

MultiLan v3.0

Multiplayer lan & network kit



1. Introduction	3
2. Installation.....	3
2.1. Import the package	3
2.2. Building.....	5
3. Running.....	8
3.1. The main menu.....	8
3.2. The waitroom	13
3.3. The map.....	15
4. Package and settings	20
4.1. Package content	20
4.2. Global parameters.....	22
4.3. In-game menu	23
4.4. Texts	24
4.5. Game menu	24
4.6. Waitroom	24
4.7. Spawn prefab.....	25
4.8. MenuCamera prefab	25
5. Customization.....	27
5.1. Change the player	27
5.2. Use your own maps	29
5.3. Change the GUI	31
6. Useful attributes.....	33
6.1. Get the current player data.....	33
6.2. Get the players list.....	34
6.3. Get current game data	34
6.4. Get the games list.....	36
6.5. Get the position of a player.....	36
7. FAQ.....	37
7.1. The network games list doesn't appear	37
7.2. Players cannot join to my public game	37
7.3. The host migration doesn't work	37
7.4. The game is lagging	38
7.5. My animations are not synchronized.....	38

1. Introduction

MultiLan is a starter kit for multiplayer network games.

With this package, a player can view the list of the games of his network (works only on Windows computers), join a game internal or external of his network by enter the host's IP or create and host a game himself.

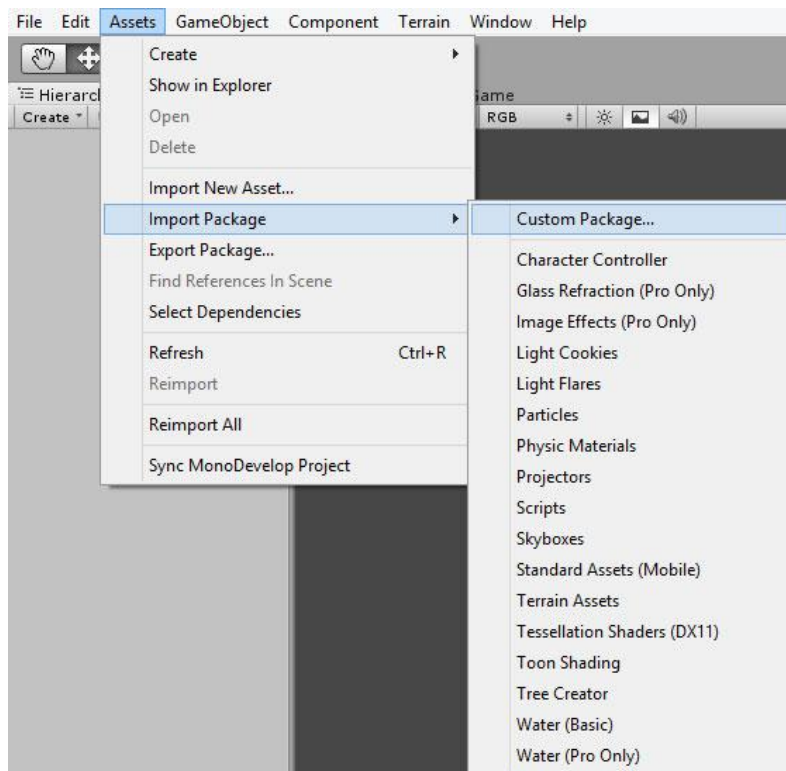
This package uses the Unity's Network class and C# scripting so it works easily and doesn't require any other installation. You just have to install the package on your Unity Editor and follow this guidebook.

This package is full compatible with MultiOnline Asset.

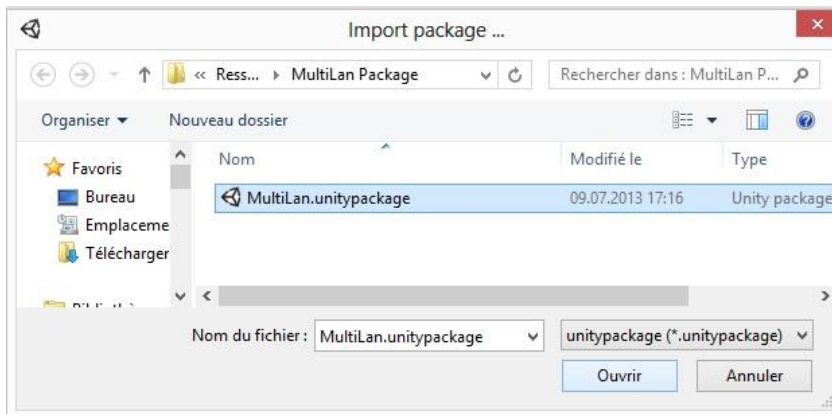
2. Installation

2.1. Import the package

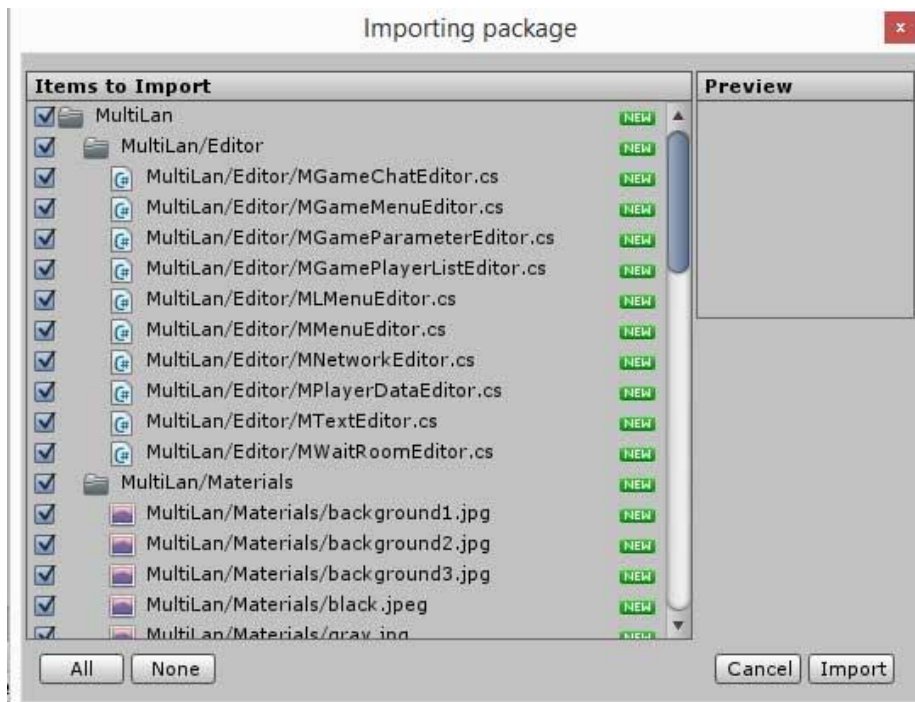
Once you have download the package, open Unity Editor, and click on Assets/Import Package/Custom Package... :



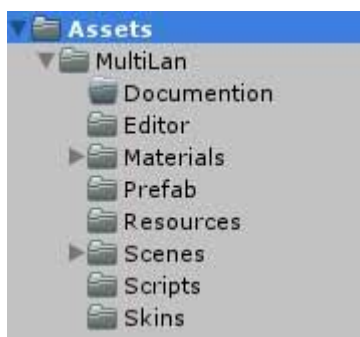
Select the package and click on "Open" :



Let all boxes checked and click on "Import" :



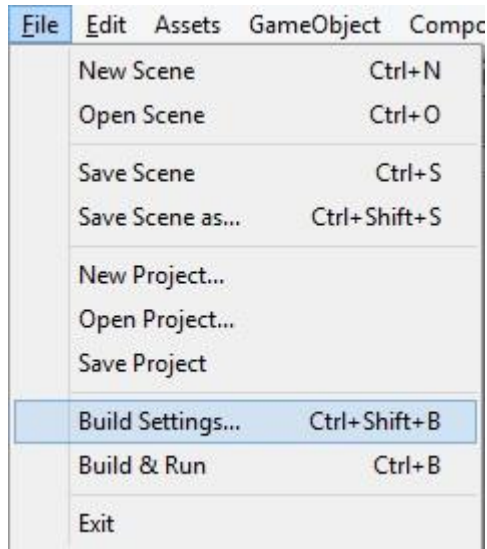
Your package is imported, you have now a folder named "MultiLan" on the Asset folder, which contains 8 other folders :



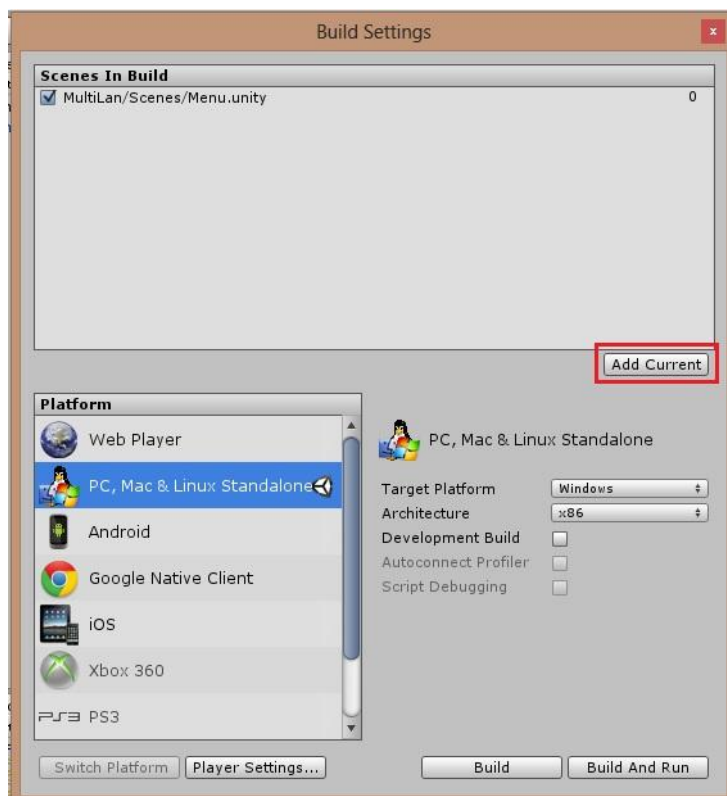
2.2. Building

Before try our package, we have to build the differents scenes together, else it's impossible to load a scene from another (so, nothing works).

First, go on the Scenes folder and open "Menu". Next, click on File/Build Settings... :

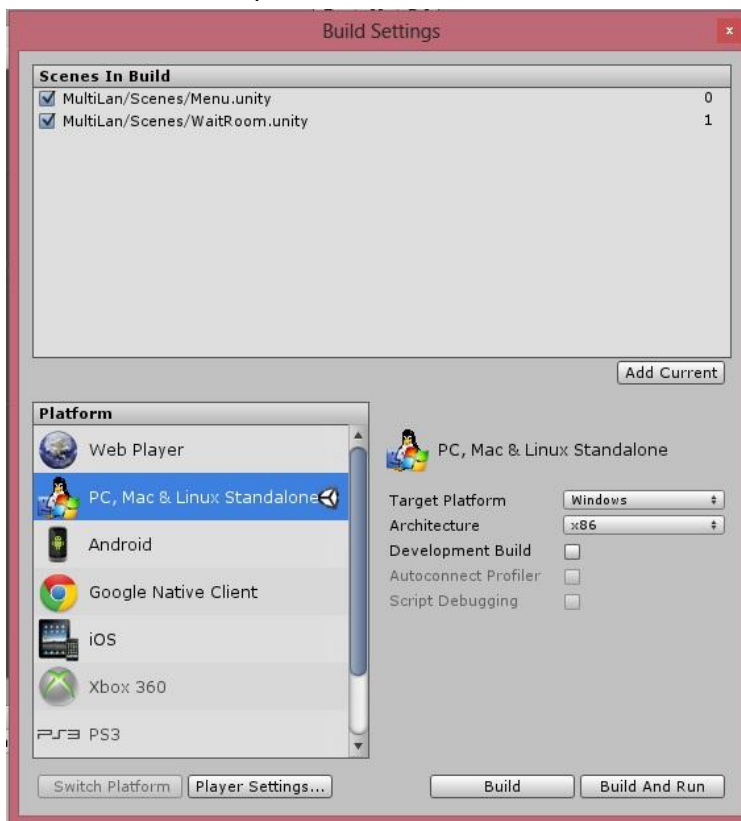


It opens the Build Settings panel. Click on the button "Add Current" in order to add the current scene on the "Scenes in Build" :

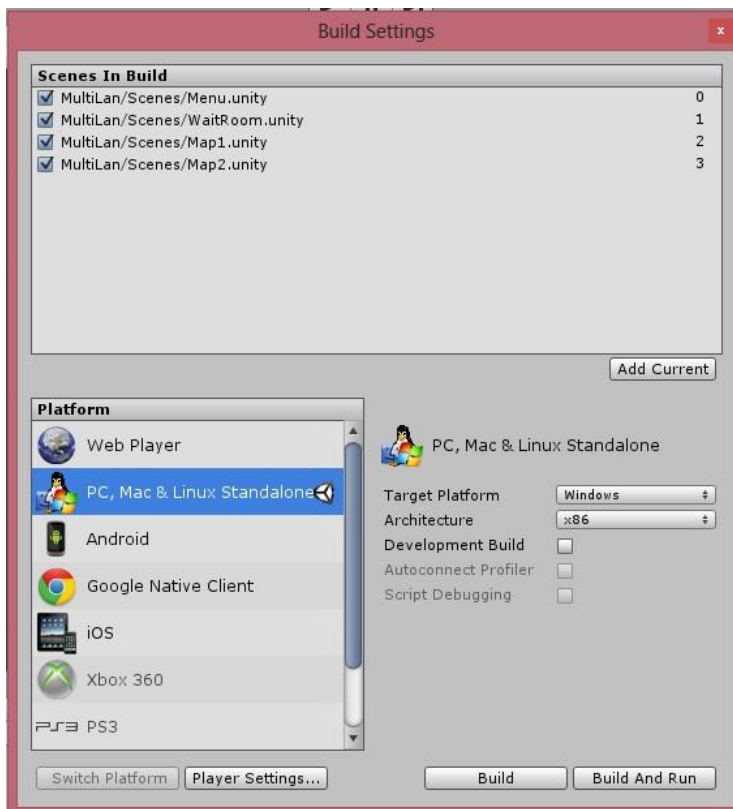


Each scene of the game must be on this list, else they cannot be loaded from the script.

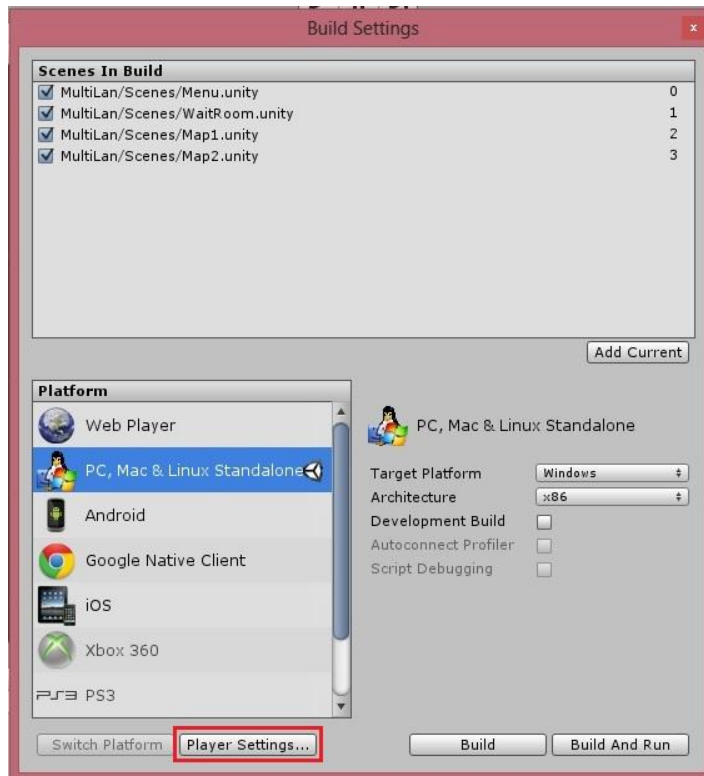
So now, let open the Build Settings panel, and open the WaitRoom scene.
Once the scene is opened, click on "Add Current" :



And do exactly the same thing for the two remaining scenes (Map1 and Map2).
Once you have finished, the Build Settings panel must look like that :



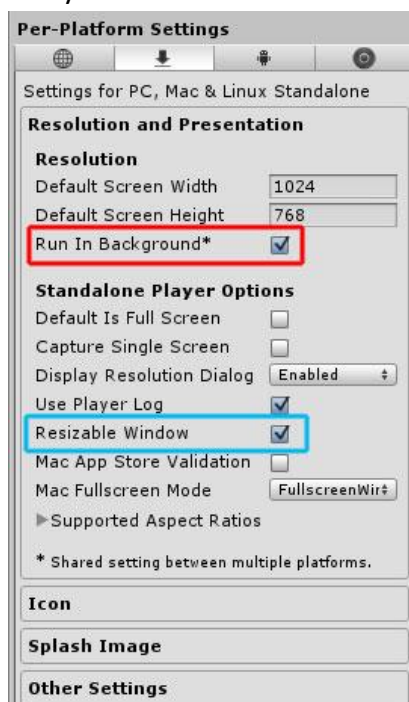
Now, click on the button "Player Settings..." :



The player Settings panel is now open in the inspector.

Check the box "Run in Background", this will allow you to try the multiplayer with many windows open in the mean time on your computer.

You can also check the box "Resizable Window", I find that's very useful when I work with many windows but it is not mandatory.



Now, you can click on the "Build" button, so that we will try the multiplayer with two windows : an host an a guest.

Yan can name your builded game as you want. When it's done, you get a .exe file and a data folder :

 MultiLanTest_Data	08.07.2013 01:00	Dossier de fichiers
 MultiLanTest.exe	07.05.2013 12:33	Application

Each time you'll want try your game with many windows, you'll have to build it first. You can go on "File/Build and Run" or use the shortcut "Ctrl+B".

It's now time to see how does the multiplayer work.

3. Running

3.1. The main menu

You can open directly your .exe file or open the scene "Menu" on your editor and click on "Play".



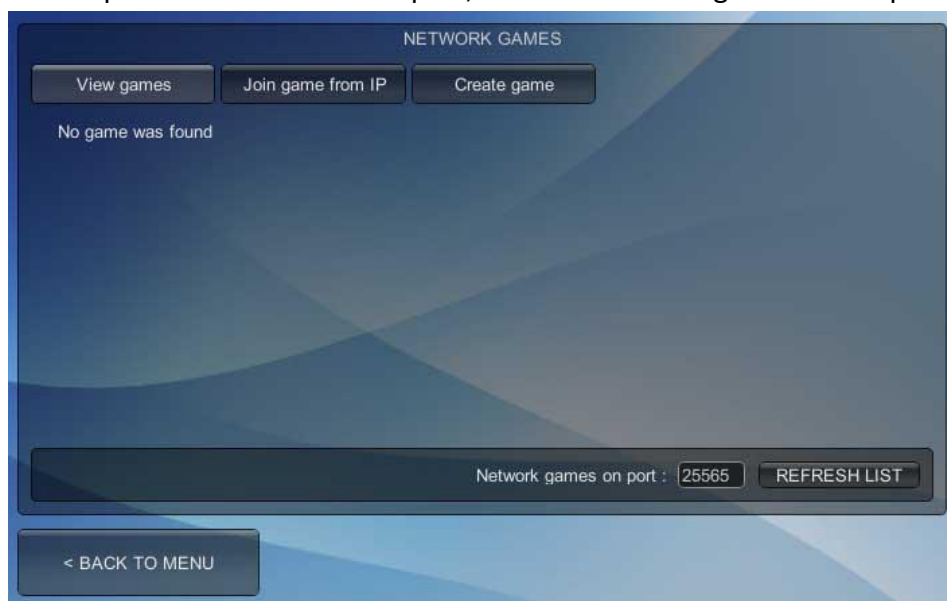
The main menu contains 3 buttons:

- Profil
- Network games
- Exit: for exit the game

- Profil : here you can choose and save the player name. For the sample, it's very basic, but you could add many things in this menu (choose avatar, show player's statistics, game preferences...)



- Network games: you have here 3 submenus :
 - View games : show all open games on your network (so, it doesn't show the full game and started game if we have chosen that we cannot join them). With the field "port" you can choose on which port you want to search games (by default, the script search on the 25565 port, but it can be changed from the parameters).



If there is an open game on your network, the game list looks like that :

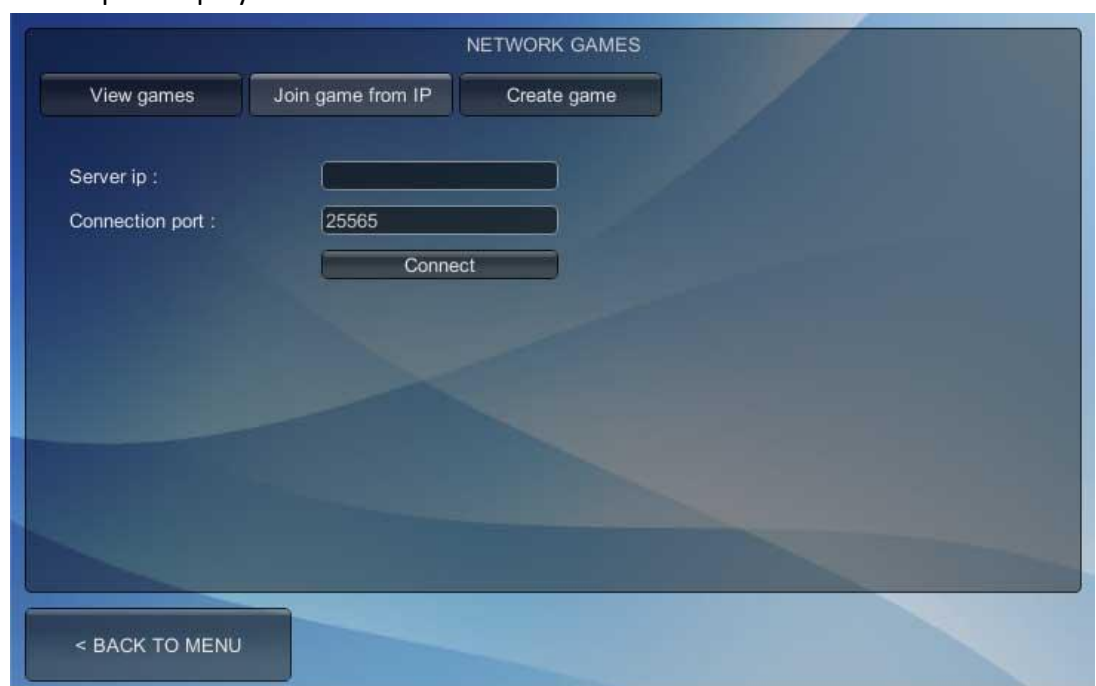


You can sort the games by game name, host name, map name, player number, game type, game status and ping.

You have a filter system on the left, where you can filter the games using the following parameters: map, ping, full games, started/no started, private/public and max player numbers (you can enable/disable the filter system, we see how later)..

And for finish, you have a search which can be used to search a game name or an host name.

- Join game from IP : for join a network or external game directly from the host's ip. The player have to fill out two fields :



- Server IP : the host's IP. If the host is on the same network as the player, the player have to put the host's private IP, else, the player have to put the host's public IP (the two IP are given on the Create Game menu)
 - Connection port : the port on which the host has created the game. By default, the field is filled by the port 25565, but it can be changed
- Create game : for create and host a game. It contains many fields :

The screenshot shows a 'NETWORK GAMES' menu with three buttons at the top: 'View games', 'Join game from IP', and 'Create game'. Below these, there are several fields for configuring a game:

- 'Game name' with an empty text box.
- 'Game type' with two buttons: 'Private' and 'Public'. Below these buttons is the text 'Only people on your network can join the game'.
- 'Server private IP' with the value '192.168.1.104'.
- 'Server public IP' with the value '86.9.202.11'.
- 'Port' with the value '25565'.
- 'Maximum players' with a numeric input showing '2' and navigation buttons '<<', '<', '>', and '>>'.

 At the bottom of the menu area is a 'Create game' button. Below the entire menu area is a '< BACK TO MENU' button.

- Game name : the game name :P
- Game type : for choose if the game will be private (only joinable by the network members), or public (everybody can join the game if they have the host's public IP address)
- Server private IP : shows the host private IP (so that he can give it to members of his network who want to join his game)
- Server public IP : shows the player public IP (so that he can give it to peoples external of his network who want to join his game)
- Port : the game's port, 25565 by default, but it can be changed
- Maximum players : select the maximum players number of the game
- Important note : so that players can join a public game, the host has to open the game's port on his router, else, even if the host chooses the button "public", nobody else of the players from his network would can join his game.
- Note : the "Server public IP" is searched from a web ip service. You can used another web service or (the best thing to do) create your own if you've got a web hosting.

Look at the comments on the source code of the script "*MLMenu.cs*", line 14 for more informations, everything is explain for use your own web page as an IP service (you don't need to know how to code for do that).

If you use MultOnline, the script *askip.php* provided with MultOnline package contains what you need to use your own ip service. Go on "*MLMenu.cs*" line 29, and put your URL of your *askip.php* page on the variable *ipOnline*.

Since we have seen everything about the menu, we are going to create a game.

So full the form and click on "Create Game" :



The screenshot shows a web interface titled "NETWORK GAMES". At the top, there are three buttons: "View games", "Join game from IP", and "Create game". Below these, the "Create game" form is visible. It includes a "Game name" field with the text "Game", a "Game type" section with "Private" and "Public" buttons (where "Public" is selected), and a note stating "Everyone can join the game with your public IP". The "Server private IP" is set to "192.168.1.104" and the "Server public IP" is "95.9.222.11". The "Port" is "25565". The "Maximum players" is set to "4", with navigation buttons "<<", "<", ">", and ">>". A "Create game" button is at the bottom of the form. Below the form is a "< BACK TO MENU" button.

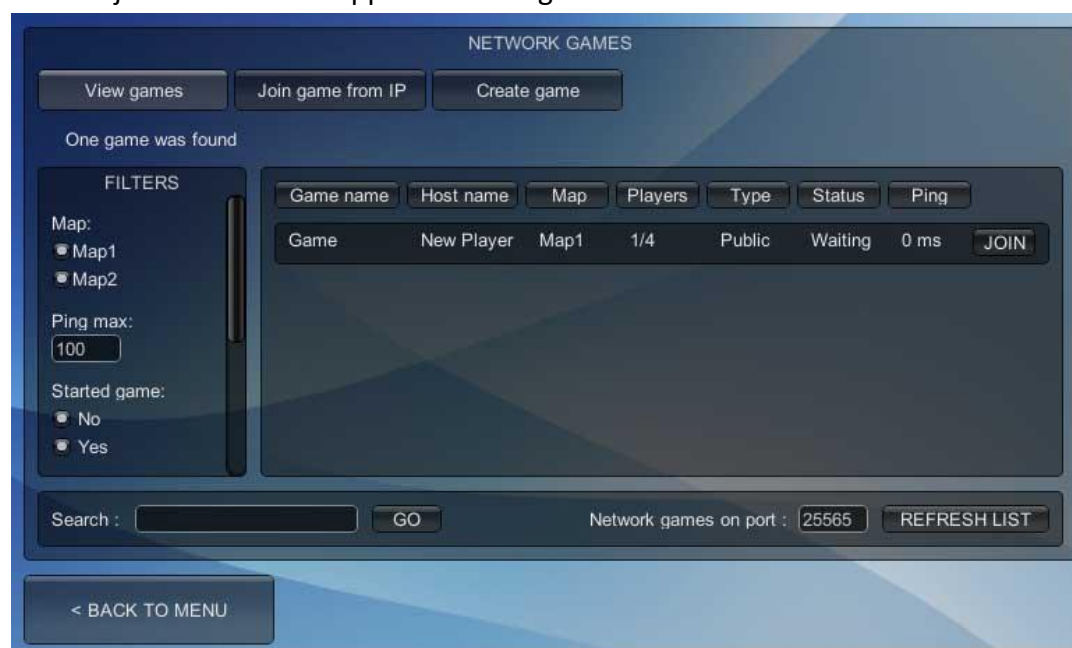
You are now in the waiting room.

3.2. The waitroom

The waitRoom shows the players list, the game settings and, since we are the host of the game, we can choose the map here and load the game.



Now, open your .exe file for join your game from another window. The game you have created just before must appears on the game list :



Click on join. You can see the waitroom as a guest. It's quite the same, you just cannot choose the map and cannot load the game:



Go back in the host's waitroom. You can see the new player with a cross button next him: it's the button to exculde a player:



If you click on it, a windows will appears to confirm the exulsion of this player:



But we will not excule our second player now since we need him to test the multiplayer.

Now, go on the window which have created the game and click on "LOAD GAME". A timer of 5 seconds begins and the map is loaded, and the players are spawned.

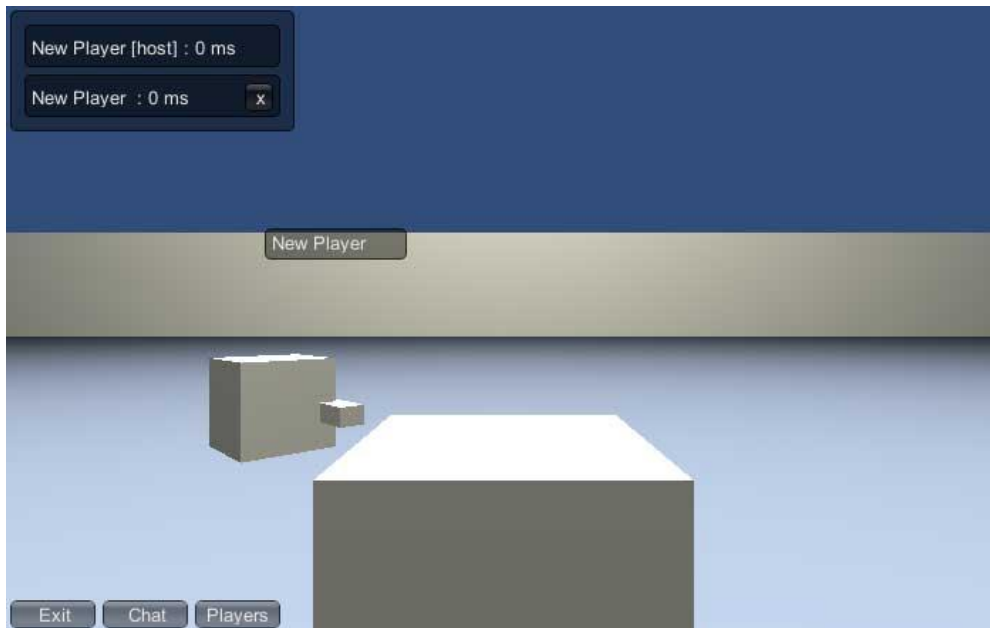
3.3. The map

You can now test the multiplayer on a very nice blue map, with two wonderful cubes :P



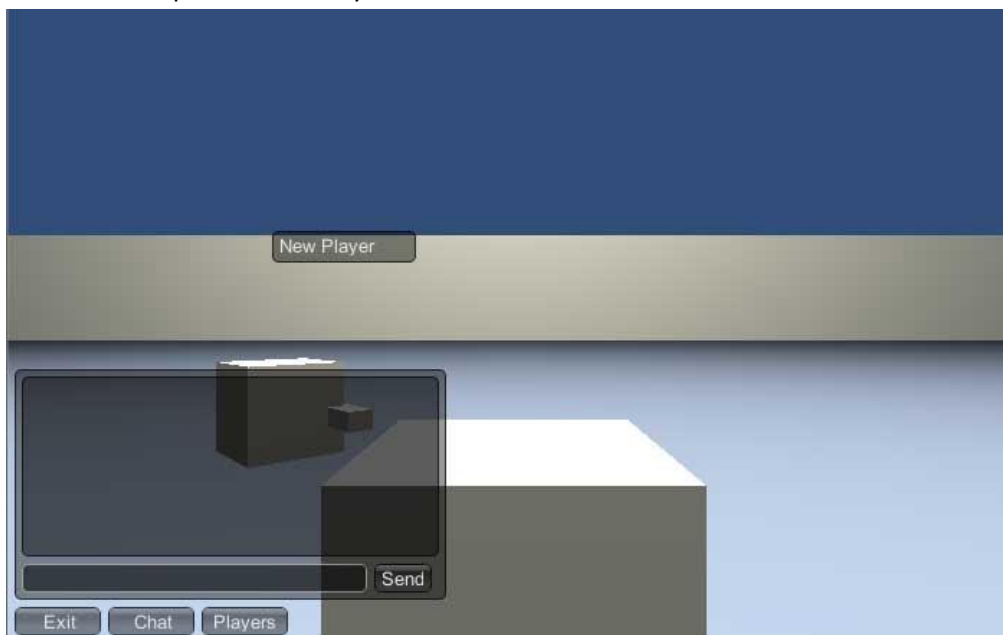
In this new version I have add the display of the player name (you can enable/disable it, I'll show you how later), and a little game menu (each button of this menu can also be enable/disable).

If you click on the menu "Players", you can see the player list, with, if you are the host, the button to exculde a player (sorry, on my example my two players have the same name !):



To get rid of this menu, just click one more time on "Players" button.

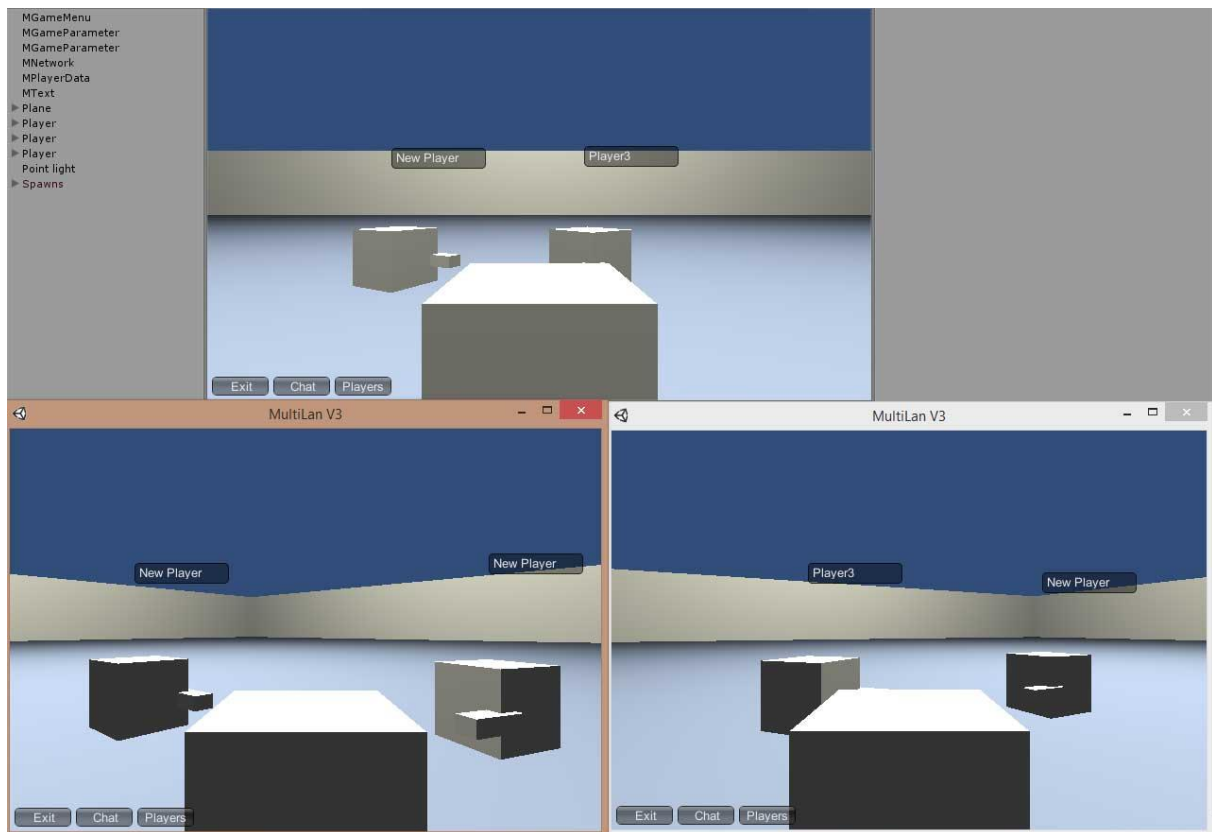
You can also open the chat if you click on "Chat" button:



... And exit the game, if you click on "Exit" - The "Esc" key makes appear the same window:

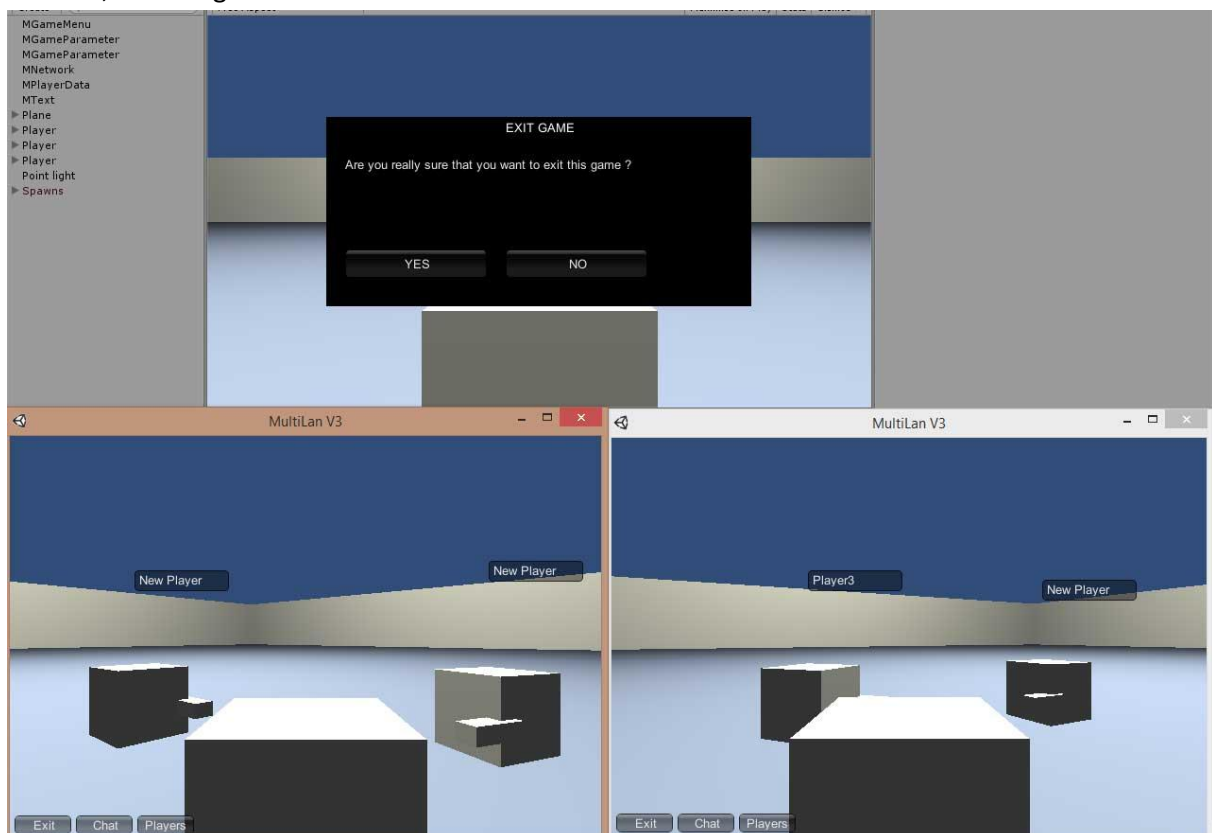


You can now add a third player on your game with opening the game on a new window. Since the game is already started, this new player can directly join it. So, click on the button "JOIN GAME" on the waitroom, and your player will join the game.



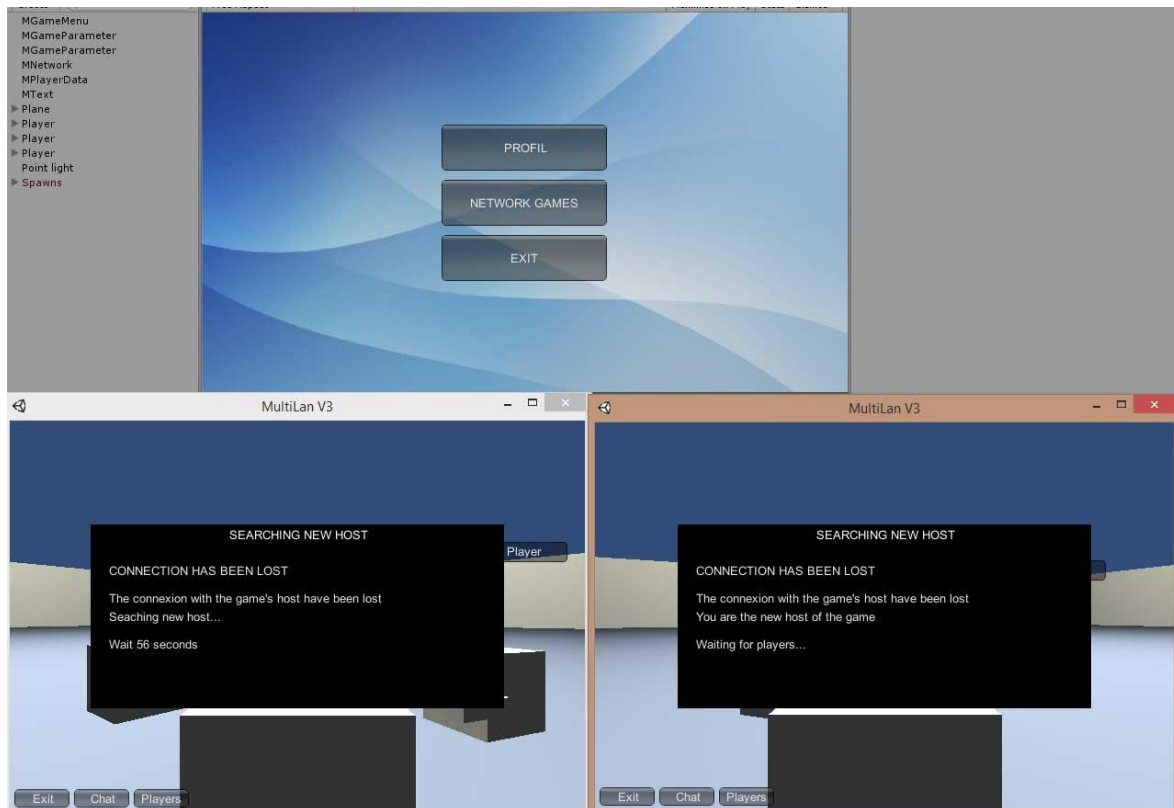
For finish, we'll test the host migration.

For that, exit the game on the window which is host.

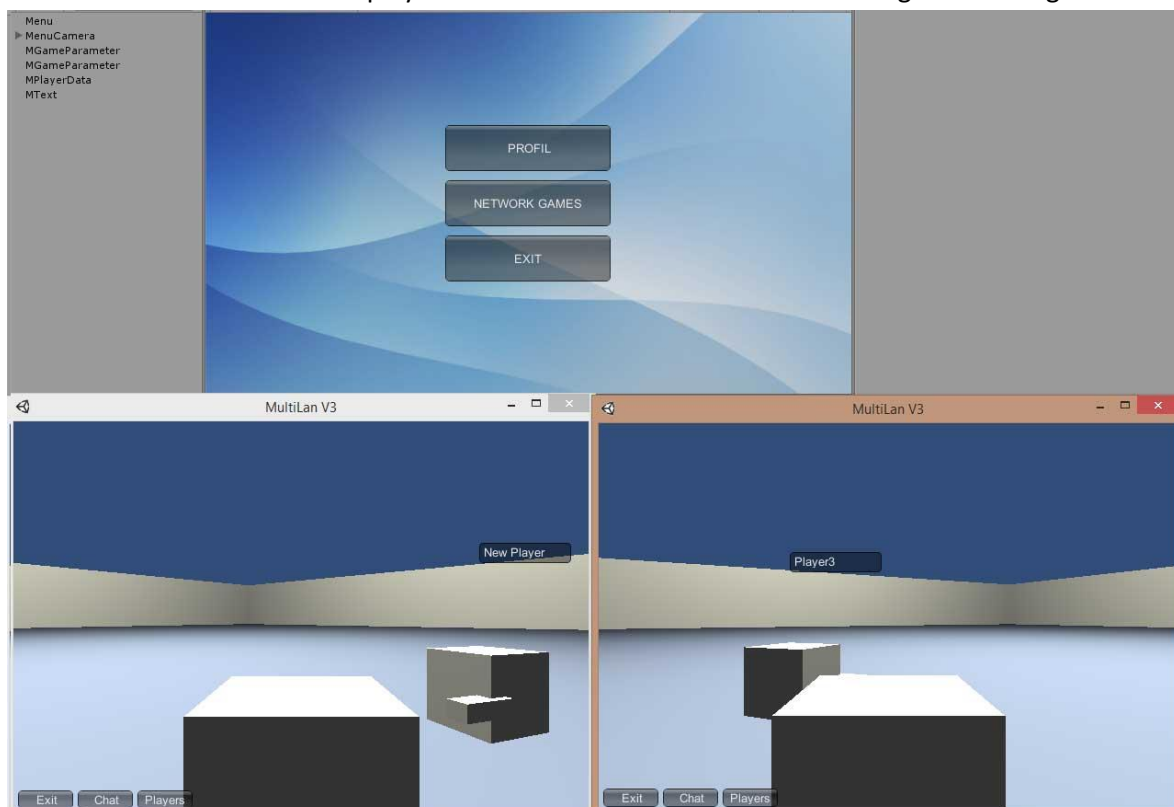


The two other player will first receive a message which says that we are searching a new host. And after few seconds, one of them will have the message "You are the new host of the game - Waiting

for players". It mean that this player is going do become the new host.



And 5 seconds later the other player is connected to the new host and the game start again:



4. Package and settings

4.1. Package content

Now, we will see the different elements of the package.

- Editor : contains the editions script, you have no reason to open them
 - MGameChatEditor
 - MGameMenuEditor
 - MGameParameterEditor
 - MGamePlayerListEditor
 - MLMenuEditor
 - MMenuEditor
 - MNetworkEditor
 - MPlayerDataEditor
 - MTextEditor
 - MWaitRoomEditor
- Prefab : contains the game prefabs
You can change it, and use it in your game scenes, I will explain how later.
 - MenuCamera : a camera prefab used on the menu scene and on the waiting room for manage the background. You can use it if you add menus on your game
 - Spawns : this gameObject is necessary for load the players, if you create your own maps, you have to add it on your scenes
 - Player : for the sample it's a simple cube but it is strongly recommended to change it for your final game :P
- Resources : contains some prefab which are automatically loaded from script using Resources.Load(). Maybe you have already a Resource folder in your project, but Unity is able to manage and load resources from multiple Resource folder, so you don't have to worry about it.
 - MGameMenu: manages the in-game menu
 - MGameParameter: manages all the game parameters
 - MNetworkManager: manages the network connections and RPC functions
 - MPlayerData: manages all the player data during the game
 - MText: manages all the texts of the asset
- Scenes : contains the four scenes of the package and an images folder :
 - Menu : the main menu of the game. It's the entry point of the game
 - WaitRoom : the waiting room, used for wait for players before start game
 - Map1 : the first sample map
 - Map2 : the second sample map
 - The MapScreen folder : contains the screenshots of the games map (so that we can show the game's map on the waiting room)

- Scripts : contains the game scripts :

All scripts which are editable have an editor (we will see it later). So you don't have to modify some values directly from scripts (but of course, you can read it and change it if you need). MFilter : functions script, use to check player's entries before use it.

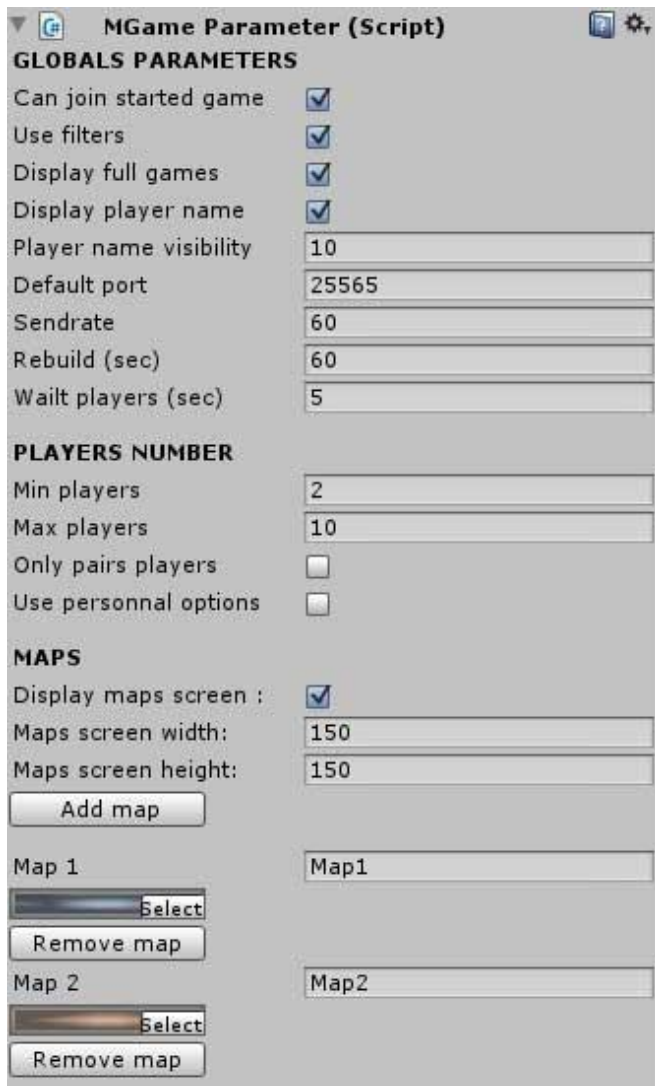
- MGame : this class is used for formatting the open network games
- MGameChat : manages the in-game chat
- MGameMenu: manages the in-game menu
- MGameParameter: this script contains the parameters of the game
- MGamePlayerList: manages the in-game player list display and options
- MLIpList: functions script, used to scan the IP of the network (for after search the open games on these IP)
- MLMenu: manages the menu specific to MultiLan
- MMenu : manages the main menu
- MNetwork : manages all the network
- MPlayerData : contains the players' informations during game (is player login, IP, username, status...)
- MPlayerMove: manages the player movements, it is currently a very basic script, you can modify many things on it and add functions
- MPlayerNetwork: disables the other players and saves our player's current position in prevent of a host migration
- MSpawn : loads the player on the map
- MText: contains all the script of the asset
- MUser : this class is only used for formatting game's user list
- MWaitRoom: manages the waitRoom

Note : many scripts and components are the same in MultiLan and MultiOnline packages. The scripts specific to MultiLan package begin with the letters "ML", the others begin with the letters "M".

In order to make MultiLan and MultiOnline full compatible and usable separately in the mean time, I had to put in comments few lines which required MultiOnline components on the scripts MMenu and MPlayerData. If you want know how to use MultiLan and Online together, everything is explained on the other documentation which comes with the package.

4.2. Global parameters

Global parameters are editable from the MGameParameter object which is on Resource folder. If you click on this object, you will have this on the inspector:



The image shows the Unity Inspector window for the **MGameParameter (Script)** object. It is divided into three main sections: **GLOBALS PARAMETERS**, **PLAYERS NUMBER**, and **MAPS**.

- GLOBALS PARAMETERS:** Contains checkboxes for 'Can join started game', 'Use filters', 'Display full games', and 'Display player name', all of which are checked. Below these are input fields for 'Player name visibility' (10), 'Default port' (25565), 'Sendrate' (60), 'Rebuild (sec)' (60), and 'Wait players (sec)' (5).
- PLAYERS NUMBER:** Contains input fields for 'Min players' (2) and 'Max players' (10). There are also checkboxes for 'Only pairs players' and 'Use personal options', both of which are unchecked.
- MAPS:** Contains a checked checkbox for 'Display maps screen :'. Below it are input fields for 'Maps screen width' (150) and 'Maps screen height' (150). There is an 'Add map' button. Below that, there are two map entries: 'Map 1' and 'Map 2'. Each entry has a small thumbnail image, a 'Select' button, and a 'Remove map' button. The text 'Map1' and 'Map2' is displayed next to the thumbnails.

Global paremeters

- Can join started game: if it's not checked, the started game will not appear in the game list.
- Use filters: if it's checked, the filters will appear on the game list.
- Display full games: if it's checked, the full games and the filter "view full games" will appear on the game list (without the button "join").
- Display player name: if it's checked, the players will have their name display above them in game.
- Player name visibility: if the previous case is checked, this field appears. You must enter here the distance from which the player name must disappear.
- Default port: the default port which will be written on the create game menu.
- Sendrate: the sendrate, more you put a big value here, more the multiplayer will be fast.
- Rebuild (sec): the time to find a new host during an host migration.
- Wait players (sec): the time to allow players find the new host during an host migration.

Players number

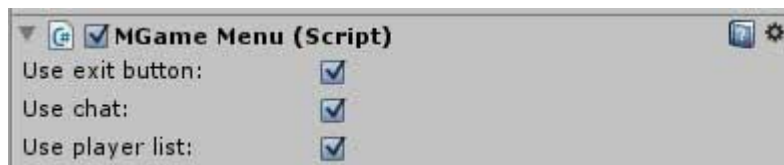
- Min players: minimum players number.
- Max players: maximum players number.
- Only pair players: if you allow only pairs player number. If you check this box, only the pairs numbers between min players and max players will appear on the player number field.
- Use personal options: if you check this box a button "Add option" appears, and a field named "Option 1" with the button "Remove option" below. You can add as much "options field" as you want, and fill it with players number options. All these number will appear on the game menu "Create game". The options will automatically sorted by increasing order, so you don't have to worry about it.

Maps

- Display map screen: if you check this box, the map screen will appear on the waitroom.
- Maps screen width: appears if the previous box is checked, define the width of the map screen in the waitroom.
- Maps screen height: appears if the previous box is checked, define the height of the map screen in the waitroom.
- Add map: to add a map in your game. If you click on the button, it add a new field where you have to enter the name of your map (you must enter the name of the scene), and you can also add the map screen using the texture field.

4.3. In-game menu

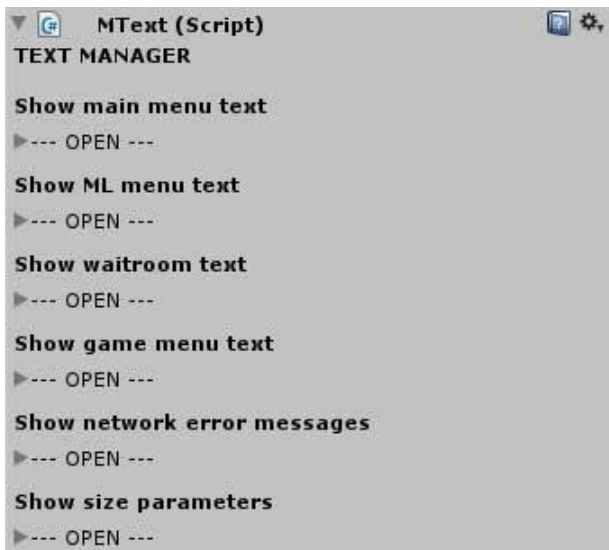
The in-game menus are editable from the MGameMenu object which is on Resource folder. If you click on this object, you will have this on the inspector:



You can define here which menu must appear in game, just check the box of the menus you want.

4.4. Texts

All the texts which are displayed on the asset are editable in a same place: from MText object, which is on Resource folder. If you click on this object, you will have this on the inspector:



You can open each panel to display the texts and change it.

4.5. Game menu

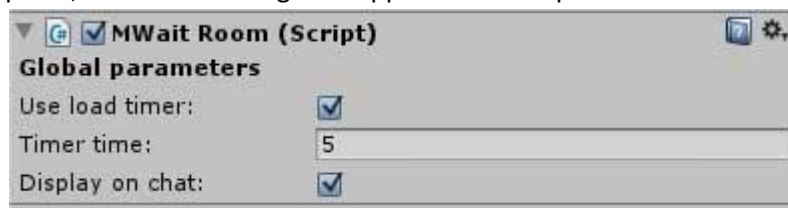
Now, go in Scenes folder, and open the scene Menu. Click on Menu object from "Hierarchy" panel, and some settings will appear in the inspector:



- Default player name: the default player name !
- Use "join from IP" menu: if it's checked, the menu "Join game from IP" will appear on "Network games" menu, else it will not.
- Use "view games" menu: if it's checked, the menu "View games" will appear on "Network games" menu, else it will not.

4.6. Waitroom

Now, in Scenes folder, and open the scene Waitroom. Click on WaitRoom object from "Hierarchy" panel, and some settings will appear in the inspector:



- Use load timer: If you want to use a timer when the host starts the game. If you check this box, you can also choose :
- Timer time: the timer's time.
- Display on chat: if the timer displays on the chat or not (if not, it displays only on the load button)

4.7. Spawn prefab

Go on the prefab folder and click on the prefab named "Spawns".

There is only one thing to set here, but that's a very important thing : the player prefab.

So, if you change the player, it's here you have to refer your player's gameObject.



4.8. MenuCamera prefab

In prefab folder, the "MenuCamera" is used to define the menu's background. If you want to change the background, go on the prefab folder, "open" the "MenuCamera" and click on " Background":



☒ Background
 ☐ Static

Tag
 Layer

Transform

Position
 X Y Z

Rotation
 X Y Z

Scale
 X Y Z

Plane (Mesh Filter)

Mesh ☒ Plane

Mesh Collider

Is Trigger ☐

Material

Convex ☐

Smooth Sphere Collide ☐

Mesh

Mesh Renderer

Cast Shadows ☒

Receive Shadows ☒

Materials

Use Light Probes ☐

background-6

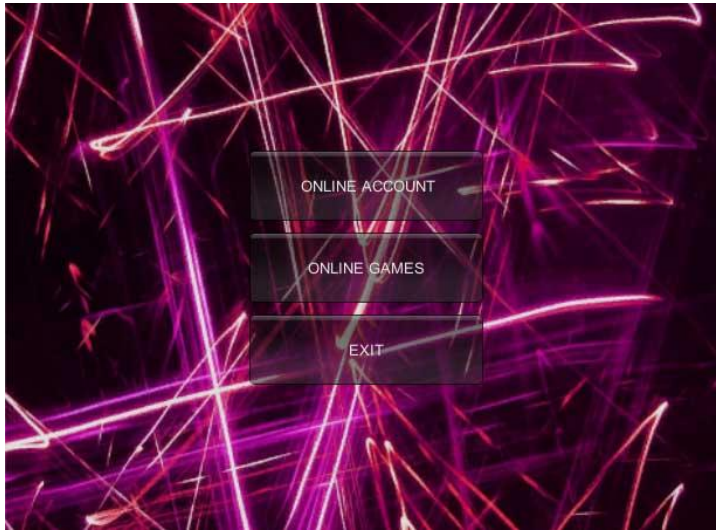
Shader

Main Color

Base (RGB)

	Tiling	Offset
x	<input type="text" value="1"/>	<input type="text" value="0"/>
y	<input type="text" value="1"/>	<input type="text" value="0"/>

Now, if you click on "play", you'll have your new background in the main menu and in the waiting room:



Note: if you want to change the background only in one scene, change the background from the MenuCamera object which is on the scene, not from the prefab.

5. Customization

We will see now few step by step examples about how customize this package to create your own game.

5.1. Change the player

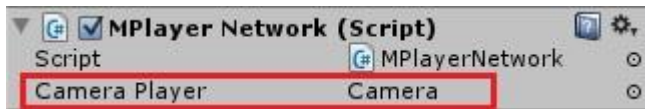
The player is one of the first things you'll want to change.

The step one is to create your own player prefab. There are just some little things he must have for working with the package :

- A network view (Add Component / Miscellaneous / Network view). The network view State Synchronization must be on "Reliable Delta Compressed" :



- The script *MPlayerNetwork* : this script manages the player around the network and disabled the other network players. So you must attach it on your player.
- Your player's camera must be attached on the script *MPlayerNetwork* :



- When your player is finish, you have to save it as a prefab, and click on the "Spawns" prefab. From here, drop your new player gameObject on the field "Player prefab" :



- You can change or remove the script *MPlayerMove*. This script is not necessary for the application's working (provided that you use another move script). But you should look at it and maybe use his "Start()" and "Update()" functions : these functions disable the player movement in some situations (host migration, exit game panel open...), so you could find it useful. So you could keep this script and change just the function *MovePlayer()* and add your new functions.

Important: if you change the player movement script, or add any new scripts on you player, you must disable these scripts for the others Network views.

For that, go on the function "Start()" in *MPlayerNetwork*, line 18 for disable the scripts for the others networkviews.

The disable instruction is this one :

```
this.GetComponent<THE NAME OF YOUR SCRIPT>().enabled = false;
```

Put this instruction in the place of the red box:

```
14 void Start () {
15     if (!networkView.isMine) { // If this is not my player
16         Destroy(cameraPlayer); // Destroy camera
17         this.enabled = false; // Disabled this script
18         this.GetComponent<MPlayerMove>().enabled = false; // Disabled
19
20     } else { // If it's my player : search the networkManager component
21         networkSrc = GameObject.Find("NetworkManager(Clone)").GetComponent<NetworkManager>();
22         networkSrc.PlayerSpawned(); // Informs the other players that
23     }
24 }
25 }
```

Note: I have made a full video tutorial about how change player. It has been made with MultiOnline, (an older version) but the process is exactly the same with MultiLan:

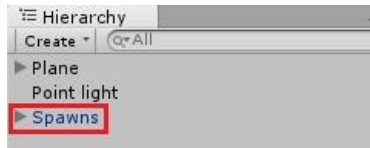
https://www.youtube.com/watch?v=obB8gj_iXoA

5.2. Use your own maps

Create new map is very easy.

- The Spawn

When you map is done, you just have to put the "Spawns" prefab on it.



This prefab contains all the spawns points for your map.

You can add as much spawn point as you want (and name it as you want too, they just have to be children of the "Spawns" gameObject). You can place them where you want, but slightly higher than the floor of your map (else the player could pass through).

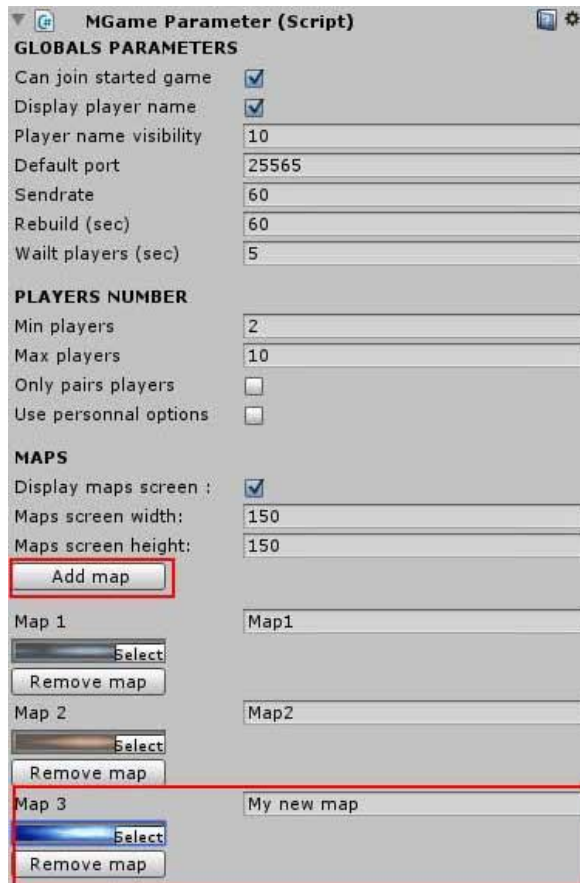
Note : the different spawns points can be named as you want, but you must not change the name of the "Spawns" prefab.

- Add the new map on the game

Now, your map is ready. You just have to add it on the game.

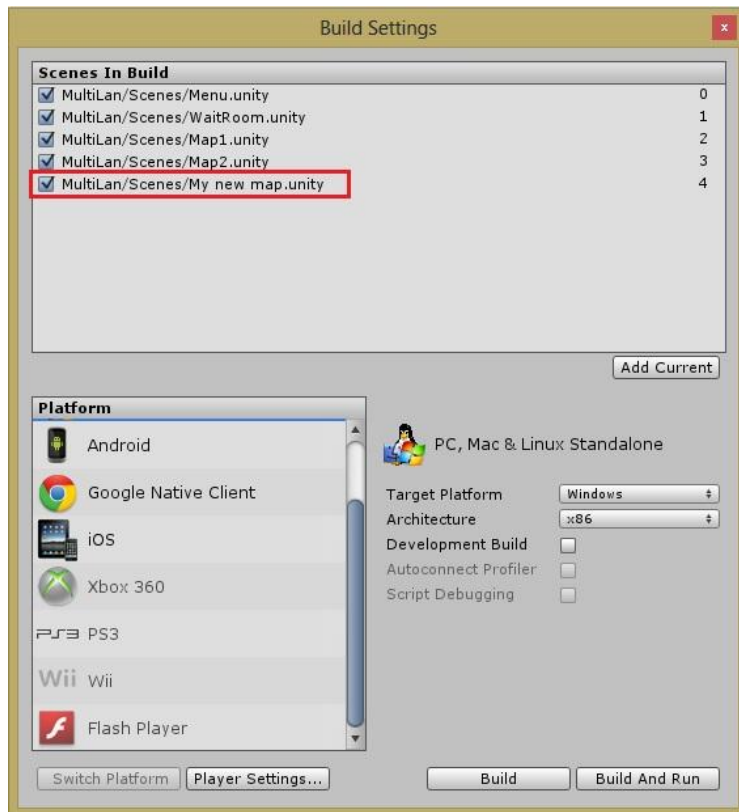
- First, go on Resource folder and drag MGameParameter object on your current scene (the scene from where you do this has not matter) .

On the inspector, add your map on the "MAPS", as it's explain on part 4.2: click on "Add new map", enter the name of your [map's scene](#) and the screen of your map on the new field (for this example, I just used a background):



Now, drop back MGameParameters on "himself", on the Resource folder, and delete the MGameParameters object which you dragged on your game scene.

Second, open the build settings on Unity (File/Build Settings...), add your new scene on the "Scenes in Build" (as show on part 2.6) and click on "Build" :



That's all, your new map will now appears on the waitroom and can be choose by the host for use it in game:



5.3. Change the GUI

If you don't like the way how the menu or the waitroom appears, you can:

- Add [GUI Skins](#): the in-game menu has currently a custom skin (named " gmSkin"), it is in Skins folder, so you can easily edit it as you want. For the other menus, the fields to add GUI Skins are already created, so you just have to create your custom GUI Skin and drop it on the field. Go on Resource folder, on MText object, you can add/change the GUI Skins:

- Add/change the GUI Skin of the main menu:



- Add/Change the GUI Skin of the waitroom menu:



- Add/Change the GUI Skin of the in-game menu (it have already a default GUI Skin, you can remove it if you want):



- Change the display from the OnGUI() and display functions. The functions which display something on the GUI begin all by the word "Display". The display and OnGUI() functions are called from the following scripts:
 - MLMenu, lines 128 to 487
 - MMenu, lines 79 to 147
 - MGameMenu, lines 46 to 290
 - MGameChat, lines 28 to 67
 - MGamePlayerList, lines 19 to 80
 - MWaitRoom, lines 78 to 261

For the main menu, the dedicated server menu and the waitRoom, you can also use the "MenuCamera" prefab to change the background color, as show on part 4.8.

6. Useful attributes

For finish, we will see some attributes defined in scripts, and which could be useful for your project. I tried to find all elements which could be useful for you, if you have other ideas, feel free to tell me in my Thread on Unity, so that I could complete this part of the documentation.

6.1. Get the current player data

All that data of the current player are save on the script MPlayerData, attached on the game object MPlayerData.

You can access to this script from any place in this way:

```
// Define the variable to store the script reference
public MPlayerData playerDataSrc;

// Assign the script reference to the variable
playerDataSrc = GameObject.Find("MPlayerData").GetComponent<MPlayerData>();

// You can now access to any data in this way:
playerDataSrc.userName; // Get the user name
```

In this way, you can access to the following data:

- **int id**: the id of the player in the database.
- **int gamelid**: the network index number of the player.
- **bool isLogin**: takes the value TRUE if the player is logged.
- **string userName**: the player name stored in the database.
- **string playerName**: the player name used with MultiLan - it doesn't depend of the database (it will have the value NULL if you don't use MultiLan).
- **string nameInGame**: the name used in the current game. This data takes the value of *userName* variable if you are on an online game, and it takes the value of *playerName* variable if you are on a LAN game (joined from MultiLan).
- **string mail**: the e-mail address of the player which is stored in database.
- **string privateIP**: the private IP of the player.
- **string publicIP**: the public IP of the player.
- **bool isInGame**: takes the value TRUE if the player is currently in game.
- **string loginKey**: the player's current login key. It's a token which is used as a session ID, it is use to authenticate the current player when he try to get data on the database across the web hosting.
- **bool isLan**: takes the value TRUE if the player is currently on a game joined from MultiLan.
- **bool isOnline**: takes the value TRUE if the player is currently logged from MultiOnline.

6.2. Get the players list

The player list of the current game's players is on MNetwork script, attached on the game object MNetwork. All the player have this list, the host/server holds the list updated, each time a player enters/exits the game, the host/server sends the new list to the others players.

You can access to this script from any place in this way:

```
// Define the variable to store the script reference
public MNetwork networkSrc;

// Assign the script reference to the variable
networkSrc = GameObject.Find("MNetwork").GetComponent<MNetwork>();

// You can now access to the player list in this way:
networkSrc.playerList;
```

This list is a [C# List](#)<MUser>. The type MUser have been create in order to store the game's players.

You can browse it using a for loop in this way:

```
for(int i = 0; i < networkSrc.playerList.Count; i++){
    // Instructions

    // Example: to get the player name:
    networkSrc.playerList[i].playerName;
}
```

You can access in this way the the following data:

- **int id:** the id of the player in the database.
- **int gameId:** the network index number of the player.
- **string privateIp:** the private IP of the player.
- **string publicIp:** the public IP of the player.
- **int playerPing:** the ping of the player - the ping are all made by the host / server
- **string playerName:** the name of the player.
- **int isPlayerInGame:** takes the value 1 if the player is currently on the game map and the value 0 if the player is in the waitroom.
- **int isGameHost:** takes the value 1 if this player is the host of the game.

6.3. Get current game data

The data of current game are on MNetwork script, attached on the game object MNetwork. These data are stored from a Object typeof MGame, this class was created to store all the useful data about a game.

We have seen before how access to this script from any place. Once you have your variable which contains a reference of the script MNetwork, you can get the game data in this way:

```
// You can now access to the game info in this way:
networkSrc.gameInfo;

// And, for example, you can get the game name:
networkSrc.gameInfo.name;
```

The MGame class contains the following data:

- **int id:** the id of the game in the database.
- **string name:** the name of the game.
- **string mapName:** the name of the game map.
- **int mapKey:** all the game maps are stores on an array (*string[]*) on MGameParameters (as we saw in part 4.1). This variable contains the key of the game map in this array.
- **int port:** the port of the game.
- **int totalPlayer:** the current total players of the game.
- **int maxPlayer:** the maximum players numbe of the game.
- **bool isUsePassword:** takes the value TRUE if the game uses a password.
- **string password:** the game password, if it exists.
- **bool isStarted:** takes the value TRUE if the game is started, that means, if the host has loaded the game map. In a dedicated server game, this variable has always the value TRUE.
- **string register:** the hour of creation the the game (format H:i)
- **string registerDate:** the full date of creation the the game (format Y-M-D H:i:s)
- **bool isOnline:** takes the value TRUE if the game was joined from MultiOnline, and is hosted by a player.
- **bool isPrivate:** takes the value TRUE if the game is privae (for MultiLan only).
- **bool isOnNetwork:** takes the value TRUE if the host is on the same network as ourself. That's useful because in this case, that mean that we have to join the game using the host's private IP instead his public IP.
- **bool isOnDedicatedServer:** takes the value TRUE if the game is hosted by a dedicated server.
- **int hostId:** the id of the host in the database.
- **string hostName:** the name of the host.
- **string hostPrivateIp:** the private IP of the host.
- **string hostPublicIp:** the public IP of the host.
- **string hostLocation:** the location of the host - only if the host is a server.
- **int ping:** the game ping for MultiLan.
- **Ping OPing:** the game ping for MultiOnline (the game ping for MultiOnline (that's two different variable because that uses two differents type)).

6.4. Get the games list

The game list is on MLMenu script, attached on the game object MLMenu.

You can access to this script from the Menu scene, in this way:

```
// Define the variable to store the script reference
public MLMenu menuSrc;

// Assign the script reference to the variable
menuSrc = GameObject.Find("MLMenu").GetComponent<MLMenu>();

// You can now access to the game list in this way:
menuSrc.gameList;
```

This list is a [C# List](#)<MGame>. We saw just before which informations we can get from this class.

You can browse the list using a for loop in this way:

```
for(int i = 0; i < menuSrc.gameList.Count; i++){
    // Instructions

    // Example: to get the game name:
    menuSrc.gameList [i].name;
}
```

6.5. Get the position of a player

If you checked the option "Display player name", a list of the position of all the current players in game is hold update. This list is on MNetwork script, attached on the game object MNetwork.

We have see before how access to this script from any place. Once you have your variable which contains a reference of the script MNetwork, you can get the game data in this way:

```
// You can now access to the position list in this way:
networkSrc.positionList;
```

This list is a [C# List](#)< MPlayersPosition>. The type MPlayersPosition have been create in order to store the player positions. This class is declared on MUser, line 170.

You can browse the position list using a for loop in this way:

```
for(int i = 0; i < networkSrc.positionList.Count; i++){
    // Instructions

    // Example: to get the player position:
    playerDataSrc.positionList[i].position;
}
```

You can access in this way the the following data:

- **int id:** the id of the user in the database.
- **string name:** the name of the player.
- **Vector3 position:** the position of the player (as Vector3).

In case you would like use this position list without use the option " Display player name", you just have to change the line 78 on MNetwork.

Change this line:

```
if(playerDataSrc.isInGame && !isGameServerRebuild &&
parameters.displayPlayerName) {
```

By this one:

```
if(playerDataSrc.isInGame && !isGameServerRebuild) {
```

7. FAQ

7.1. The network games list doesn't appear

The network games list works by finding all the active computers of your network for search if they have created a game. This function works only on Windows computers because for find the active computers of your network the script use the Windows function NetServerEnum(), which use the Netapi32 DLL.

Note : if the network games list doesn't appears on your configuration, you can always join a game with the host's IP. If the host is on your network, enter his private IP, else, enter his public IP.

7.2. Players cannot join to my public game

If you create a public game, don't forget to properly open the game's ports on your server, else, nobody (except the people in your network) can connect to your game.

Check also your antivirus and firewall who will probably send you a warning message when you'll create a game. So, authorize your application on your antivirus and firewall parameters, for private and public communications.

7.3. The host migration doesn't work

When the host leaves the game, all the guest lose the connexion. The script will then try to find a new host among the remaining players in game. If a player have opened the game's port on his server, he'll become the new host, and all the other player will connect on him. But if neither player have his port open, nobody can become the new host, and the game ends.

Note 1 : the privates games doesn't require to open ports on the server, and host migration works always very well since everybody can became host with no special condition.

Note 2 : when the game is already started, the new host must be in game. Players in the waiting room cannot become new host when if game has already been started.

7.4. The game is lagging

If you have some lag problems, you can try to change the Sendrate, as explain on part 4.1. You can also change the field "Observed" on the player's NetworkView component : the "Observed" field is currently "Player(Transform)", you can try to replace it by "Player(Rigidbody)". For that, you just have to go on your player prefab, and drop his "Rigidbody" component on the field "Observed" of the NetworkView component.

7.5. My animations are not synchronized

When you use animations across a network game, you must call the animations from a RPC function, else, the animation will not be played for the other players.

Example in C# for one animation :

```
// Call this function when the event which triggers your animation happen
void RunAnimation() {
    // Call the RPC function SyncAnimation on all players
    networkView.RPC("SyncAnimation", RPCMode.All);
}

[RPC]
void SyncAnimation() {
    // Play the animation
    animation.Play("the name of your animation");
}
```

Example in C# for many animation :

```
/* Call this function when the event which triggers your first animation
happen */
void RunAnimation1() {
    // Call the RPC function SyncAnimation on all players with parameter 0
    networkView.RPC("SyncAnimation ", RPCMode.All, 0);
}

/* Call this function when the event which triggers your second animation
happen */
void RunAnimation2() {
    // Call the RPC function SyncAnimation on all players with parameter 1
    networkView.RPC("SyncAnimation ", RPCMode.All, 1);
}

[RPC]
void SyncAnimation(int index) {
    if(index == 0){ // If the index parameter is 0 :
        // Play this animation :
        animation.Play("the name of your first animation");
    } else if(index == 1){ // else, if the index parameter is 1 :
        // Play this animation :
        animation.Play("the name of your second animation");
    }
    /* And you can add here as much animation your want by using other
    index numbers */
}
```

Note : on the examples, you can replace *animation.Play()* , by *animation.CrossFade()* if you want to use this function.

Note 2: I made a full tutorial about how changing the player and synchronize animations accross network, here it is: https://www.youtube.com/watch?v=obB8gj_iXoA