โค้ดใช้ทดสอบระบบออนไลน์

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using Unity.Netcode;

using UnityEngine.SceneManagement;

public class test\_scene\_name : NetworkBehaviour

{

private TMP\_Text currenSceneText;

private bool canSwitchScene = true;

private void Start()

{

currenSceneText = GetComponent<TMP\_Text>();

NetworkManager.Singleton.Shutdown();

currenSceneText.text = SceneManager.GetActiveScene().name;

}

private void Update()

{

if ((Input.GetKeyDown(KeyCode.Alpha1) || Input.GetKeyDown(KeyCode.Keypad1)) && canSwitchScene == true)

{

if (SceneManager.GetActiveScene().name == ("1 - netcode intro")) { return; }

SceneManager.LoadScene("1 - netcode intro");

canSwitchScene = false;

}

if ((Input.GetKeyDown(KeyCode.Alpha2) || Input.GetKeyDown(KeyCode.Keypad2)) && canSwitchScene == true)

{

if (SceneManager.GetActiveScene().name == ("2 - connection approval")) { return; }

SceneManager.LoadScene("2 - connection approval");

canSwitchScene = false;

}

if (Input.GetKeyDown(KeyCode.R) && canSwitchScene == true)

{

SceneManager.LoadScene(SceneManager.GetActiveScene().name);

}

}

}

โค้ดควบคุมระบบออนไลน์

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using Unity.Netcode;

public class MainGameManager : NetworkBehaviour

{

//MainGameManger is attached to NetworkManger, its duty is to control the online system.

//created in week4 - netcode introduction(project setup, RPC-method, lauching as host/client/server)

void OnGUI() //control functional buttons created by script

{

GUILayout.BeginArea(new Rect(10, 10, 300, 300)); //create area for buttons

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

//is current player not the client? and...

//is current player not the server?

{

StartButtons(); //if true, show the start button, everytime enter the game,

//it will show buttons that ask: will you start as host, client, or server?

}

else

{

StatusLabels(); //after choose one of three, it will display that you are

//a host, client, or server

SubmitNewPosition(); //also showing move or request to move; depend on what you are

}

GUILayout.EndArea(); //close the division area of buttons

}

static void StartButtons() //the buttons ask you to choose to be a host, client, or server

{

if (GUILayout.Button("Host")) NetworkManager.Singleton.StartHost(); //start game as the host

if (GUILayout.Button("Client")) NetworkManager.Singleton.StartClient(); //...as the client

if (GUILayout.Button("Server")) NetworkManager.Singleton.StartServer(); //...as the server

}

static void StatusLabels() //the status shown that you are a host, client, or server after choosing one of them

{

var mode = NetworkManager.Singleton.IsHost ? //the mode you chose (host/client/server)

//first it will check that you are a host or not.

"Host" : NetworkManager.Singleton.IsServer ? "Server" : "Client"; //if not, are you a server? if not you are a client.

GUILayout.Label("Transport: " +

NetworkManager.Singleton.NetworkConfig.NetworkTransport.GetType().Name); //the type of transport you're using, not to bother really.

GUILayout.Label("Mode " + mode); //display the mode from condition in previous lines

}

static void SubmitNewPosition() //a button that moves the player to new position when pressed

{

if (GUILayout.Button(NetworkManager.Singleton.IsServer ? "Move" : "Request position change")) //draw a button, if you're server; the butto says move,

//if not; request postion change

{

var playerObject = NetworkManager.Singleton.SpawnManager.GetLocalPlayerObject(); //refer the player object of that local

var player = playerObject.GetComponent<MainPlayer>(); //reference script MainPlayer

player.Move(); //move the player using Move method in MainPlayer

}

}

}

โค้ดควบคุมผู้เล่น

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using Unity.Netcode;

public class MainPlayer : NetworkBehaviour

{

//the MainPlayer script is attached to player object

//the player object use ClientNetworkTransform, just know that it's better than typical NetworkTransform

//created in week4 - netcode introduction(project setup, RPC-method, lauching as host/client/server)

public NetworkVariable<Vector3> Position = new NetworkVariable<Vector3>(); //the position saved after random, and will be used later.

public float speed = 15f;

public float rotateSpeed = 30f;

Rigidbody rigidbody;

private void Start()

{

rigidbody = this.gameObject.GetComponent<Rigidbody>();

}

private void FixedUpdate()

{

float translation = Input.GetAxis("Vertical") \* speed;

float rotation = Input.GetAxis("Horizontal") \* rotateSpeed;

translation \*= Time.deltaTime;

Quaternion turn = Quaternion.Euler(0, rotation, 0);

rigidbody.MovePosition(rigidbody.position + this.transform.forward \* translation);

rigidbody.MoveRotation(rigidbody.rotation \* turn);

}

public override void OnNetworkSpawn() //overriding OnNetworkSpawn to execute Move() after conneting to network

{

Move(); //move the player to random pose

}

public void Move() //already explained that

{

if (NetworkManager.Singleton.IsServer) //are you the server? if yes, random postion and move you(server) to that place immediately

{

var randomPosition = GetRandomPostionOnPlanet(); //random the postion

transform.position = randomPosition; //change postion

Position.Value = randomPosition; //store the position in vector3

}

else

{

SubmitPositionRequestServerRpc(); //if you are the client, ask the server to move you to new position

}

}

[ServerRpc] //requesting the server

void SubmitPositionRequestServerRpc(ServerRpcParams rpcParams = default) //ServerRpcParams is an optional attribute for runtime, nothing to bother really

{

Position.Value = GetRandomPostionOnPlanet(); //store the randomed position

transform.position = Position.Value; //keep the position at last randomed Vector3

}

static Vector3 GetRandomPostionOnPlanet() //random position method and return as Vector3

{

return new Vector3(Random.Range(3f, -3f), 1f, Random.Range(3f, -3f)); //random the pos in x-z coordinate

}

private void Update() //Update, nothing special

{

//transform.position = Position.Value; //keep the position at last randomed Vector3

//\*\*tihs was once used in Update and moved to submit pos-change-request, but had been removed to make player controlling functional

}

}

โค้ด login เกม

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using Unity.Netcode;

using System.Text;

using UnityEngine.UI;

using System;

public class LogInManager : NetworkBehaviour

{

public Text playerNameInputField; //the text get from inputfield

public GameObject loginPanel;

public GameObject leaveButton;

//private void Start()

//{

// NetworkManager.Singleton.OnServerStarted += HandleServerStarted;

// NetworkManager.Singleton.OnClientConnectedCallback += HanldeClientConnected;

// NetworkManager.Singleton.OnClientDisconnectCallback += HandleClientDisconnect;

// loginPanel.SetActive(true);

// leaveButton.SetActive(false);

//}

//private void OnDestroy()

//{

// if (NetworkManager.Singleton == null) { return; }

// NetworkManager.Singleton.OnServerStarted -= HandleServerStarted;

// NetworkManager.Singleton.OnClientConnectedCallback -= HanldeClientConnected;

// NetworkManager.Singleton.OnClientDisconnectCallback -= HandleClientDisconnect;

//}

//private void HandleServerStarted()

//{

// //throw new NotImplementedException();

//}

//private void HanldeClientConnected(ulong clientId)

//{

// print(playerNameInputField.text + "client id: " + clientId);

// if (clientId == NetworkManager.Singleton.LocalClientId)

// {

// loginPanel.SetActive(false);

// leaveButton.SetActive(true);

// }

// //throw new NotImplementedException();

//}

//private void HandleClientDisconnect(ulong obj)

//{

// loginPanel.SetActive(true);

// leaveButton.SetActive(false);

//}

public void Host()

{

NetworkManager.Singleton.ConnectionApprovalCallback += ApprovalCheck; //subscript ConnectionApprovalCallback to approvealCheck

NetworkManager.Singleton.StartHost(); //start Host function

}

private void ApprovalCheck(byte[] connectionData, ulong clientId,

NetworkManager.ConnectionApprovedDelegate callback)

{

//throw new NotImplementedException(); //show red line debug that the method is functional;

string playerName = Encoding.ASCII.GetString(connectionData); //encode and store the name of previous player joining network.

bool approveConnection = playerName != playerNameInputField.text; //ApproveConnetion indicate that you can join the game or not

//this where we will put logic argument.

//From this example, if player existed player is not the same as

//the new player connected in.

//so true, and new player can join the game.

//In the other hand, unapproved to join.

if(playerNameInputField.text == "no name")

{

print("not approved");

approveConnection = false;

}

print(playerName + " " + playerNameInputField.text);

bool createPlayerObject = true; //are you allowed to create your player object

//bool approveConnection = true; //this is for true test

callback(createPlayerObject, /\*playerPrefabHash\*/ null, approveConnection, /\*positionSpawnAt\*/ null, /\*rotationSpawnWith\*/ null);

//this callback is crucial part of spawning player object, where to start, and are you allowed to join or not

//createPlayerObject (bool): will player object(character) be created?

//playerPrefabHash (uint): specific prefab index to be created for that player; while set to null, it will use default prefab assigned in MainGameManager which index is 0

//approval connection (bool): are you approved to join the network? pretty plain really.

//positionSpawnAt (Vector3): where to spawn player.

//rotationSpawnWith (Vector3): rotation of player.

//\*\*\*fyi: uint is alt. type of int which cannot be negative value.

}

public void Client()

{

NetworkManager.Singleton.NetworkConfig.ConnectionData =

Encoding.ASCII.GetBytes(playerNameInputField.text); //encoding and store player's name value for later checking aprroval

NetworkManager.Singleton.StartClient(); //start the client

}

}