Author: Thibault Helle

Unity3D Documentation

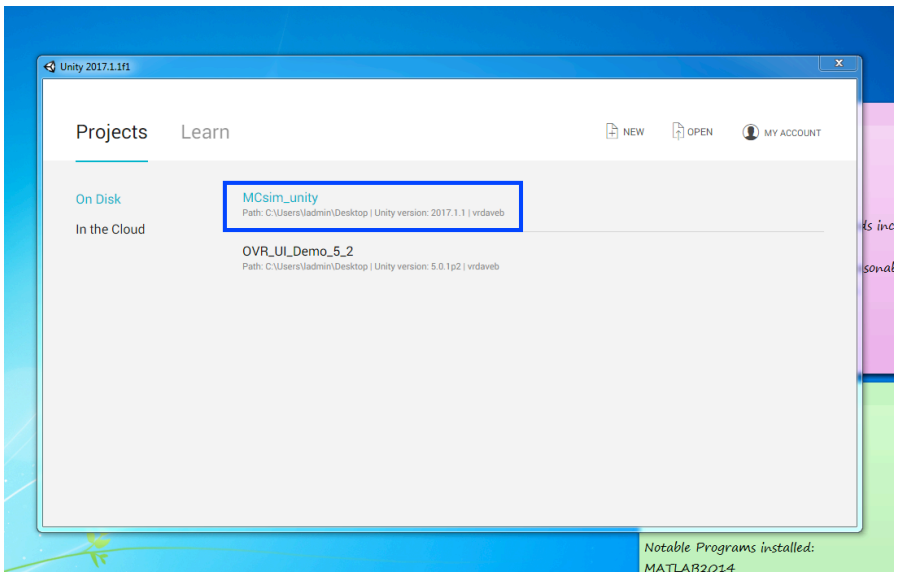
Links to Basic unity tutorial

Unity Basics: https://www.youtube.com/watch?v=UWXVbZgXNA4
C# scripting in unity: https://www.youtube.com/watch?v=tzq-DJeNIrU&t=525s

Starting Unity

1. Open the **unity.exe** application by clicking the symbol like in the image below.

2. Open the project **MCsim_unity:**



3. The scene (**race_track_lane)** should now be opened and you will see a window with all the *GameObjects* at the left pane and the game view in the middle. **Press** the **Play button** to **start** the game. Make sure that the VR-headset is **plugged in** with USB-3.0 and the HDMI-cable. OBS! Do not start the game before the raspberry pi's ROSCORE server has started!

Author: Thibault Helle

4. To stop the game, just press the **Stop button** at the same place.

Edit motorcycle behavior in C#

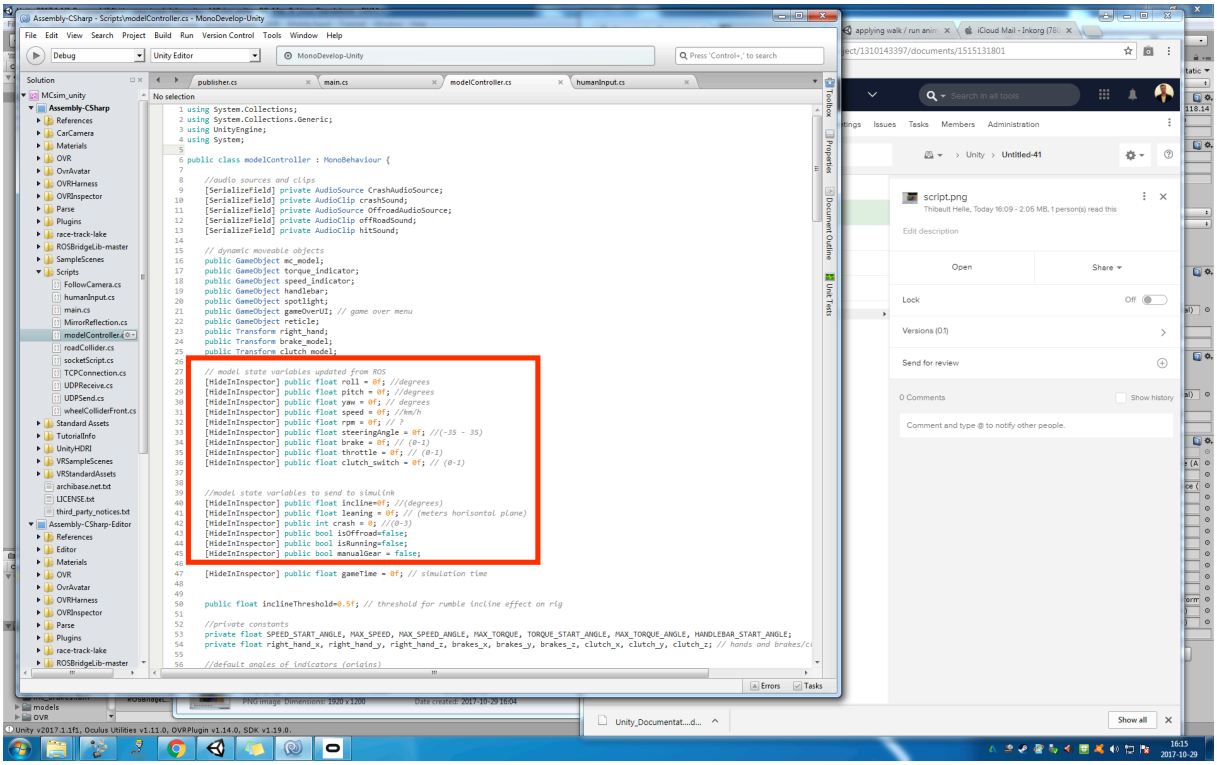
All C#-scripts must be attached to a GameObject, i.e. the "**Motorcycle**" game-object has the most important script called **modelController.cs.** This is where most of the dynamics happens and where the public state variables of the model are (x, y, z, rotations, throttle, onRoad, speed etc…) and can be accessed from other scripts as they are public. Edit the script by **right-clicking** on the **attached** script's in the Game Object's **inspector**.

Author: Thibault Helle

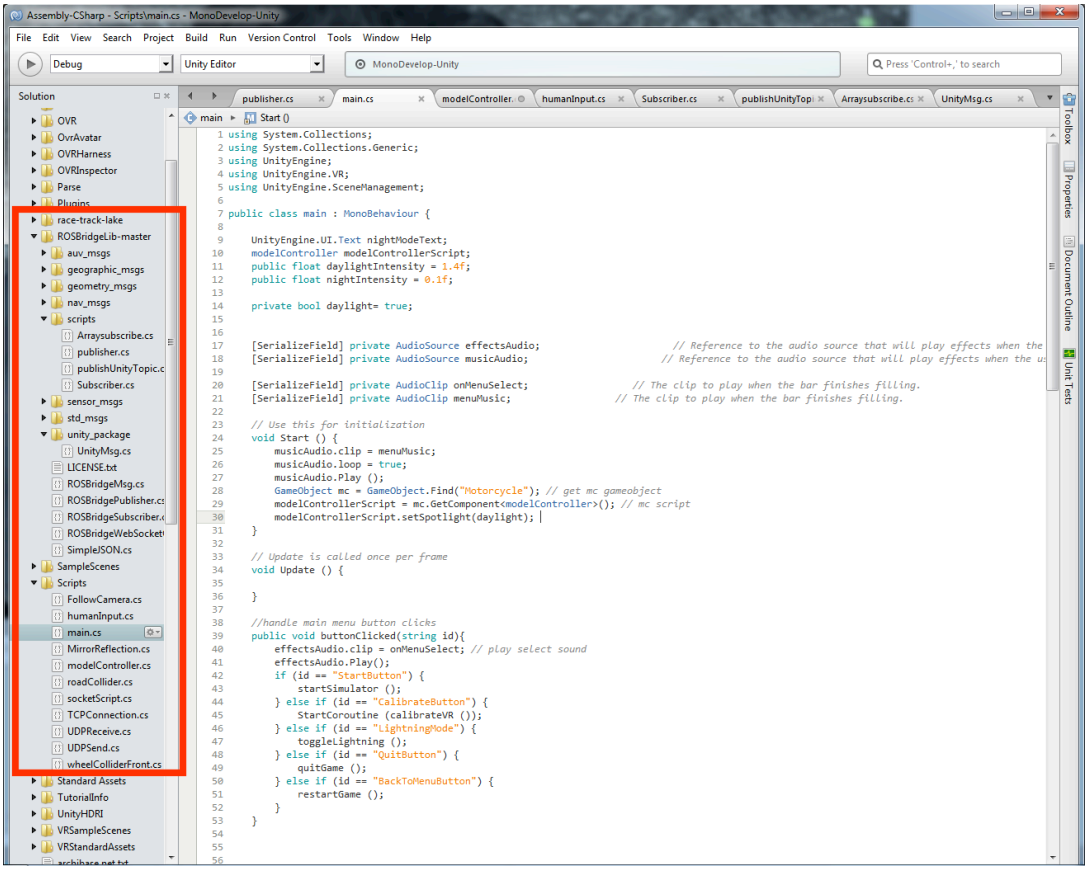
Another application, **MonoDevelop** will open where you can open and debug the code live, even when the game is running. In the **modelController.cs** as shown below, you can find all the variables that is important.

Variables needs to be **public** for it to be read/written by the ROS scripts (later in this guide). If you don't put [HideInInspector] in front of the code, you will be able to dynamically edit the constant value in the GUI inspector in Unity. The void **Start**() function is the first function to be started when the game starts, and **Update**() is run every frame update, if the script is enabled (checkbox in inspector).

Author: Thibault Helle



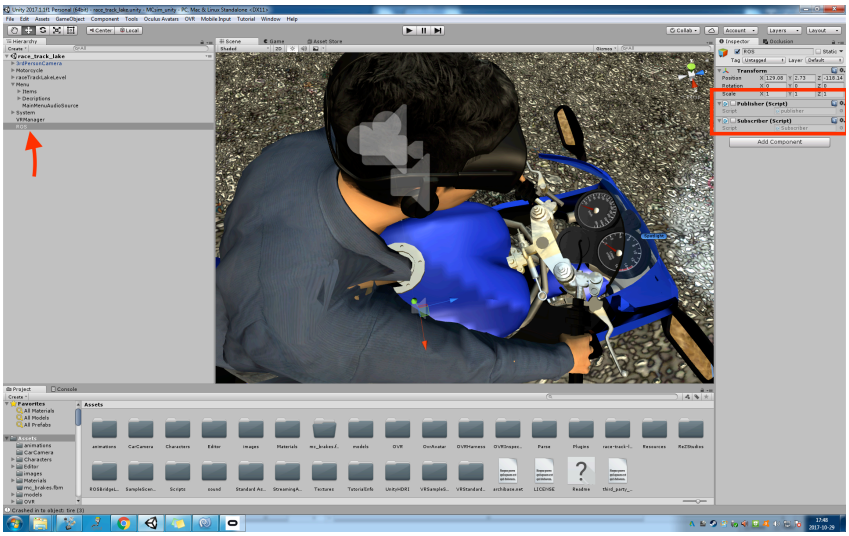
You can find the other script here in the folders as shown in the image below. The main.cs script handles the main events, such as pressing button at the menu, playing music, hiding menu etc.



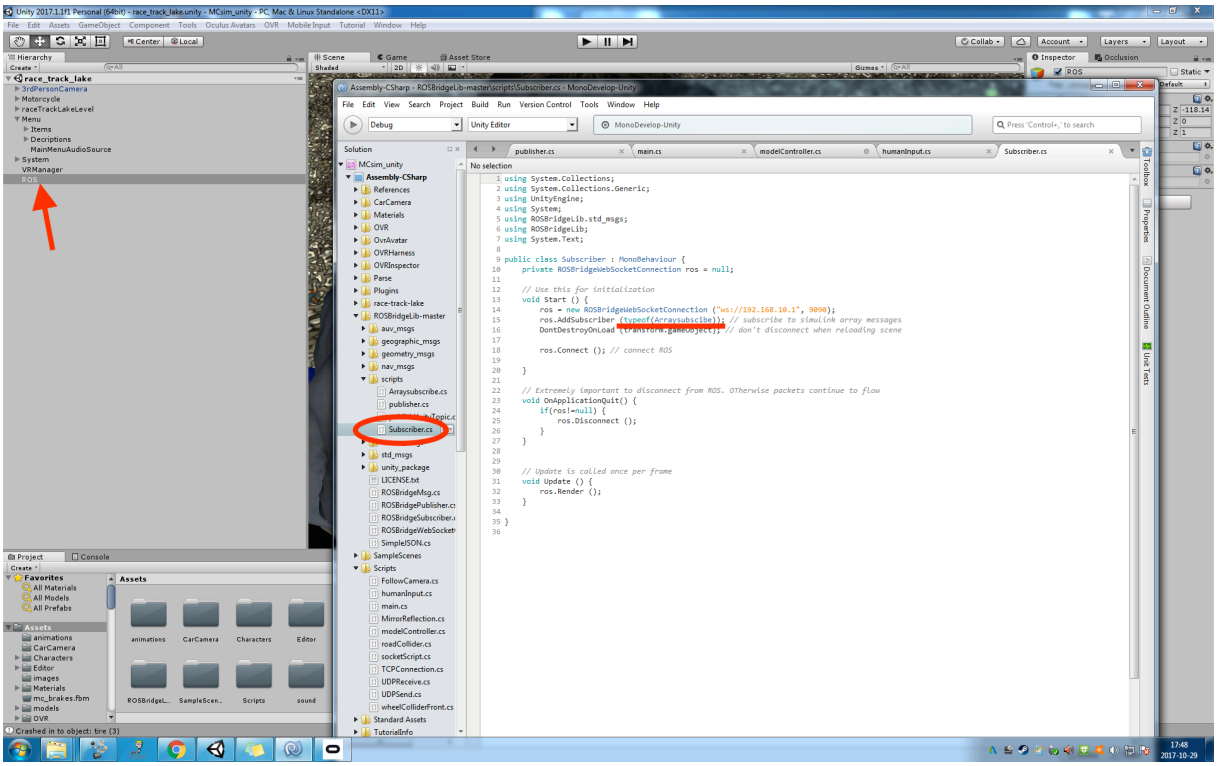
Author: Thibault Helle

Edit incoming/outgoing data

Click on the **ROS** gameobject to see the attached ROS scripts like in the image below.

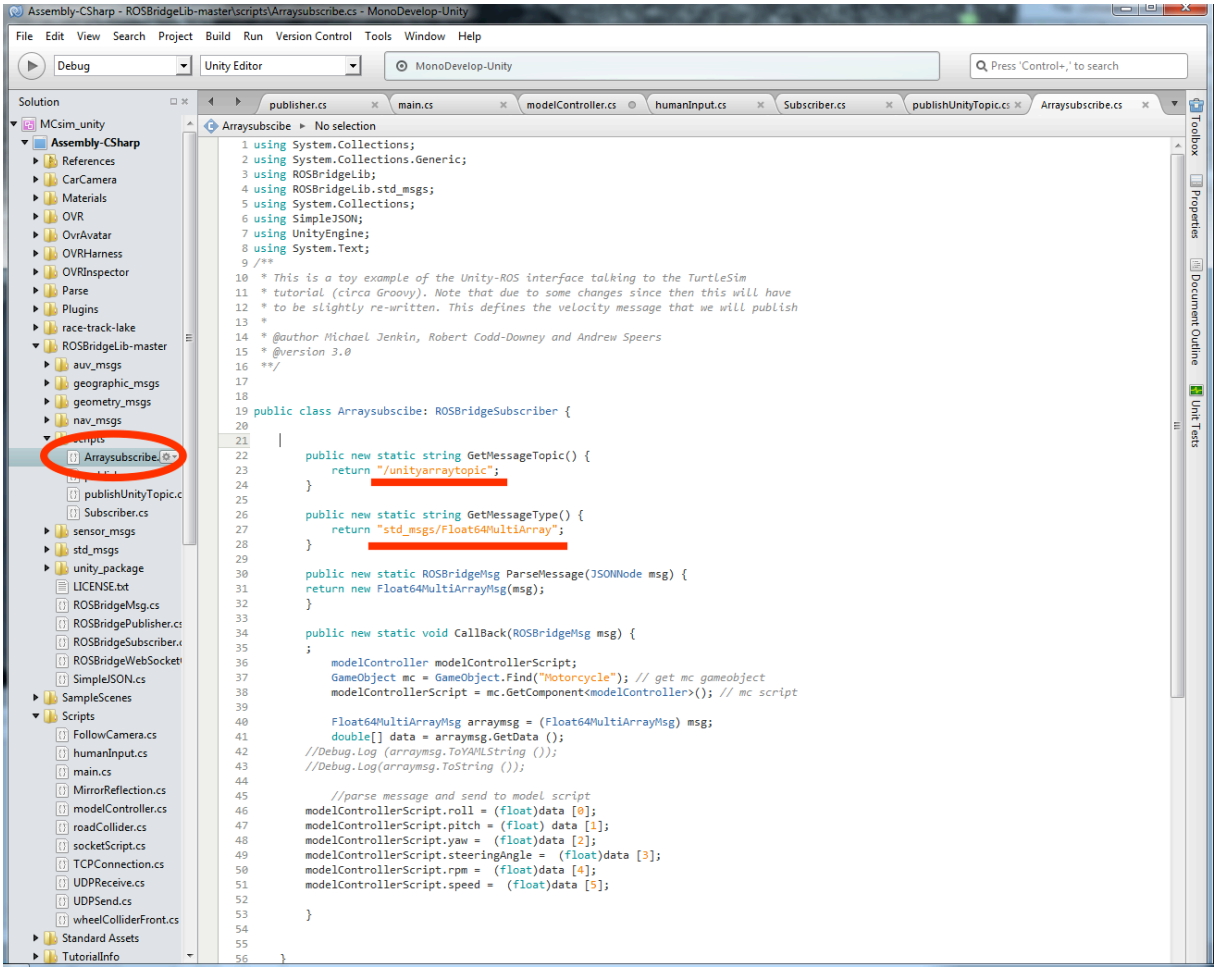


In the **Subscriber.cs** script we can add topics to be subscribed to, and of what type of message it is. Like in the image below, we are adding a subscription to **Arraysubscribe,** which is a class type.



To see or edit the type, open up the script of the same name, Arraysubscribe.cs. In the image below, you can see the topic name "**/unityarraytopic**" and the message type "**std_msgs/Float64MultiArray**", which is a standard ROS message type.

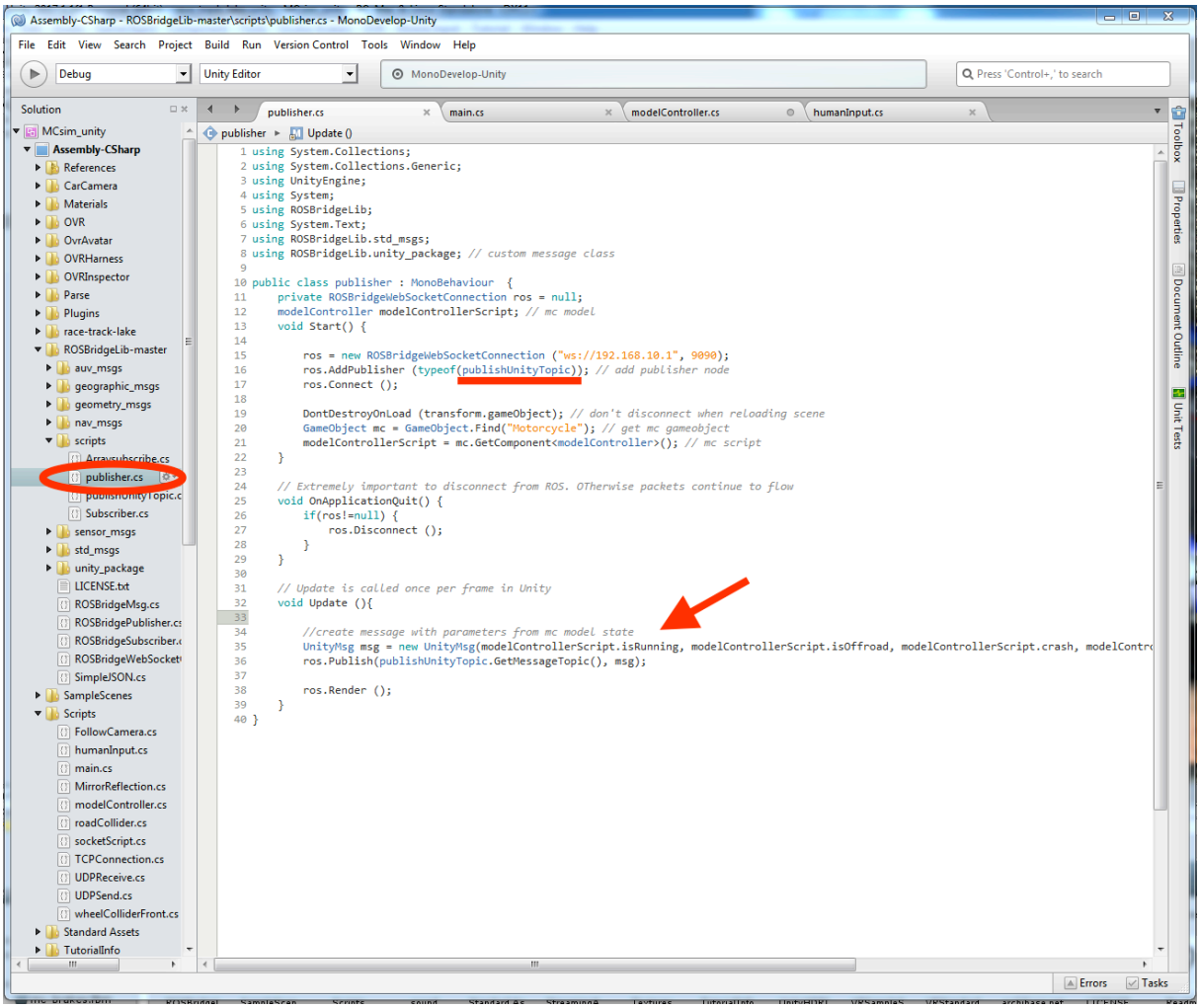
Author: Thibault Helle



We then update the GameObject “Motorcycle” state variables to by accessing the script as seen above in the code. Where all data from the incoming messages are passed to the corresponding variable to the model script.

Author: Thibault Helle

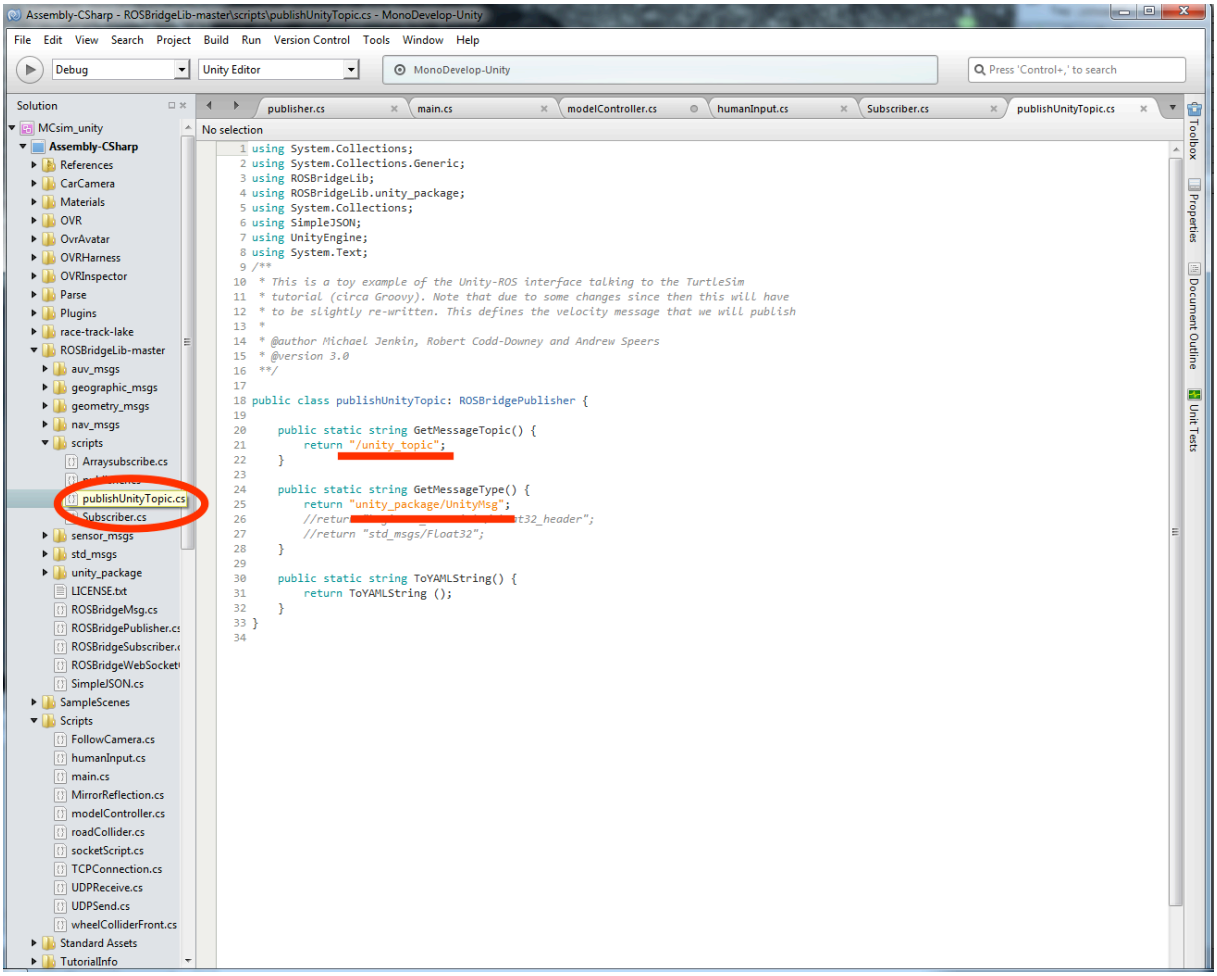
Now for publishing a message using ROS, we use the **publisher.cs** script to add publishers with a certain type. In this script below we have a topic type of **publishUnityTopic**. In the Update() function you can edit what message type the topic needs and pass what variables to send from the running modelcontroller script.



Here (above image) we send a custom made message with variable names (instated of an array) which is easier to handle. To see the message class declaration and to add/remove data fields, open **UnityMsg.cs** script in the **unity_package** folder.

Author: Thibault Helle

Now to see the topic declaration type, open **publishUnityTopic.cs**. Here you define what message type that will be sent.



If you have other questions on how to add/edit things just ask me! ☺