CapsuleMove

9/19/2023

This is a proof of concept project combining the multiplayer movement of Netcode for Game Objects with Sam's InputListener.cs for use with Quest2 headset. It can be compiled for Windows or Oculus. Button or key A starts it as a Host, while button or key B starts it as a client. Then keyboard keys A,S,D and W move the capsule in Windows, and the right joystick moves it in Quest2.

From <https://medium.com/eincode/getting-started-with-multiplayer-player-movement-f9f7f6a4217>

Make a new project, make it run Android

Project Settings, Player, made Company name EastRidges

Project Settings, XR Plug-in Management, check Oculus (Initialize XR on Startup)

Make sure **Netcode for Game Objects** is installed from Package Manager, Add package by name, com.unity.netcode.gameobjects Add

Also add Package Manager, **Multiplayer Tools**

(If not there, then:

* Package Manager → Add package from git URL
* Paste "[https://github.com/Unity-Technologies/com.unity.multiplayer.samples.coop.git?path=/Packages/com.unity.multiplayer.samples.coop#main](https://github.com/Unity-Technologies/com.unity.multiplayer.samples.coop.git?path=%2FPackages%2Fcom.unity.multiplayer.samples.coop#main)"
* Hit Add.

)

Make a main scene

Make a Plane called Floor

Build and run

(saw floor, immersive view – got here)

Try changing main camera to XRRig. If can't we may have to import Package Manager-> My Assets -> Oculus Integration - but that's a very big deal . -> It did change to XRRig, which has camera offset and Main Camera inside it; status line still says Checking camera components.

Create an empty game object, call it **Network Manager**, (reset the transform?) and in Inspector, Add Component->Network Manager. (might be under Netcode?)

Select the Transport Layer (Unity Transport) and set ip address to local machine IP

Check box Allow Remote Connections

Remember to open the port (i.e. 7777) in your computer's firewall

(build and run works)

Double check you have Package Manager, Multiplayer tools (which is needed needed to have ClientTransportLayer instead of the standard Transport Layer –

Make sure **Oculus XR Plugin** is loaded in Package Manager too

(they are)

Make empty object, name it **InputManager**

Add a tag, “InputReaderTag”

Add component, **Network Object** (not sure if required but it worked on CodeMonkey trial)

add component, script, Sam's InputReader.cs, but edited InputReader.cs to say using Unity.Netcode and made the class inherit from NetworkBehaviour instead of MonoBehaviour

Make a capsule in the scene, call it **Player**, reset transform, offset the Y by 1 (the offset doesn't seem to work in spawned objects which might be why CodeMonkey made an empty object called Player and added a capsule to it - no matter)

Make a **Prefabs** folder and drag and drop the Player game object from the scene to that folder to make it a prefab. Now can delete the Player game object from the scene

In Player Inspector, add Component, Network Object

Make **PlayerNetwork.cs** script for Player

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.EventSystems;

using Unity.Netcode;

using System;

public class PlayerNetwork : NetworkBehaviour

{

public InputReader Inputs;

internal bool InputReaderGameObjectExists; //false

internal bool InputReaderComponentExists; // false

private void Start()

{

}

private void Update()

{

if (!IsOwner)

{

return;

}

//assign Inputs again even though was done in the Start() method

GameObject go = GameObject.FindGameObjectWithTag("InputReaderTag");

if (go != null)

{

InputReaderGameObjectExists = true;

Inputs = go.GetComponent<InputReader>();

if (Inputs != null)

{

InputReaderComponentExists = true;

}

}

Vector3 moveDir = new Vector3();

float moveSpeed = 3f;

float vertSpeed = 0;

//if (InputReaderComponentExists) //NOTE This returns true and vert speed is up if not commented out...

// vertSpeed = 0.5f;

//else

// vertSpeed = -0.5f;

if ((go != null) && (InputReaderComponentExists) && (Inputs.RightControllerFound))

//if(false)

{

//this should be true for the Quest2 headset

Vector2 joystick = Inputs.rightJoystick;

moveDir = new Vector3(joystick.x, vertSpeed, joystick.y);

//if (Inputs.ButtonA)

// moveDir.x = +1f;

//else if (Inputs.ButtonB)

// moveDir.x = -1f;

}

else

{

//this should work for the Windows keyboard

if (Input.GetKey(KeyCode.W)) moveDir.z = +1f;

if (Input.GetKey(KeyCode.S)) moveDir.z = -1f;

if (Input.GetKey(KeyCode.A)) moveDir.x = -1f;

if (Input.GetKey(KeyCode.D)) moveDir.x = +1f;

//the following code works for debugging when now input is found...

////////just make some motion to show it's alive

//////moveSpeed = 1; //slower

//////int cycle = DateTime.Now.Second;

//////if (cycle % 4 == 0) //modulus

//////{

////// moveDir.x = +1f;

//////}

//////else if (cycle % 4 == 1)

//////{

////// moveDir.z = +1f;

//////}

//////else if (cycle % 4 == 2)

//////{

////// moveDir.x = -1f;

//////}

//////else

//////{

////// moveDir.z = -1f;

//////}

}

moveDir.y = vertSpeed;

transform.position += moveDir \* moveSpeed \* Time.deltaTime;

}

}

Drag Player from the Prefabs folder into the Player Prefab field in NetworkManager in Inspector

Add component **Client Network Transform** to the Player prefab, and check only X, Y and Z Positions to be synced. To get Client Network Transform option to add, go to Window, Package Manager, add from github:

<https://github.com/Unity-Technologies/com.unity.multiplayer.samples.coop.git?path=/Packages/com.unity.multiplayer.samples.coop#main>

then you can Add Component and search for Client Network Transform.

Let's omit making a canvas, but just add a script to the Floor object calling it **NetworkManagerUI.cs**

using System.Collections;

using System.Collections.Generic;

using Unity.Netcode;

using UnityEngine;

using UnityEngine.UI;

public class NetworkManagerUI : MonoBehaviour

{

//do I need to put [SerializeField] before public InputReader Inputs??

public InputReader Inputs;

private bool hasStartedNetworkManager; //false

private void Awake()

{

}

/// <summary>

/// called every frame

/// </summary>

private void Update()

{

if (!hasStartedNetworkManager)

{ }

if (Inputs.ButtonA)

{

NetworkManager.Singleton.StartHost();

hasStartedNetworkManager = true;

}

else if (Inputs.ButtonB)

{

NetworkManager.Singleton.StartClient();

hasStartedNetworkManager = true;

}

} //from if hasn't already started network manager host or client

}

(consider putting [SerializeField] before declaring Inputs? -> no)

Drag and drop InputManager to the Inputs field on Floor's Inspector

Build and Run and press the A button to test! Does it move with the joystick? ( by the way, the capsule is not raised above the floor, but bisected – that’s ok.)

(If building for Quest2, make sure it is plugged in and build object Quest2 is selected in Build Settings. Once you run it once in the headset you can unplug it and run it in the headset as a client by clicking Search Apps, click in the search text field which brings up option for categories- choose "Unknown Sources" and it CapsuleMove should be there)

(….ignore this….

I noticed I hadn’t made a tag in InputManager called InputReaderTag yet, so did that and now it works!.

Also I had to check the AllowRemote Connections in order to connect headset to the host on the computer. It all works, the proof of concept is successful.

Note I didn’t have to put [SerializeField] before Inputs in NetworkManagerUI

Didn’t have to add Player to Network Prefabs (also not done in CodeMonkey)

I didn’t have to use OculusIntegration

….from ignore this….)