DOCUMENTATION

Thank you for downloading this example asset!

If you need help with anything I can help you!

If you have feedback or anything else, send me an email to: axlplay@gmail.com

“Menu” scene:

There everything is handled excepting the Player.

Let's start with the **MultiplayerMenuManager.cs:**

It's attached to the Menu scene at the gameobject “Multiplayer Manager” there we reference all the Menu screens so we can enable / disable them depending on the photon connection status!.

The first time you play the game it will ask you to enter your name and connect to photon, after that, it will remember your name and will connect automatically!

“Matchmaking” button will pair you up with players with the same level (you can change that in the lobby screen)

The game is quite simple, if you click the ground you'll move to that place, and if you click other player, you'll shoot him, if you hit him 10 times he will die.

Within MultiplayerMenuManager.cs:

the **Start method** will check if we had login before so we can connect automatically to photon.

“Lobby” region includes the UI update of the lobby screen, matchmaking, photon disconection, etc

ChangeUserLvl method is just to test the matchmaking based on player's level feature

Room Selector region includes all the things you can do in the Room List screen

Room region includes the **“ClickPlay” method** which starts the game for all users (only master client can click this button)

LoadGameScene method will load the level for all users since it's a method marked by [PunRPC] and it's being called with the following code:

```
photonView.RPC("LoadGameScene", RpcTarget.All);
```

That will use the “Photon View” component attached to the gameobject, and call that method across all the players in that room.

LeaveRoom method will leave the room, leave the chat room (photon handles that using different rooms) and destroying the UI messages.

Create Room region includes the custom room creation.

The magic of Photon is done in Callbacks region:

OnCreatedRoom(): there we handle the creation of the room, if it was created by a player we will only set the property “mode” to “waiting” new players to join the room, once the game starts “mode” will be set to “playing” so all the new players that join after the game started, will be redirected to the Game Scene.

If the room was created automatically by the level based matchmaking, we will set another property. “level” so we will only join to rooms that have our same level.

OnJoinRandomFailed() called when trying to play matchmaking but there's no open room.

OnLeftRoom () enable lobby UI and disable room UI

OnPlayerEnteredRoom() or OnPlayerLeftRoom() update in room UI

OnRoomListUpdate() when we are looking what rooms are open, we will constantly get the room list updated, it means that a room was created, or a room was closed and it's not anymore on the list.

OnConnectedToMaster() we have to join to the Lobby which is different from a room and most people get confused. We must be in a lobby to be able to use most features of photon. A room is where you play with other people, the lobby functionality is so you can join or create any room.

OnJoinedLobby() we are ready to set up our properties and connect to the Chat Client so we can later join to a room and be able to chat with other players in the same room.

Region chat:

We need to call `chatClient.Service();`

in an Update method to constantly update the chat and receive / send new messages

OnGetMessages method will get us the new messages

“Game” scene:

Player spawning is called in “GameManager”.cs when the level is loaded

ALL objects spawned through PhotonNetwork.Instantiate MUST be located in a resources

folder and have a PhotonView component attached to it.
Bullet script: Applies damage when it collides with a player on the MASTER client
It's detected only on the master client to avoid hacking or lag issues, this way , the master client will be the one handling this logic.

Player script:

OnEnable() when player is created it might be due to a player joining an in progress game, so we ask the current data of other players in the same room, so we can know at what position are them and start synchronizing the game from there, we also ask for the health so we can set their health on our client.

OnPhotonSerializeView() there we synchronize the player next position and rotation

The player shooting and animations are synchronized through Pun RPC's
When the player clicks, we call `photonView.RPC("Shoot", RpcTarget.All);`

so all players run the “Shoot” method.

Hope this example has helped you in some way, I had this demo on my pc and decided to share it and document it, so I would appreciate that you leave a review on the Asset Store.

This is a free asset so please be kind!

Thank you!

Oliver