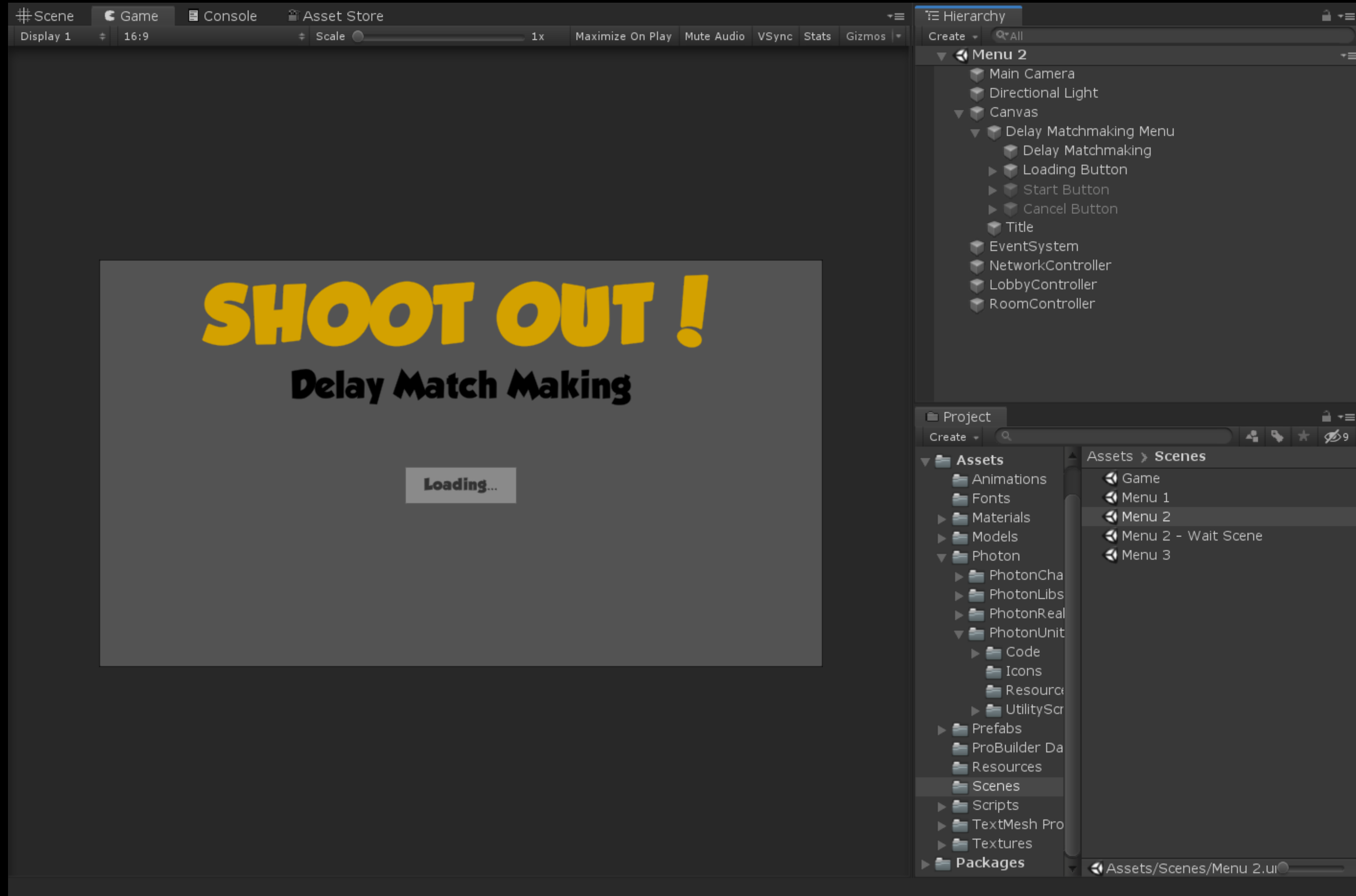




How long are you going to wait for your partner to join the game? How about we set a timer after which the game is automatically started once at-least two players are in the room?

Delay Matchmaking lets you wait for only a limited amount of time for your friends to join the game and also gives you the flexibility to exit a game within the time.

Setup the menu environment.



Lobby Controller, creating and joining room.

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

0 references
public class LobbyControllerQM : MonoBehaviourPunCallbacks
{
    public GameObject loadingButton;
    public GameObject startButton;
    public GameObject cancelButton;

    12 references
    public override void OnConnectedToMaster()
    {
        PhotonNetwork.AutomaticallySyncScene = true;
        loadingButton.SetActive(false);
        startButton.SetActive(true);
    }

    0 references
    public void StartGame()
    {
        startButton.SetActive(false);
        cancelButton.SetActive(true);
        PhotonNetwork.JoinRandomRoom();
    }
}
```

```
public override void OnJoinRandomFailed(short returnCode, string message)
{
    CreateRoom();
}

2 references
private void CreateRoom()
{
    string roomName = "Room: " + Random.Range(1, 10000).ToString();
    RoomOptions roomOptions = new RoomOptions()
    {
        IsOpen = true,
        IsVisible = true,
        MaxPlayers = (byte)2
    };
    PhotonNetwork.CreateRoom(roomName, roomOptions);
}

10 references
public override void OnCreateRoomFailed(short returnCode, string message)
{
    CreateRoom();
}

0 references
public void CancelGame()
{
    cancelButton.SetActive(false);
    startButton.SetActive(true);
    PhotonNetwork.LeaveRoom();
}
}
```

Loading the wait scene.

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

0 references
public class RoomControllerQM : MonoBehaviourPunCallbacks
{
    1 reference
    public override void OnEnable()
    {
        PhotonNetwork.AddCallbackTarget(this);
    }

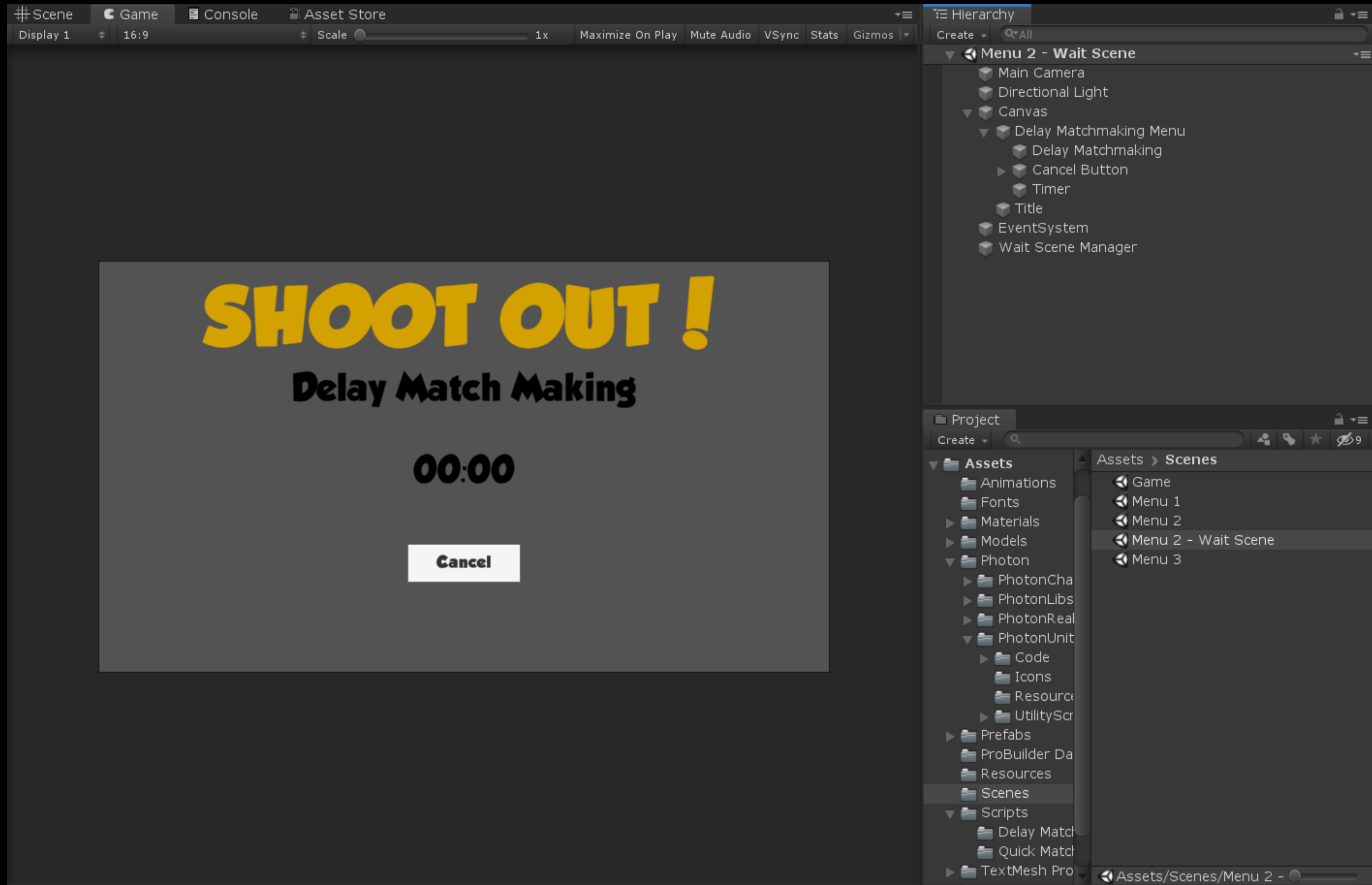
    3 references
    public override void OnDisable()
    {
        PhotonNetwork.RemoveCallbackTarget(this);
    }

    14 references
    public override void OnJoinedRoom()
    {
        StartGame();
    }

    1 reference
    private void StartGame()
    {
        if(PhotonNetwork.IsMasterClient)
        {
            PhotonNetwork.LoadLevel(1);
        }
    }
}
```

Let's now make the wait scene.

Setup the waiting room environment.



Wait Scene Manager (Add PhotonView along with this script)

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

0 references
public class WaitSceneManager : MonoBehaviourPunCallbacks
{
    public TextMeshProUGUI timer;
    public int maximumTime;

    private int currentTime;
    private PhotonView photonView;

    0 references
    private void Start()
    {
        photonView = GetComponent<PhotonView>();
        currentTime = maximumTime;
        timer.text = "00:" + currentTime.ToString();
    }

    10 references
    public override void OnPlayerEnteredRoom(Player newPlayer)
    {
        photonView.RPC("CoroutineCaller", RpcTarget.All);
    }

    10 references
    public override void OnPlayerLeftRoom(Player otherPlayer)
    {
        StopAllCoroutines();
        currentTime = maximumTime;
        timer.text = "00:" + currentTime.ToString();
    }
}
```

```
[PunRPC]
0 references
private void CoroutineCaller()
{
    StartCoroutine("StartTimer");
}

0 references
private IEnumerator StartTimer()
{
    while (true)
    {
        yield return new WaitForSeconds(1);
        currentTime -= 1;
        timer.text = "00:" + currentTime.ToString();

        if (currentTime == 0)
            StartGame();
    }
}

1 reference
private void StartGame()
{
    if (PhotonNetwork.IsMasterClient)
    {
        PhotonNetwork.LoadLevel("Game");
    }
}
```

Custom Matchmaking

Want some more?

**If yes, we will give
you more!**

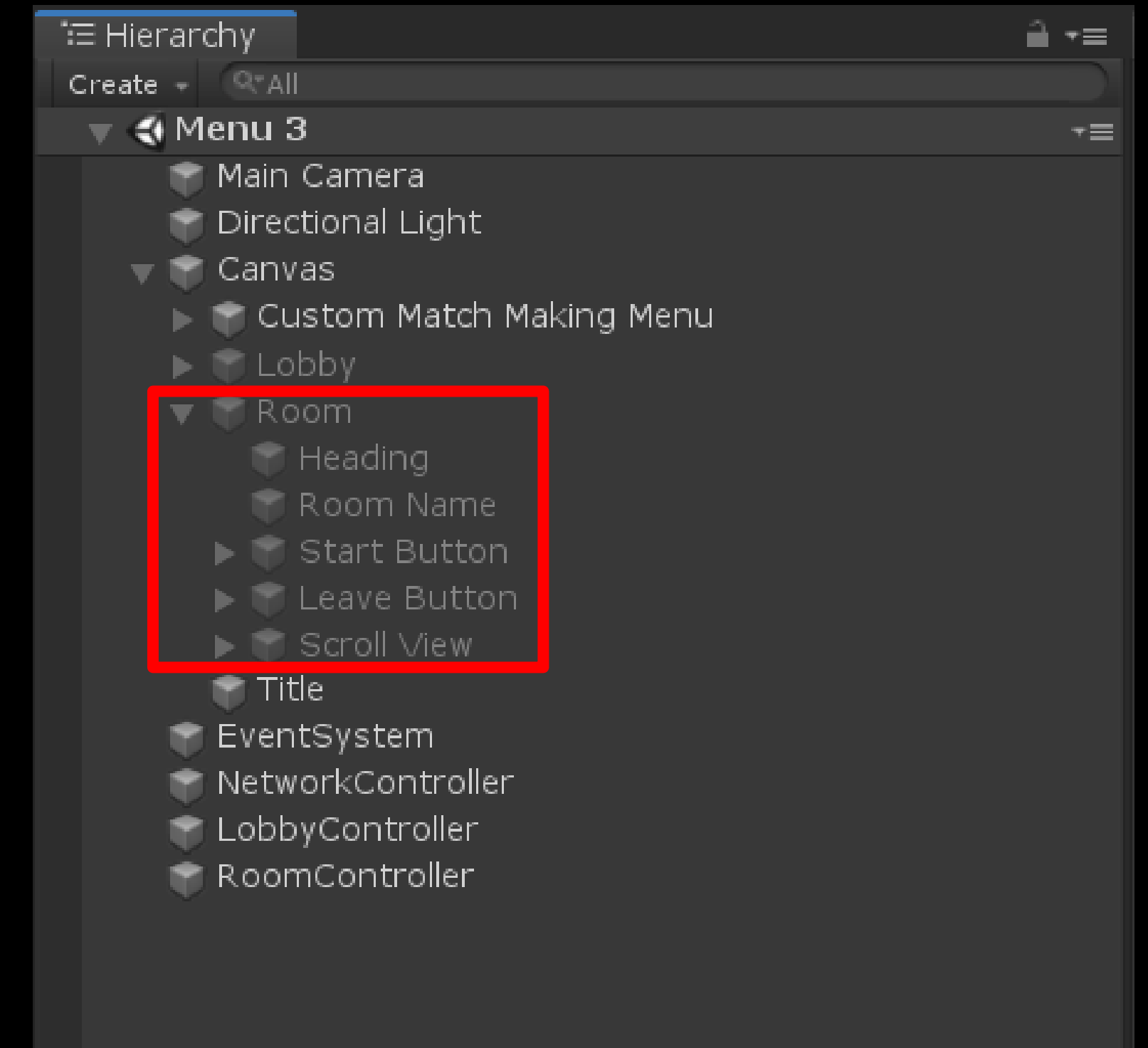
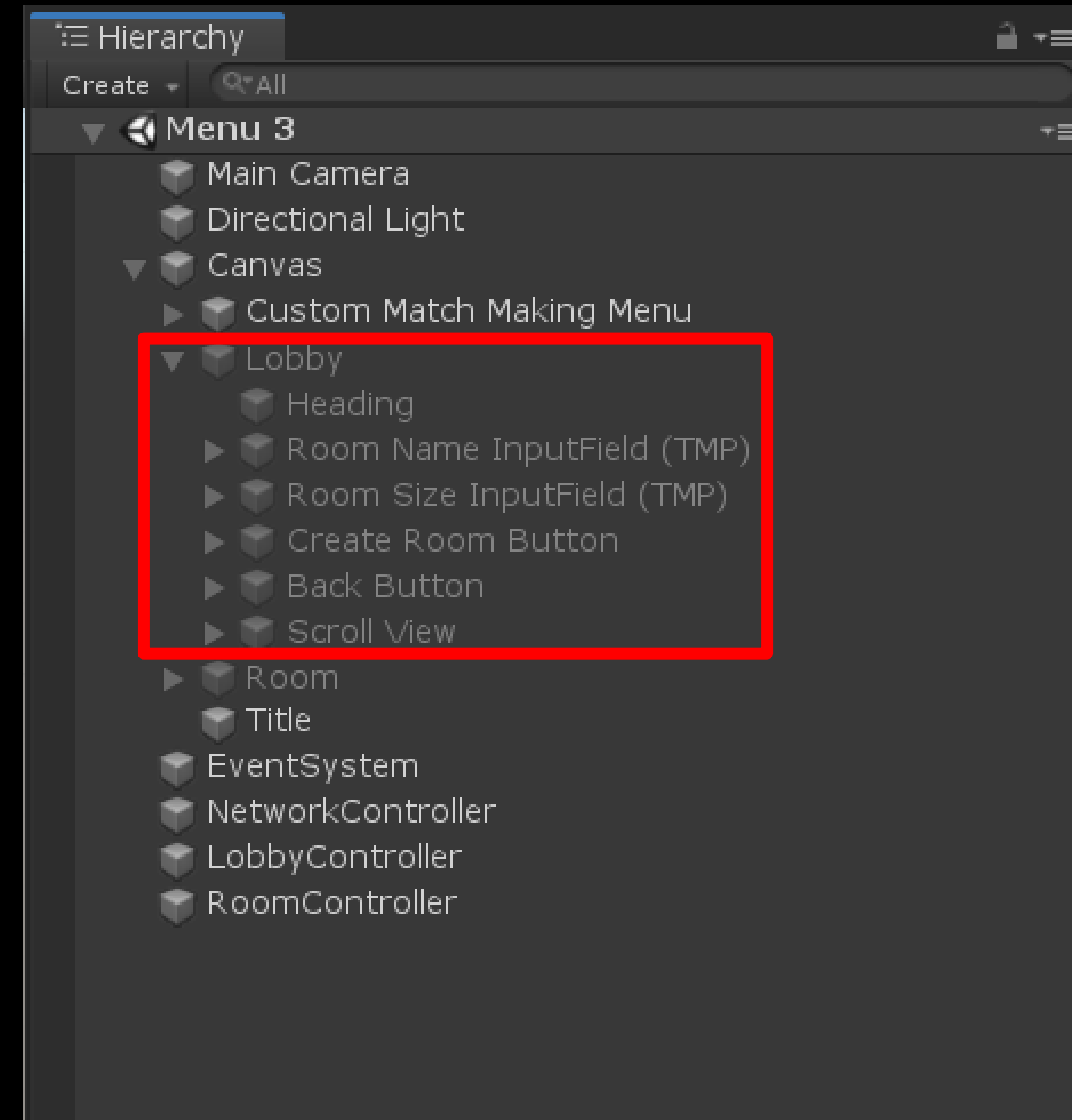
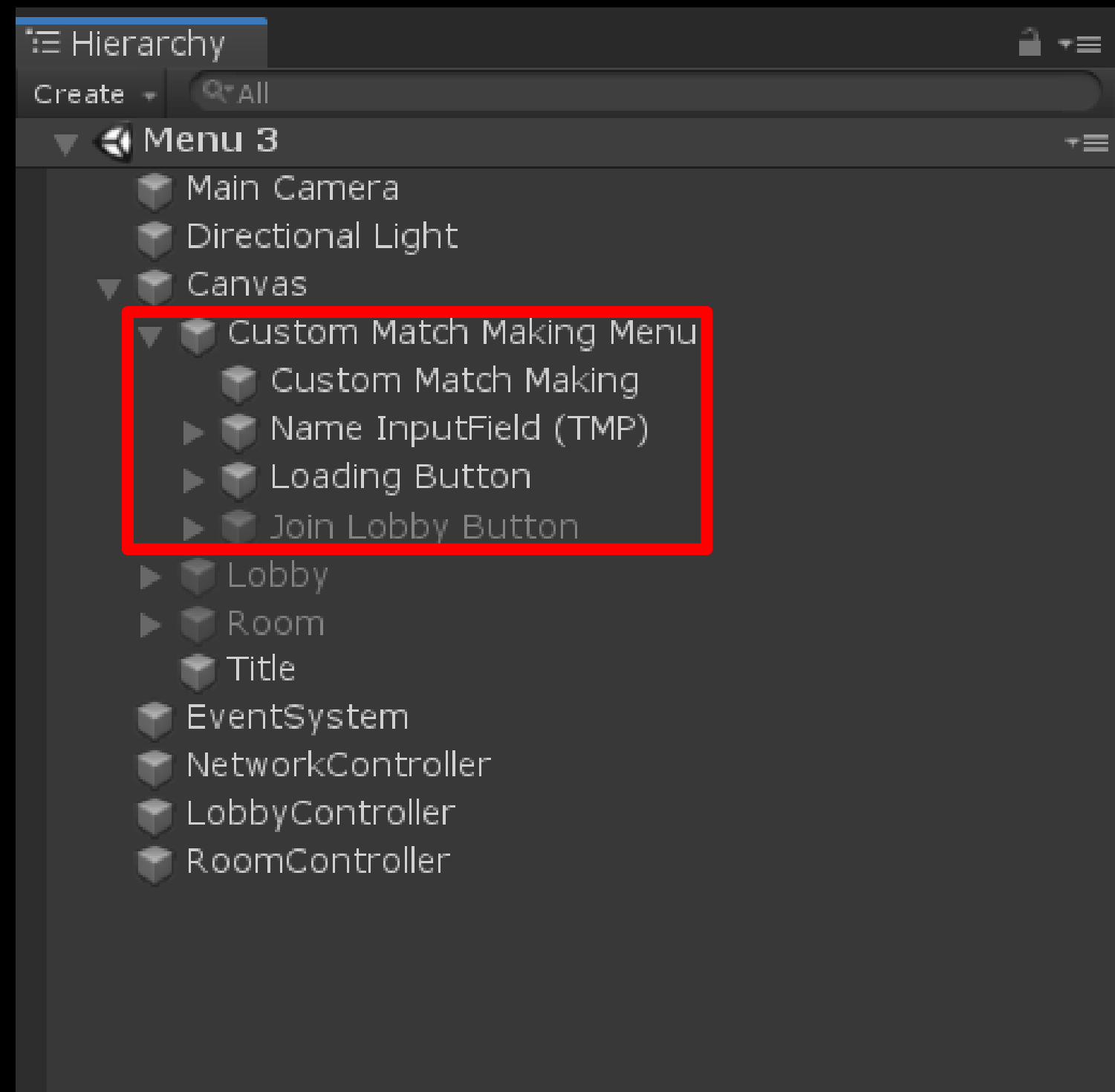
memecreator.org



Want to have more flexibility?
Want to have much more customised
experienced?

The obvious trade off is that you got to put
in more efforts, a little more complicated,
but a lot more flexible and customisable.
Let's go for it.

Setup the entire menu - Sign-up page, Lobby, Room.



Lobby Controller.

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

0 references
public class LobbyController : MonoBehaviourPunCallbacks
{
    public GameObject joinLobbyButton;
    public GameObject lobbyPanel;
    public GameObject mainPanel;
    public TMP_InputField playerNameInput;
    public Transform roomsContainer;
    public GameObject roomListingPrefab;

    private string roomName;
    private int roomSize;
    private List<RoomInfo> roomListings;

    13 references
    public override void OnConnectedToMaster()
    {
        PhotonNetwork.AutomaticallySyncScene = true;
        joinLobbyButton.SetActive(true);
        roomListings = new List<RoomInfo>();

        string defaultName = "Player " + UnityEngine.Random.Range(0, 1000).ToString();
        PhotonNetwork.NickName = PlayerPrefs.GetString("Player Name", defaultName);
        playerNameInput.text = PhotonNetwork.NickName;
    }
}
```

```
0 references
public void PlayerNameUpdate(string nameInput)
{
    PhotonNetwork.NickName = nameInput;
    PlayerPrefs.SetString("Player Name", nameInput);
}
```

```
0 references
public void JoinLobby()
{
    lobbyPanel.SetActive(true);
    mainPanel.SetActive(false);
    PhotonNetwork.JoinLobby();
}
```

```
11 references
public override void OnRoomListUpdate(List<RoomInfo> roomList)
{
    int tempIndex;
    foreach (RoomInfo room in roomList)
    {
        Debug.Log(room);
        if (roomListings != null)
        {
            tempIndex = roomListings.FindIndex(ByName(room.Name));
        }
        else
        {
            tempIndex = -1;
        }

        if (tempIndex != -1)
        {
            roomList.RemoveAt(tempIndex);
            Destroy(roomsContainer.GetChild(tempIndex).gameObject);
        }

        if (room.PlayerCount > 0)
        {
            roomListings.Add(room);
            ListRoom(room);
        }
    }
}
```

Lobby Controller.

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

0 references
public class LobbyController : MonoBehaviourPunCallbacks
{
    public GameObject joinLobbyButton;
    public GameObject lobbyPanel;
    public GameObject mainPanel;
    public TMP_InputField playerNameInput;
    public Transform roomsContainer;
    public GameObject roomListingPrefab;

    private string roomName;
    private int roomSize;
    private List<RoomInfo> roomListings;

    13 references
    public override void OnConnectedToMaster()
    {
        PhotonNetwork.AutomaticallySyncScene = true;
        joinLobbyButton.SetActive(true);
        roomListings = new List<RoomInfo>();

        string defaultName = "Player " + UnityEngine.Random.Range(0, 1000).ToString();
        PhotonNetwork.NickName = PlayerPrefs.GetString("Player Name", defaultName);
        playerNameInput.text = PhotonNetwork.NickName;
    }
}
```

```
0 references
public void PlayerNameUpdate(string nameInput)
{
    PhotonNetwork.NickName = nameInput;
    PlayerPrefs.SetString("Player Name", nameInput);
}
```

```
0 references
public void JoinLobby()
{
    lobbyPanel.SetActive(true);
    mainPanel.SetActive(false);
    PhotonNetwork.JoinLobby();
}
```

```
11 references
public override void OnRoomListUpdate(List<RoomInfo> roomList)
{
    int tempIndex;
    foreach (RoomInfo room in roomList)
    {
        Debug.Log(room);
        if (roomListings != null)
        {
            tempIndex = roomListings.FindIndex(ByName(room.Name));
        }
        else
        {
            tempIndex = -1;
        }

        if (tempIndex != -1)
        {
            roomList.RemoveAt(tempIndex);
            Destroy(roomsContainer.GetChild(tempIndex).gameObject);
        }

        if (room.PlayerCount > 0)
        {
            roomListings.Add(room);
            ListRoom(room);
        }
    }
}
```


Lobby Controller.

```
1 reference
private void ListRoom(RoomInfo room)
{
    if(room.IsOpen && room.IsVisible)
    {
        GameObject tempListing = Instantiate(roomListingPrefab, roomsContainer);
        RoomButton tempButton = tempListing.GetComponent<RoomButton>();
        tempButton.SetRoom(room.Name, room.MaxPlayers, room.PlayerCount);
    }
}

0 references
public void OnRoomNameChanged(string nameIn)
{
    roomName = nameIn;
}

0 references
public void OnRoomSizeChanged(string sizeIn)
{
    roomSize = int.Parse(sizeIn);
}
```

```
0 references
public void CreateRoom()
{
    RoomOptions roomOptions = new RoomOptions() { IsVisible = true, IsOpen = true, MaxPlayers = (byte)roomSize };
    PhotonNetwork.CreateRoom(roomName, roomOptions);
}

11 references
public override void OnCreateRoomFailed(short returnCode, string message)
{
    Debug.LogError("Failed to create room. Check if room name is unique.");
}

0 references
public void CancelMatchmaking()
{
    mainPanel.SetActive(true);
    lobbyPanel.SetActive(false);
    PhotonNetwork.LeaveLobby();
}

1 reference
private static System.Predicate<RoomInfo> ByName(string name)
{
    return delegate (RoomInfo room)
    {
        return room.Name == name;
    };
}
```

Delegates & Predicates

A delegate is an object which refers to a method or simply, it's a reference type variable that holds reference to methods.

Delegates in C# are similar to method pointers in C/C++.

A predicate is a method that contains a set of criteria and checks whether the passed parameter meets these criteria or not and returns a Boolean.

In simpler terms, **a predicate delegate is a way to reference a more complex class by using one of its basic member variables to identify it.**

Delegates & Predicates



A delegate is an object which refers to a method or simply, it's a reference type variable that holds reference to methods.

Delegates in C# are similar to method pointers in C/C++.

A predicate is a method that contains a set of criteria and checks whether the passed parameter meets these criteria or not and returns a Boolean.

In simpler terms, a predicate delegate is a way to reference a more complex class by using one of its basic member variables to identify it.

Room Controller.

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

0 references
public class RoomController : MonoBehaviourPunCallbacks
{
    public int gameSceneIndex;
    public GameObject lobbyPanel;
    public GameObject roomPanel;
    public GameObject startButton;
    public Transform playersListContainer;
    public GameObject playerButtonPrefab;
    public TextMeshProUGUI roomName;

    3 references
    private void ClearPlayerListings()
    {
        for(int i = playersListContainer.childCount - 1; i >= 0; i--)
        {
            Destroy(playersListContainer.GetChild(i).gameObject);
        }
    }

    3 references
    private void ListPlayers()
    {
        foreach(Player player in PhotonNetwork.PlayerList)
        {
            GameObject tempListing = Instantiate(playerButtonPrefab, playersListContainer);
            TextMeshProUGUI tempText = tempListing.transform.GetChild(0).GetComponent<TextMeshProUGUI>();
            tempText.text = player.NickName;
        }
    }
}
```

```
15 references
public override void OnJoinedRoom()
{
    roomPanel.SetActive(true);
    lobbyPanel.SetActive(false);

    roomName.text = PhotonNetwork.CurrentRoom.Name;

    if(PhotonNetwork.IsMasterClient)
    {
        startButton.SetActive(true);
    }
    else
    {
        startButton.SetActive(false);
    }

    ClearPlayerListings();
    ListPlayers();
}

9 references
public override void OnPlayerEnteredRoom(Player newPlayer)
{
    ClearPlayerListings();
    ListPlayers();
}
```

Room Controller.

```
10 references
public override void OnPlayerLeftRoom(Player newPlayer)
{
    ClearPlayerListings();
    ListPlayers();
    if(PhotonNetwork.IsMasterClient)
    {
        startButton.SetActive(true);
    }
}

0 references
public void StartGame()
{
    if(PhotonNetwork.IsMasterClient)
    {
        PhotonNetwork.CurrentRoom.IsOpen = false;
        PhotonNetwork.LoadLevel(gameSceneIndex);
    }
}

1 reference
private IEnumerator RejoinLobby()
{
    yield return new WaitForSeconds(1f);
    PhotonNetwork.JoinLobby();
}

0 references
public void BackOnClick()
{
    lobbyPanel.SetActive(true);
    roomPanel.SetActive(false);
    PhotonNetwork.LeaveRoom();
    PhotonNetwork.LeaveLobby();
    StartCoroutine(RejoinLobby());
}
}
```

Room Button.

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

2 references
public class RoomButton : MonoBehaviour
{
    public TextMeshProUGUI nameText;
    public TextMeshProUGUI sizeText;

    private string roomName;
    private int roomSize;
    private int playerCount;

    0 references
    public void JoinRoomOnClick()
    {
        PhotonNetwork.JoinRoom(roomName);
    }

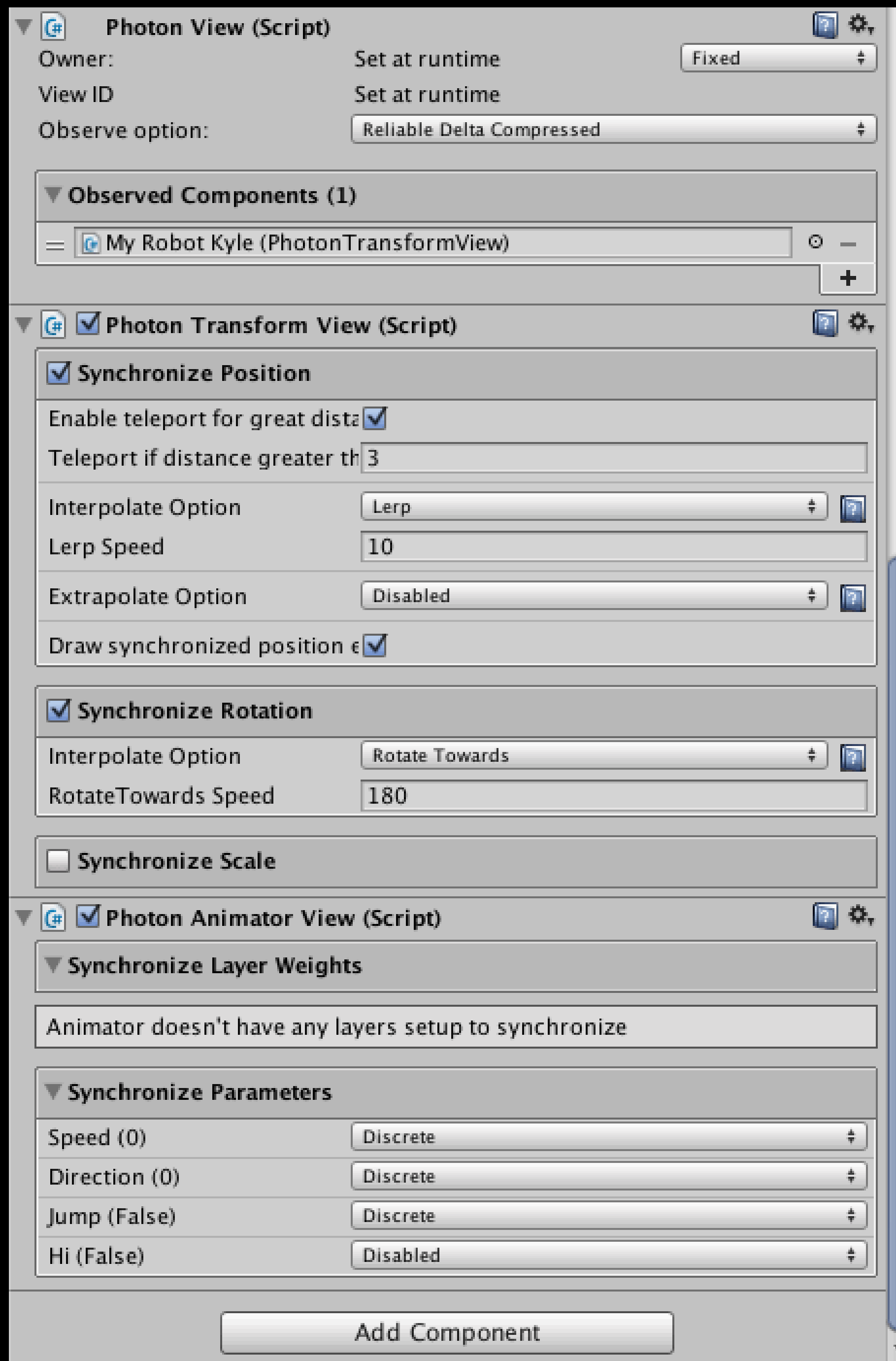
    1 reference
    public void SetRoom(string nameInput, int sizeInput, int countInput)
    {
        roomName = nameInput;
        roomSize = sizeInput;
        playerCount = countInput;
        nameText.text = nameInput;
        sizeText.text = countInput + "/" + sizeInput;
    }
}
```


Player Networking, a.k.a Game Setup

Player Instantiation

```
// we're in a room. spawn a character for the local player. it gets synced by using PhotonNetwork.Instantiate  
PhotonNetwork.Instantiate(this.playerPrefab.name, new Vector3(0f,5f,0f), Quaternion.identity, 0);
```

A gameobject has to be instantiated over the Photon Network server for it to be present across all the clients present in the room. The gameobject has to be in the resources folder if we are referencing it using a string, its name.



Client Communication

Inter-client communication can be established using 'PhotonView' component. Various aspects of Client-A's player gameobject like its transform and animation state will be sent to Client-B's player gameobject and vice versa.

Remote Procedure Calls - RPC

Remote Procedure Calls are method calls on remote clients in the same room. In simpler terms, a particular function that has to run on all the clients present in the room must have the '[PunRPC]' attribute.

Also, function call for a PunRPC function is somewhat different.

```
[PunRPC]
0 references
public void PlayShootEffect()
{
    this.muzzleFlash.Play();
    this.shell.Play();
}
```

```
GetComponent<PhotonView>().RPC("PlayShootEffect", RpcTarget.AllBuffered);
```

Thank You !