Lab 1

For this module, we are going to create a simple network game with some networking features using Netcode from Unity. Unity Netcode is a mid-level networking library built for the Unity game engine to abstract networking. Netcode includes MLAPI and it is just recently released in. We are going to use this since it is a good compromise as a mid level API and going forward Netcode will be likely to be used and maintained.

# Installing Netcode

At the time of writing, Netcode can be installed through Unity package installer. The current official guide can be found in https://docs-multiplayer.unity3d.com/netcode/current/about/index.html.

**When you are using the lab (which has network restrictions), please download the template project which already has Netcode in it** hence you can just load the template project and skip to step 3.

To install it at home on your own machine:

# Setting Up Basic Network Project

1. Open the Unity Hub.
2. Click New.
3. Select type ‘2D’
4. Select the location to save the project.
5. Open the Unity Package Manager by navigating to **Window** > **Package Manager** on Unity’s main menu.
6. Click  in the status bar and select **Add package from name**
7. You could install Netcode for Gameobjects by either:
   1. Enter the Git URL by clicking the Copy option in that codeblock and paste it in the Package Manager.

com.unity.netcode.gameobjects

We recommend that you use the **Copy** function in the code block above to copy the URL as other methods may result in errors. Just hover and click Add

* 1. You can also look Netcode for Gameobjects which is on the list and then click Install button which is on the bottom right of the Package Manager window

1. The package installs and appears as Netcode for GameObjects under Packages in the Package Manager window

# Creating Network Manager and selecting the Transport

In this section we will add a Network Manager and add a Transport to our project.

1. Right click in the Hierarchy tab of the Main Unity Window.
2. Select **Create Empty**.
3. Rename the GameObject **NetworkManager**.

A screenshot of a computer

Description automatically generated

1. Select **NetworkManager** object and Click **Add Component** called **NetworkManager** in the Inspector tab.

A screenshot of a computer

Description automatically generated

1. Inside the **NetworkManager** Component tab, locate the NetworkTransport field.
2. Click "Select Transport".

A screenshot of a computer

Description automatically generated

1. Select UnityTransport.

A screenshot of a computer

Description automatically generated

1. Save your scene.

# Creating an object to spawn for each connected player

This section adds in a player object and spawns it for each connected player.

1. Crate a player. At this point we will just use a box as the player so create a box 2d sprite and lets name it **Player.**
2. You can add Rigidbody2D and BoxCollider2D components to the **Player** object
3. Drag and drop the **Player** object to the assets folder to automatically create the **Player** prefab
4. Select **NetworkManager**.
5. Inside the **NetworkManager** Component tab, locate the NetworkPrefabs field.
6. Click + to create a slot.
7. Drag this player prefab from above into the new empty slot. This indicates that this prefab should be synced
8. On the **Player Prefab**, drag and drop the player prefab **Player** so when a client connects to the game, automatically spawn this prefab as the character for the connecting client. If you do not have the default selected for any prefab the game will crash on client connect. You should also check the position of where the player prefab is to make sure it is not off the level
9. Save the Scene and add the Scene to the Build Settings
10. You can test the game by running the game in the editor. When you run the game, the NetworkManager will be under DontDestroyOnLoad. Click on the NetworkManager object and on the inspector you can see there is button “Start Host”

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

When you click the Start Host button you should see the player spawned into the game environment.

# Adding Network Game Menu Scripts

This section will add some scripts to the project which will contain the new features we will be covering in this module.

1. On the **Scripts** folder, right click and create a script called MultiPlayerManager
2. Create an empty GameObject rename it MultiplayerManager.
3. Add the MultiPlayerManager script as a component.
4. Open the MultiPlayerManager.cs script.
5. Edit the MultiPlayerManager.cs script to match the following.

using Unity.Netcode;

using UnityEngine;

public class MultiPlayerManager : MonoBehaviour

{

void OnGUI()

{

GUILayout.BeginArea(new Rect(10, 10, 300, 300));

if (!NetworkManager.Singleton.IsClient && !NetworkManager.Singleton.IsServer)

{

StartButtons();

}

else

{

StatusLabels();

}

GUILayout.EndArea();

}

static void StartButtons()

{

if (GUILayout.Button("Host")) NetworkManager.Singleton.StartHost();

if (GUILayout.Button("Client")) NetworkManager.Singleton.StartClient();

if (GUILayout.Button("Server")) NetworkManager.Singleton.StartServer();

}

static void StatusLabels()

{

var mode = NetworkManager.Singleton.IsHost ?

"Host" : NetworkManager.Singleton.IsServer ? "Server" : "Client";

GUILayout.Label("Transport: " +

NetworkManager.Singleton.NetworkConfig.NetworkTransport.GetType().Name);

GUILayout.Label("Mode: " + mode);

}

}

Notes:

You can also connect to the server or host in another computer in the network by changing the IP address in the Connect Address of the UnityTransport component in the NetworkManager.

You could try to make a menu which the player can specify the parameters for the network connection especially for the IP Address on the ConnectionData on the UnityTransport component