MiR Project Updates Documentation

Real time data for WebGL application.

Useful information for web requests in Unity.

web requests are possible on Unity only and exclusively with the class they provide us 'UnityWebRequest', other external classes or libraries could cause problems in builds.
You can find examples on how to use it here.

Usually the server responds with a Json, in order to convert the Json in a programming object we have to deserialize it, but before we must define the class that represents the data we are going to receive.
In order to create that class I suggest you 2 ways:

1. Copy the json response from the server (or from a sample response usually findable in the Swagger, in your case here)
2. Using this service

And then paste the class in a C# script.
Replace the 'RootObject' class name with the name you want (better if equals to file name, for consistency).

Unity wants that every class is set as public and Serializable in order to be able to serialize/deserialize it, more info here.

Only public fields are serializable/deserializable.
So, we must remove all the getters and setters, if any.

To Serialize objects to json, or to deserialize json to get objects, with Unity the class recommended is 'JsonUtility'.
Here some instructions on how to use it.

Useful information for Navigation Mesh in Unity.

In order to get smoother navigation in Unity, we could use the AI solution Unity provides (AI Navigation).

Here is a good video to start with. I suggest to watch it before go on.

Basically you have to download the AI Navigation package, create a NavMeshSurface, after setting your environment (floor, wall, and non-movable objects) as static, you can create your agent type (I set up for you a MiR Agent) and bake the surface.

I added to the Scene 'NavMeshMovement' to the player the script 'NavMeshAgent' and set everything as the MiR agent created before.

I set to the agent an acceleration of 120 in order to simulate a linear speed as much similar as possible.

I created a script called 'PlayerMir_NavMesh' which is similar to 'PlayerMir' but using the NavMesh for the movement.

And 'Quaternion.Slerp' for the rotation as requested.

Web Sockets connections are blocked on WebGL by Unity Engine,

in order to solve this issue, there are 3 ways.

Solution 1:

This solution is entirely developed in the Unity3D Engine.

The first solution is based on continuous requests to the server,

web requests are possible on Unity only and exclusively with the class they provide us 'UnityWebRequest', other external classes or libraries could cause problems in builds.
You can find examples on how to use it here.

Usually the server responds with a Json, in order to convert the Json in a programming object we have to deserialize it, but before we must define the class that represents the data we are going to receive.
In order to create that class I suggest you 2 ways:

3. Copy the json response from the server (or from a sample response usually findable in the Swagger, in your case here)
4. Using this service

And then paste the class in a C# script.
Replace the 'RootObject' class name with the name you want (better if equals to file name, for consistency).

Unity wants that every class is set as public and Serializable in order to be able to serialize/deserialize it, more info here.

Only public fields are serializable/deserializable.
So, we must remove all the getters and setters, if any.

To Serialize objects to json, or to deserialize json in order to get objects, with Unity the class recommended is 'JsonUtility'.
Here some instructions on how to use it.

This solution has been provided with the Unity Project.

Solution 2:

This solution provides a way to use WebSockets/MQTT real time data in a webGL application.

Avoiding the polling to the server. This could improve the performance of the application, especially if the application manages more devices, using more APIs endpoints.

This solution consists in creating C# Scripts as usually in the Unity side, assign the scripts to GameObjects and create public functions (non-static), which just care about taking parameters and use them (in our case could be the "MoveMir" function).

Once done that, we can build our WebGL (but before, go to Project Settings -> Player -> Publishing Settings and set Compression Format to 'disabled).

Now we can create a 'Next.js' web application.

(To use Next.js, is necessary that you have some knowledge of React.js, but on the website you can find also very nice tutorials on how to start with Next.js)

Considering that Next.js is a javascript framework based on React, here we can use React packages.

So in our Next.js application we can install the following package react-unity-webgl in order to embed our Unity WebGL Build inside a dynamic web application.

Thank to the 'sendMessage' function the library provide us, from the web client application we can call unity functions.

How all this is useful for web sockets?

Because on the web application side, we can easily get real time data from a web socket service (also MQTT), and every time a message arrives from the server to our application, we can call a Unity Function from 'outside'.

The Idea is to set up a mqtt and/or websocket channel in order to get MiR real time data, and call the 'MoveMir' function we previously created.

This solution has been provided with a next.js template.

Solution 3:

(NOT PROVIDED), just for info.

You could set up a Unity JS Plugin, suggested by the answer of this question.

I would suggest you to start from this repository.

(currently is not working, it is not maintained anymore, but I think is a good starting point.)