TOO LONG DIDN'T READ

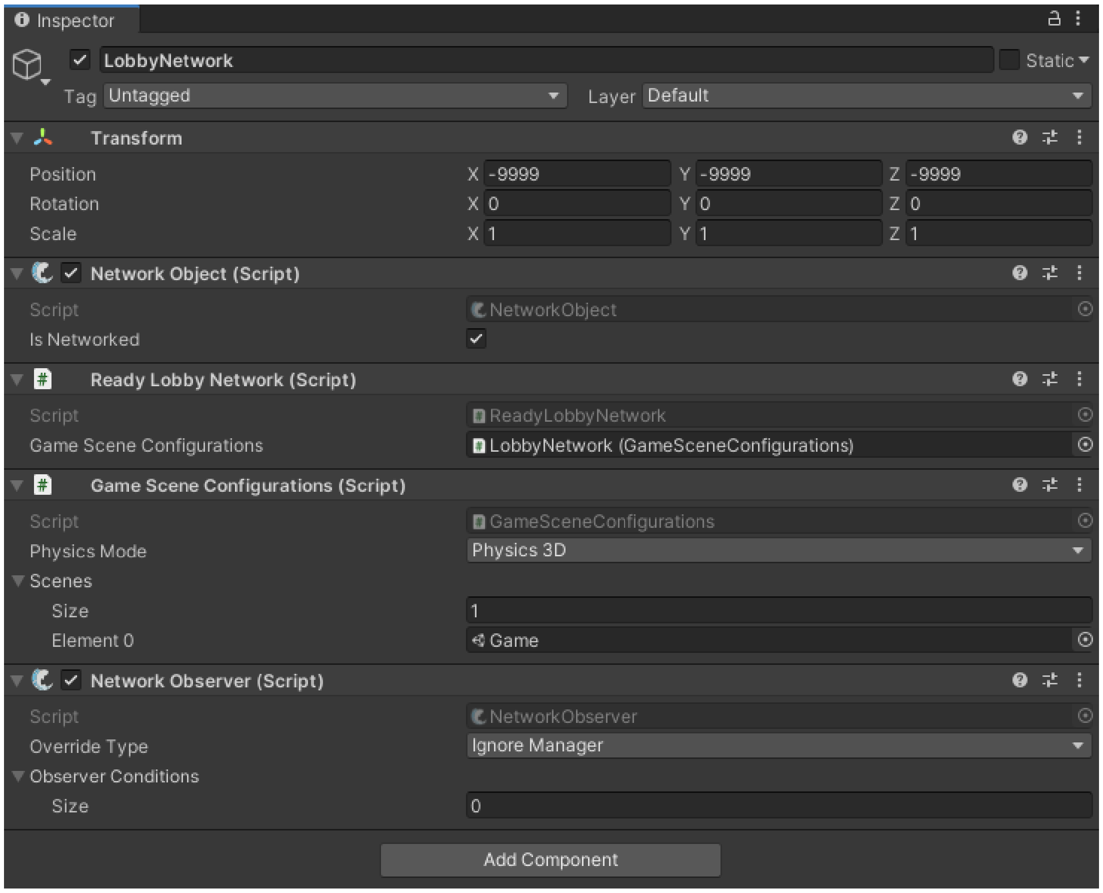
Open the LobbyNetwork prefab. Expand the Game Scene Configuration component. Add scenes you wish to load when a game starts under Scenes. Set which Physics Mode your game uses. That's it.

MODIFYING THE LOBBY

Select the LobbyNetwork prefab within your Project tab.

You will notice a GameSceneConfiguration script on it. If your game depends upon physics set the *Physics Mode* to whichever physics type your game will use.

Next is the Scenes property. Here you will specify which scenes to load when the room starts. I have added my demo scene, Game. You may also load and unload scenes during the middle of a game.



Inspector

LobbyNetwork — Static

Tag Untagged — Layer Default

Transform

Position	X -9999	Y -9999	Z -9999
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Network Object (Script)

Script	NetworkObject
Is Networked	✓

Ready Lobby Network (Script)

Script	ReadyLobbyNetwork
Game Scene Configurations	LobbyNetwork (GameSceneConfigurations)

Game Scene Configurations (Script)

Script	GameSceneConfigurations
Physics Mode	Physics 3D
Scenes	
Size	1
Element 0	Game

Network Observer (Script)

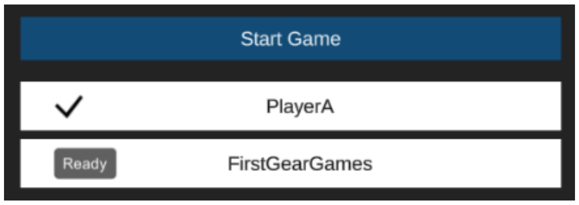
Script	NetworkObserver
Override Type	Ignore Manager
Observer Conditions	
Size	0

The LobbyNetwork script is responsible for managing rooms and players. You can inherit from LobbyNetwork and override actions, or even add additional functionality.

Lobby and Worlds utilizes inheritance to allow modifications without requiring you to modify the core of Lobby and Worlds.

ReadyLobbyNetwork is an example of inheriting from LobbyNetwork. The ReadyLobbyNetwork adds the new functionality of requiring all players to ready-up before the room can start.

The lobby UI prefabs may also be replaced without modifying the core of Lobby and Worlds. A custom prefab was made for the ‘ready-up’ functionality. The prefab is named ReadyMemberEntry and is used to show which players are ready before starting a match, as well toggling the ready state.



If you were to open the ReadyMemberEntry prefab you will find a script named ReadyMemberEntry added as a component. Similar to ReadyLobbyNetwork, this script inherits from the default member entry, MemberEntry, making custom changes to the lobby easy.

You will assign the majority of your lobby prefabs in the Lobby scene, under Lobby -> Canvases. The full path to the entry prefab reference is Lobby -> Canvases -> JoinCreateRoomCanvas -> CurrentRoomMenu, I strongly encourage you to make a duplicate of the Lobby scene before making any changes. This will prevent you from losing work should you update Lobby and Worlds.

When it comes time for you to customize the lobby to your needs I encourage you to inspect overrides and public events within the LobbyNetwork and MemberEntry scripts to see what is available to you. Should there be something you would need exposed to get your project going, let me know on my Discord so we can get it in there.

GAME DESIGN

Lobby and Worlds is designed to separate your game logic from the lobby logic. In other words, you can develop your game and it’s scenes normally, without worrying about the lobby aspect of it. Included is an example game, King of the Hill. Open the example scene, Game.

Lobby and Worlds uses scene stacking for it’s rooms. Scene stacking is when the same scene is loaded multiple times on the server. The main hurdle when stacking scenes is how you use physics. When scenes are stacked each scene gets it’s own PhysicsScene. You can get which PhysicsScene belongs to a scene by using scene.GetPhysicsScene; there is a 2D reference as well. Collisions are automatically separated between physics scenes, but you must specify which PhysicsScene to make cast on, such as a box cast. For example, gameObject.scene.GetPhysicsScene().Raycast(). For more information on on this check out Unity’s documentation, keeping in mind there is a 2D version as well.
LINK: https://docs.unity3d.com/ScriptReference/PhysicsScene.html

There is one more gotcha with physics scenes, but it’s relatively small. In the Game scene expand the Scene Design object, and select the SimulatePhysics object. On it is a script named SimulatePhysics, open this script. When scene stacking you must also manually simulate the physics step for that scene; SimulatePhysics shows an easy way to accomplish this.

Within the same Game scene there is another useful script which shows how you may get your game started. Select the GameplayManager object in the scene, and then open the GameplayManager script on it. Since the example game is small, this script controls the entire game. For your project, you will likely have a variety of scripts managing your game. More importantly, the GameplayManager script shows how you may start your game, and spawn players.

Ignoring the game logic, there are a few things worth noting in the GameplayManager script. I have a public void called FirstInitialize which links the lobby and room details with the game instance for that scene. This method is called from my custom LobbyNetwork, which as we discussed earlier you can make without modifying the core of Lobby and Worlds. When initialized I also subscribe to the LobbyNetwork OnClientStarted, and OnClientLeftRoom events. All events have descriptions, it’s important you review all overrides and events available to you within the LobbyNetwork for a full understanding of what abilities you have.

When LobbyNetwork_OnclientStarted is called the GameplayManager knows a client has started a game. The manager then compares if the room which the client started, matches the room for it’s instance. This is where previously initializing with the RoomDetails comes in. After validating that the client did join the same instance, an object is spawned for that player. Written out this may seem like a complicated process, but when reviewing the GameplayManager script you will find it’s actually only a few lines of code.

There is a very important thing to keep in mind when spawning objects over the network within Lobby and Worlds. Visiting the SpawnPlayer method within the GameplayManager script you will notice that after instantiating the player, it’s moved to the correct scene before network spawning. This is important as spawning the player in the correct game scene will result in proper visibility of other game elements.

Don’t forget to leave the room if you wish to move players back to the lobby. In the GameplayManager I call _lobbyNetwork.TryLeaveRoom(playerNob). This will remove the player from the room and automatically load them back into the lobby.

Thank you for taking the time to read this, hope you enjoy Lobby and Worlds!