Unity Multiplayer Chat Server Part 2 – Unity part 2

1. Create a new script in the client’s folder we created under the scripts folder made in the unity project. Call this “Client.cs”

Here is the code to make, after following part one.

using UnityEngine;

using System.Collections;

using System.Net.Sockets;

using System.IO;

public class Client : MonoBehaviour

{

private bool socketReady;

private TcpClient socket;

private NetworkStream stream;

private StreamWriter writer;

private StreamReader reader;

public void ConnectToServer()

{

// if already connected, ignore this function

if(socketReady)

return;

// Default host / port values

string host = “127.0.0.1”;

int port = 6321;

// Overwrite default host / port values, if there is something in those boxes.

string h;

int p;

h – GameObject.Find(“HostInput”).GetComponent<InputField>().text;

if (h ! = “”)

host = h;

int.TryParse(GameObject.Find(“PortInput”).GetComponent<InputField>().text, out p);

if (p != 0)

port = p;

// Create the socket

try

{

socket = new TcpClient(host, port);

stream = socket.GetStream();

writer = new StreamWriter(stream);

reader = new StreamReader(stream);

socketReady = true;

}

catch(Exception e)

{

Debug.Log(“Socket error : “ + e.Message);

}

}

private void Update()

{

if(socketReady)

{

if(stream.DataAvailable)

{

string data = reader.ReadLine();

if(data != null)

OnIncomingData(data);

}

}

}

private void OnIncomingData(string data)

{

Debug.Log(“Server : “ + data);

}

}

Now its time to go back to create a function that sends messages. Update your server script under folder server.

Update by copying and pasting this script.

using UnityEngine;

using System.Collections;

using System.Net.Sockets;

using System.Collections.Generic;

using System;

using System.Net;

public class Server : MonoBehaviour

{

private List<ServerClient> clients;

private List<ServerClient> disconnectList;

public int port = 6321;

private tcpListener server;

private bool serverStarted;

private void Start()

{

clients = new List<ServerClient>();

disconnectList = new List<ServerClient>();

try

{

server = new TcpListener(IPAddress.any, port);

server.Start();

StartListening();

serverStarted = true;

Debug.Log(“Server has been started on port “ + port.ToString());

}

catch(Exception e)

{

Debug.Log(“Socket error: “ + e.Message);

}

private void Update()

{

if (!serverStarted)

return;

foreach (ServerClient c in clients)

{

// is the client still connected?

if(!IsConnected(c.tcp))

{

c.tcp.Close();

disconnectList.Add;

continue;

}

// check for message from client

else

{

NetworkStream s = c.tcp.GetStream();

if(s.DataAvailable)

{

StreamReader reader = new StreamReader(s, true);

string data = reader.ReadLine();

if(data != null)

OnIncomingData(c, data);

}

}

}

}

private void StartListening()

{

server.BeginAcceptTcpClient(AcceptTcpClient, server);

}

private bool IsConnected(TcpClient c)

{

try

{

if (c != null && c.Client != null && c.Client.Connected)

{

if (c.Client.Poll(0, SelectMode.SelectRead))

{

return !(c.Client.Recieve(new byte[1], SocketFlags.Peek) == 0);

}

return true;

}

else

return false;

}

catch

{

return false;

}

}

private void AcceptTcpClient(IAsyncResult ar)

{

TcpListener listener = (TcpListener)ar.AsyncState;

clients.Add(new ServerClient(listener.EndAcceptTcpClient(ar)));

StartListening();

// send a message to everyone, say someone has connected

Broadcast(clients[clients.Count – 1].clientName + “has connected”, clients);

}

private void OnIncomingData(ServerClient c, string data)

{

Debug.Log(c.clientName + “ has sent the following message! : “ + data);

}

private void Broadcast(string data, List<ServerClient> cl)

{

foreach(ServerClient c in cl)

{

try

{

StreamWriter writer = new StreamWriter(c.tcp.GetStream());

writer.writeLine(data);

writer.flush();

}

catch(Exception e)

{

Debug.Log(“Write error : “ + e.Message + “ : to client “ + c.clientName);

}

}

}

}

public class ServerClient

{

public TcpClient tcp;

public string clientname;

public ServerClient(TcpClient clientSocket)

{

clientName = “Guest”;

tcp = clientSocket;

}

Under your Server object(game object created in tutorial #1), add the client script, and on the button add the Server game object to the OnClick Event (over to the right, down the inspector a little. Go to the right scroll option on the On Click function, and go to the bottom function, and search for ConnectToServer();. Add Server.cs to the ServerGameObject

1. Go to build setting, (under file on the top right), a new box appears, click on Player Settings at the bottom of this box, under Resoulution and Presentations under the inspector, toggle “Run In Background”
2. Add a panel inside of your UI Panel for chat, Inside the scrollview, Create UI Panel, and under the layout make it “stretch” (toggling the box on the left before the input fields, and make sure it is stretching inside the chat window. Create a UI Text inside this panel. This is where the chatting will occur inside your client. Add to a new file in your unity project called “Prefab”. Drag the prefab into the chat window that fits inside your chat panel. Add a vertical layout group(under the inspector via search) and expand on width. Now delete all the messages (if you laid them out right) and its times to update our Client script,

Delete and copy and replace this script into the Clients.cs

1. Create a new script in the client’s folder we created under the scripts folder made in the unity project. Call this “Client.cs”

Here is the code to make, after following part one.

using UnityEngine;

using System.Collections;

using System.Net.Sockets;

using System.IO;

public class Client : MonoBehaviour

{

public GameObject chatContainer;

public GameObject messagePrefab;

private bool socketReady;

private TcpClient socket;

private NetworkStream stream;

private StreamWriter writer;

private StreamReader reader;

public void ConnectToServer()

{

// if already connected, ignore this function

if(socketReady)

return;

// Default host / port values

string host = “127.0.0.1”;

int port = 6321;

// Overwrite default host / port values, if there is something in those boxes.

string h;

int p;

h – GameObject.Find(“HostInput”).GetComponent<InputField>().text;

if (h ! = “”)

host = h;

int.TryParse(GameObject.Find(“PortInput”).GetComponent<InputField>().text, out p);

if (p != 0)

port = p;

// Create the socket

try

{

socket = new TcpClient(host, port);

stream = socket.GetStream();

writer = new StreamWriter(stream);

reader = new StreamReader(stream);

socketReady = true;

}

catch(Exception e)

{

Debug.Log(“Socket error : “ + e.Message);

}

}

private void Update()

{

if(socketReady)

{

if(stream.DataAvailable)

{

string data = reader.ReadLine();

if(data != null)

OnIncomingData(data);

}

}

}

private void OnIncomingData(string data)

{

Debug.Log(“Server : “ + data);

GameObject go = Instantiate(messagePrefab, chatContainer.transform) as GameObject;

go.GetComponentinChildren<Text>().text + data;

}

}

Head back inside the game, and inside Server gameobject, under client Add ChatWindow (the panel inside the scrollview), and add the message Prefab created.

1. Create a new panel (to put for sending messages) and create an InputField and Send Button beside it, name the InputField to “SendInput”

Update the Clients script to include this function

private void Send(string data)

{

if (!socketReady)

return;

writer.writeLine(data);

writer.flush();

}

public void OnSendButton()

{

string message = GameObject.find(“SendInput”).GetComponent<InputField().text;

Send(message);

}

Attach OnSendButton to the send button’s onclick event. Client Function, OnSendButton

On the Server.cs, change OnIncomingData(ServerClient c, string data)’s function to

This

Broadcast(data, clients);

1. Remember to change the server on and off for the computers that you want to host the server.
2. Change “guest” name for different pc’s to tailored to the person’s name

Change under // send a message to everyone say somone has connecter, under AcceptTcpClient() function to this

Broadcast(“%NAME”, new List<ServerClient>() { clients[clients.count – 1)});

And change OnIncomingData(string data)

Just underneath the function put this (in Clients.cs)

if (data.Contains(“%NAME”))

{

c.clientName = data.Split(‘|’)[1];

Broadcast(c.clientName + “has connected”, clients);

return;

}

Don’t forget to add this to the top of client after class. Call it this

public string clientName;

So you can change the name in the inspector.

Under server.cs, after the foreach loop in update() add this

for (int I = 0; I < disconnectList.Count – 1; i++)

{

BroadCast(disconnectList[i].clientName + “has disconnected”, clients);

clients.Remove(disconnectList[i]);

disconnectList.RemovesAt(i);

}

1. Create a new function in Clients.cs

Two new functions will be created at the bottom, for closing the client.

private void CloseSocket()

{

if (!socketReady)

return;

writer.Close();

reader.Close();

socket.Close();

socketReady = false;

}

private void OnApplicationQuit()

{

CloseSocket();

}

private void OnDisable()

{

CloseSocket();

}