6.0

BASIC

WIFI

LOCAL

MULTIPLAYER

Connect everyone on the network

learn step by step to develop your mmo!

SUMMARY

QUICK START

HOW TO DEVELOP

SAMPLES

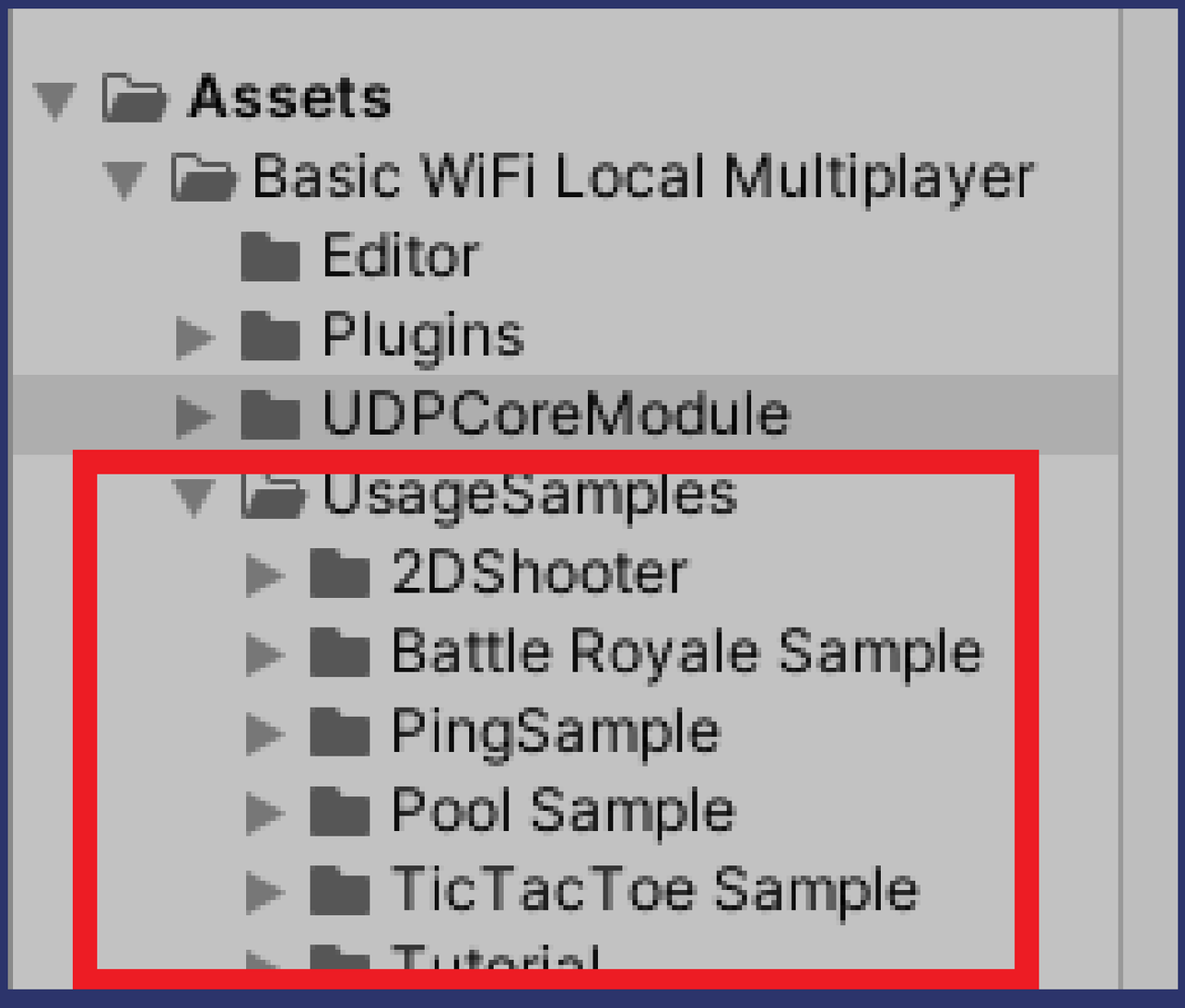
CONTACT & SUPPORT

QUICK START

SETUP SAMPLES

This guide will show you how to create lots of amazing applications with few code
lines!

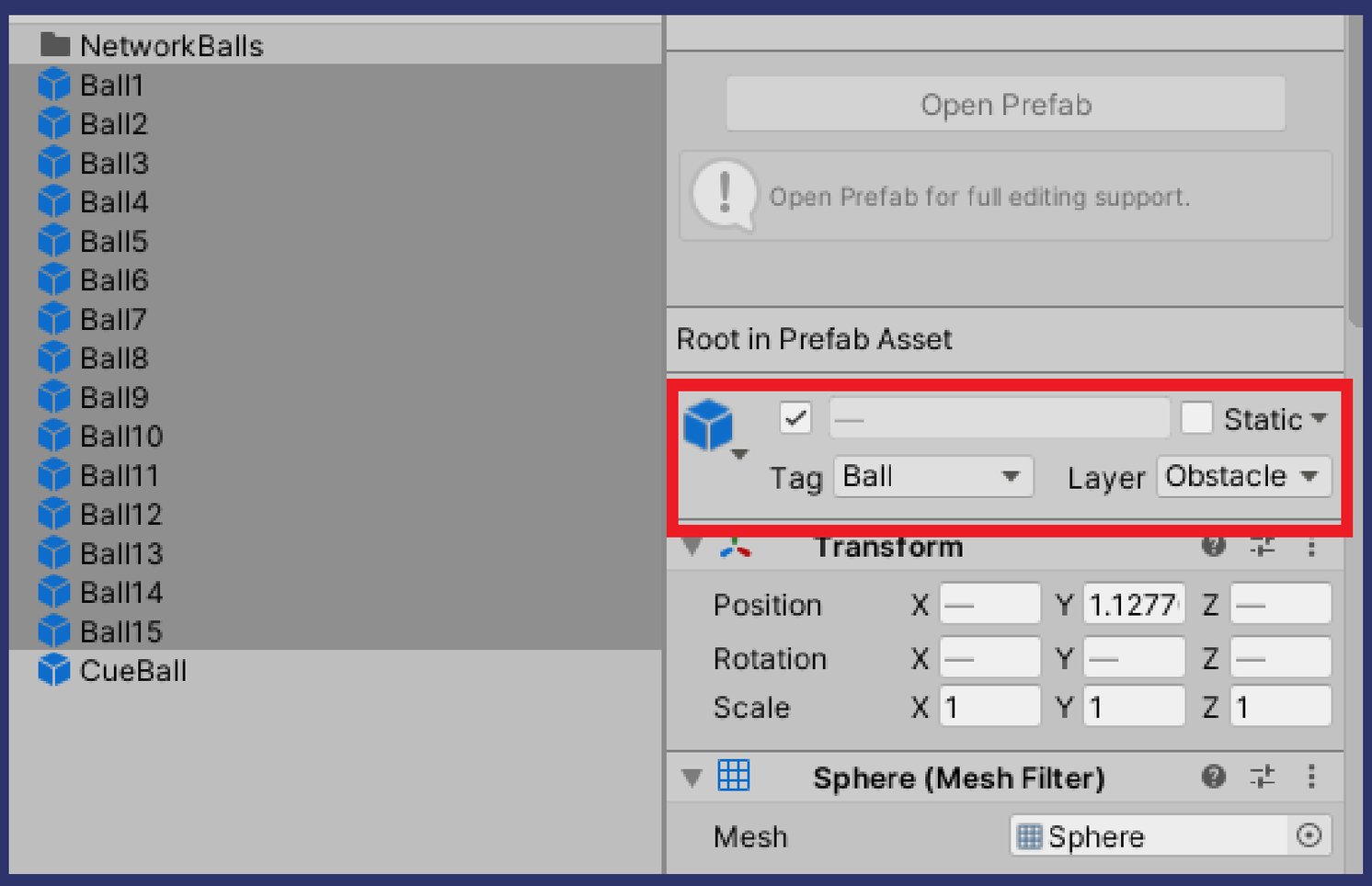
In the project folder, choose an sample of your preference and select the main scene of the sample:

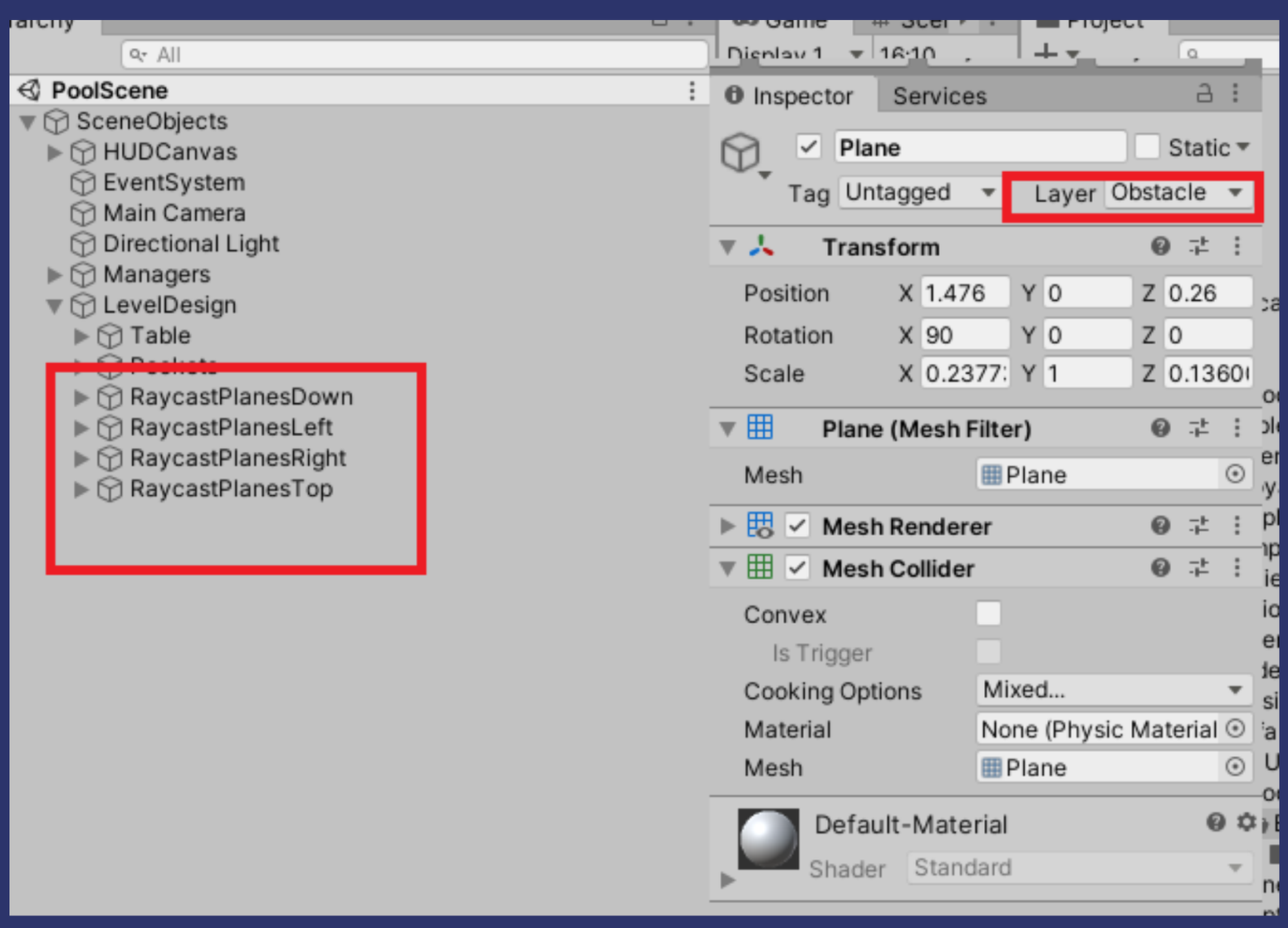


for Pool samples it is necessary that you create a layer called "***Obstacle***" , then you must add this to the following objects:

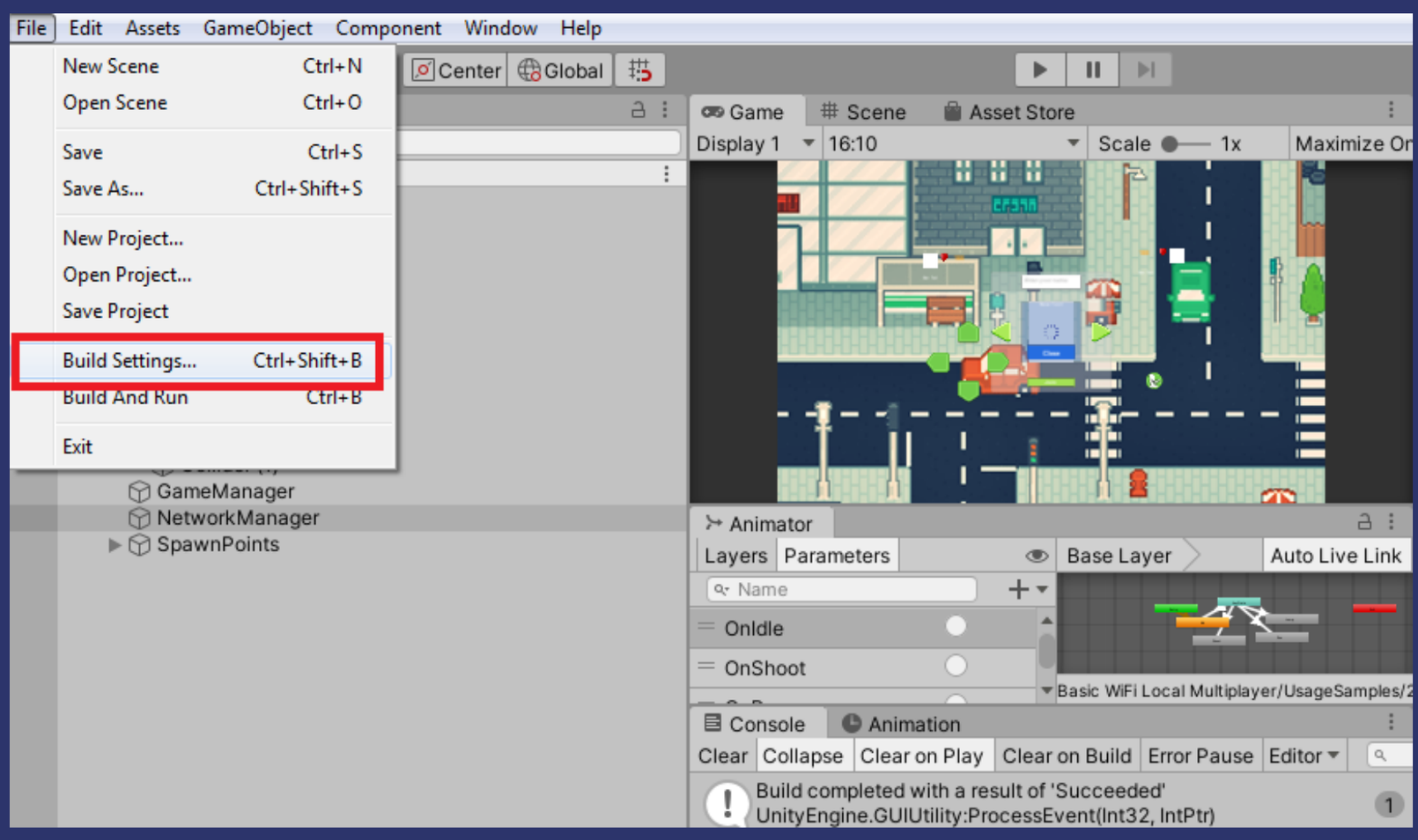
Pool Sample

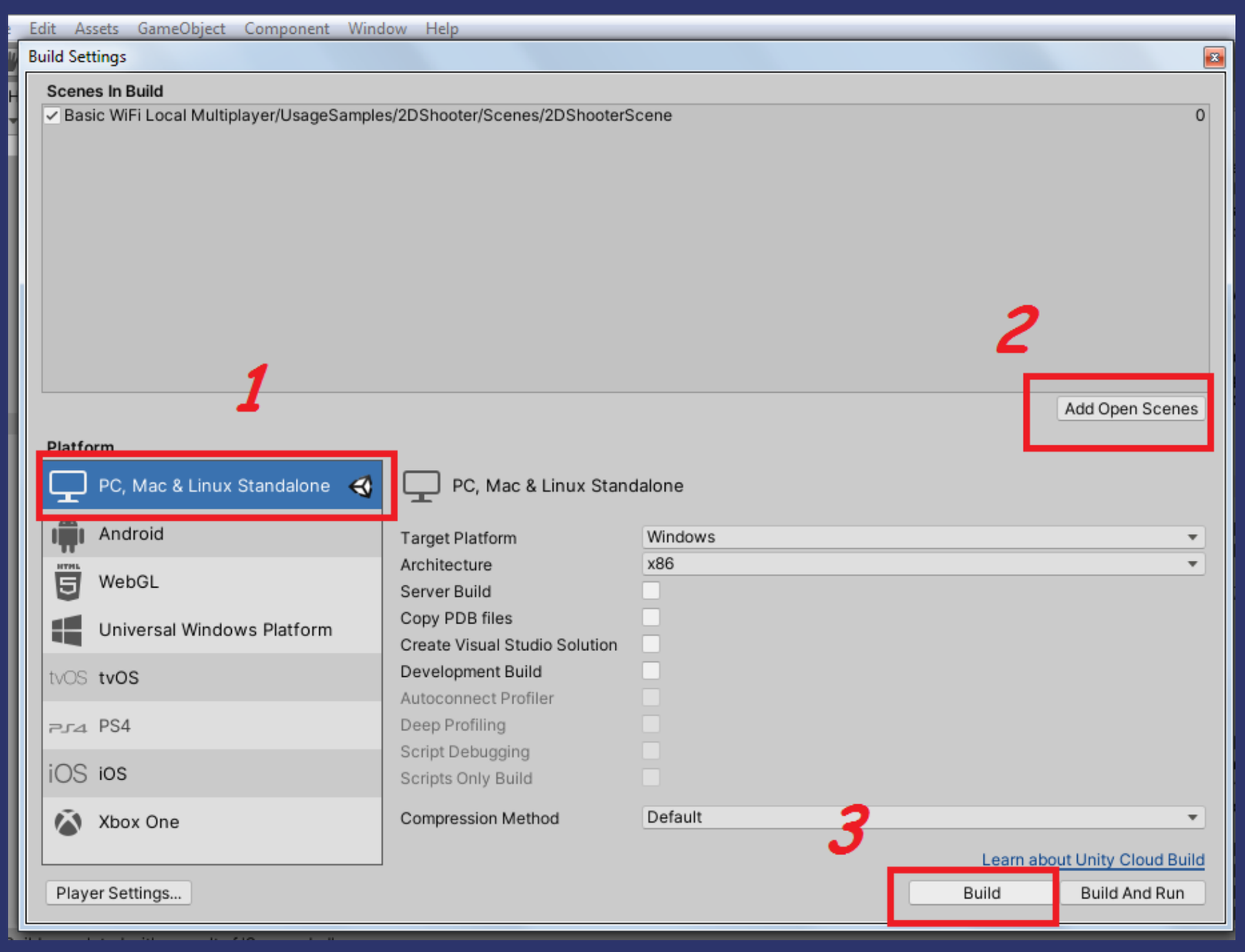
add the "**Obstacle**" Layer for all **ball prefabs** (except cueBall) and **NetworkBalls**. Also add to the gameObjects "**Planes**" in the Hierarchy within RaycastPlanes.

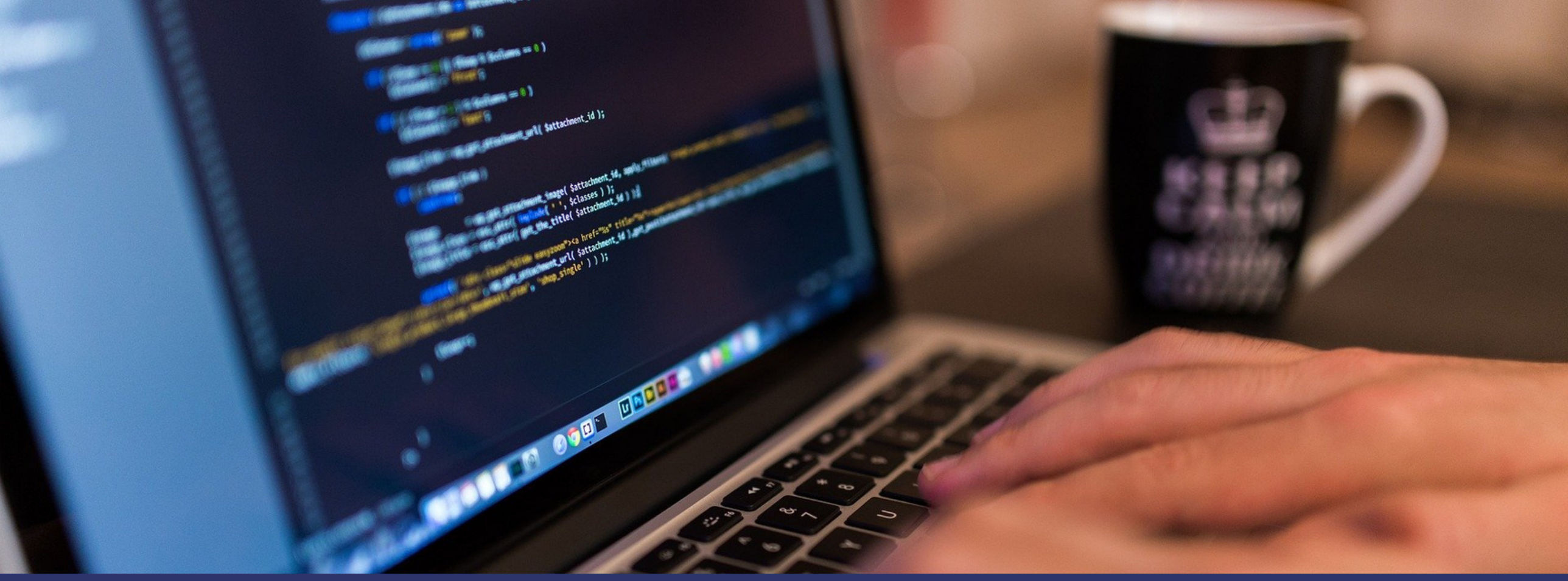




How to Test Game with more players







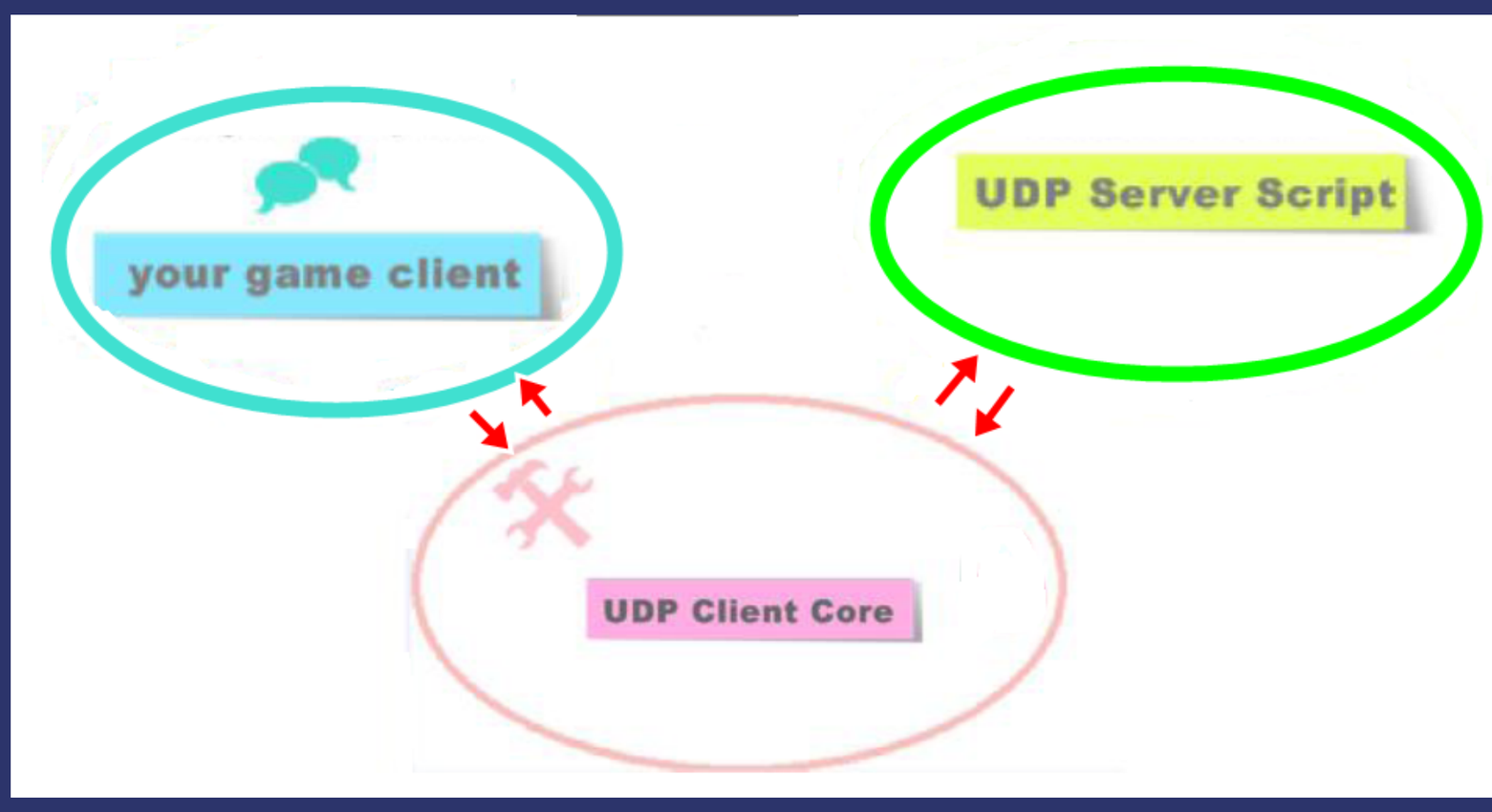
How to develop

Before developing your network aplication, keep in mind that the application will be divided into two parts:

- The Front-end: all visual, game mechanics and physics will be done in unity. Whatever game you have in mind, we will just use a main class called Network Manager to do all the interaction with the server.

- The Back-end: here comes the server, which will make the connection between all game clients.

UDPComponent



The UDP Core library is responsible for encapsulating all low-level operations required to open a UDP Socket and send messages to a UDP server.

There is no need to make any changes or modifications to this module.

1. The main class of this module is UDPComponent.cs, this class has the ability to act as a client or server, depending on the your choice.

2. This component as a client, provides three methods responsible for: connecting to a server(UDPComponent.connect()), sending messages to the server (UDPComponent.EmitToServer()) and receiving messages from the server (UDPComponent.On()) respectively.

3. Acting as a server also offers three methods: start server(UDPComponent.StartServer()), sending messages to the clients(UDPComponent.EmitToClient()) and
receiving messages from the clients (UDPComponent.On())

Note: the operation of this library
is very similar to the popular
socket.io library.

CONNECT TO SERVER

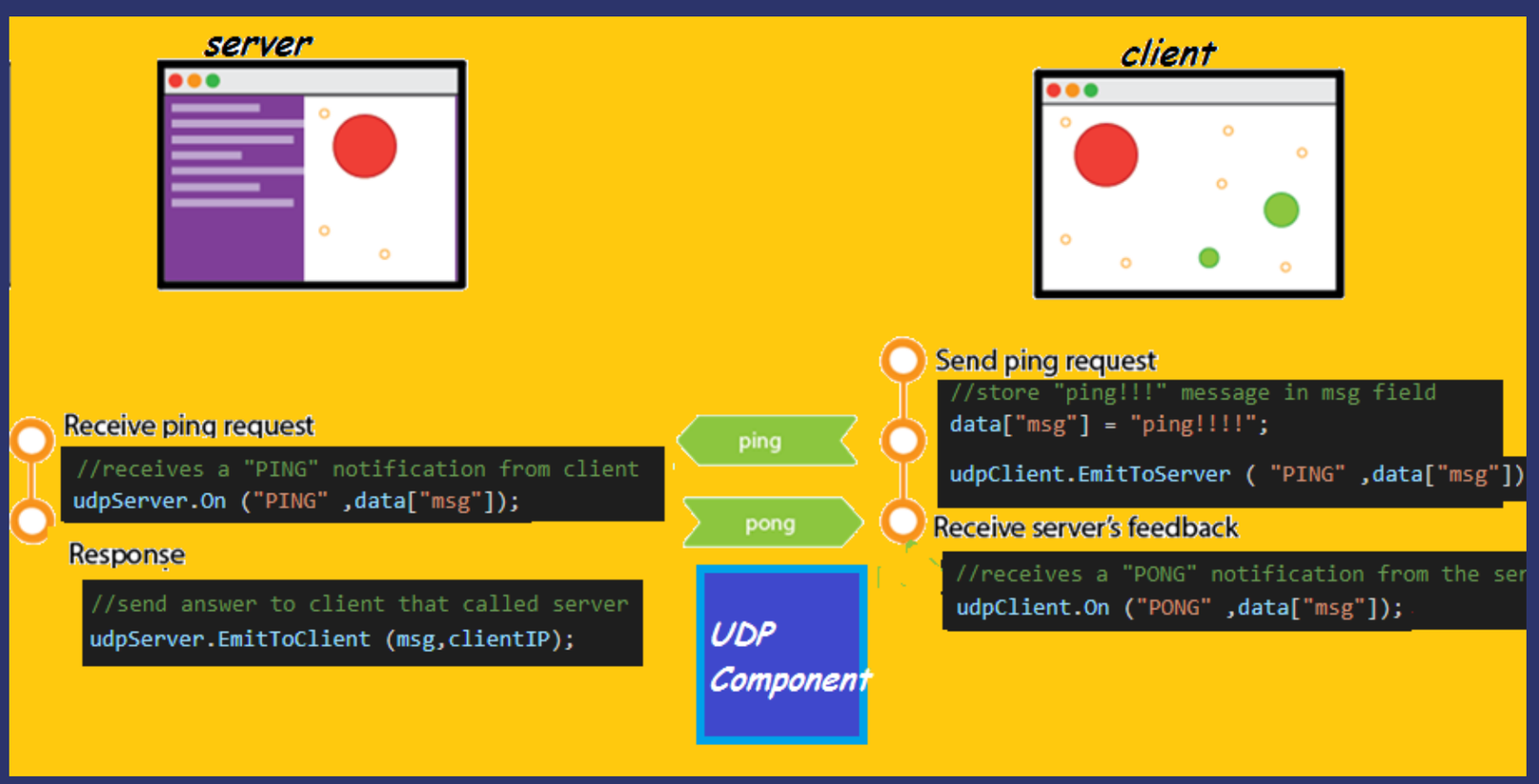
Connecting to a server with just one single line:

NetworkManager.cs

```
/// <summary>
/// Connect client to Server.cs
/// </summary>
public void ConnectToUDPServer()
{
    //connect to Server
    udpClient.connect (udpClient.GetServerIP (), serverPort, clientPort);

}
```

SENDING AND RECEIVING MESSAGES FROM THE SERVER.JS



You can use send information to server using the **UDPComponent.EmitToServer** method:

NetworkManager.cs

```
public void EmitPing() {


        // check if there is a server running
        if (udpClient.serverRunning) {

            //hash table <key, value>
            Dictionary<string, string> data = new Dictionary<string, string>();

            //JSON package
            data["callback_name"] = "PING";

            //store "ping!!!" message in msg field
            data["msg"] = "ping!!!!";

            //The Emit method sends the mapped callback name to  the server
            udpClient.EmitToServer (data["callback_name"] ,data["msg"]);


        }

        else
        {
            PingCanvasManager.instance.ShowAlertDialog("please start the server");
        }
```

the server get the client network manager message using:

server.js

```
// Use this for initialization
void Start () {

    // if don't exist an instance of this class
    if (instance == null) {

        //it doesn't destroy the object, if other scene be loaded
        DontDestroyOnLoad (this.gameObject);

        instance = this;// define the class as a static variable

        udpServer = gameObject.GetComponent<UDPComponent>();
        //receives a "PING" notification from client
        udpServer.On ("PING", OnReceivePing);

    }
    else
    {
        //it destroys the class if already other class exists
        Destroy(this.gameObject);
    }

}
```

- In server you can use send information to client using the **EmitToClient** method:

```
void  OnReceivePing(UDPEvent data )
{
    /*
        * data.pack[0]= CALLBACK_NAME: "PONG"
        * data.pack[1]= "ping"
    */

    Debug.Log("receive ping from game client");

     msg = Encoding.ASCII.GetBytes ("PONG:pong!!!");

     //send answer to client that called server
    udpServer.EmitToClient( msg, data.anyIP);

     Debug.Log ("[INFO] PONG message sended to connected player");

}
```

to receive messages from the server you can create a listener with the following command:

```
udpClient.On ("PONG", OnPrintPongMsg);
```

with the server information in hand, we can update the game status, create a function to treat the callback event:

```
/// <summary>
/// Prints the pong message which arrived from server.
/// </summary>
/// <param name="data">received package from server.</param>
public void OnPrintPongMsg(UDPKingEvent data)
{

    /*
     * data.pack[0]= CALLBACK_NAME: "PONG"
     * data.pack[1]= "pong message!!!"
    */

     Debug.Log ("received message: "+data.pack[1] +" from server by callbackID: "+data.pack[0]);

}
```

- The same steps can be applied for the other network functions in your game.

hands-on!

in this asset you will find the most varied examples for you to start learning how to develop your own application!

Contact us

rio3dstudios@gmail.com

rio3dstudios.com