How to use the OSC communication system within Unity V1, September 27, 2015

Communicating with Unity scripts can be done in various ways. Here we discuss communication with a Processing app. Often you can use Processing for handling interaction with devices like Arduino's etcetera. You can then use the processing app as an "intermediate" for sending such data to Unity scripts.
The solution can be used for sending as well as receiving data to/from Unity. It is based on the OSC protocol. Note that we assume here that although in this manual we aim at connecting Unity and Processing, the same approach can be used for connecting Unity to other applications, provided the OSC protocol is supported. For the case of Processing we rely on an existing library called "oscP5"

Setting up.

1. The first step is to include the OSC scripts in your Unity project. Simply unzip the OSC.zip directory, and place it into the "Assets" folder from your Unity project.
2. In Processing, use the "Import Library" from the Sketch menu, and import the oscP5 library.

Sending data from Processing to Unity.

Inside Unity, you create scripts and attach such scripts as "Component", to some Unity Game object from the hierarchy. Typically some Game Object that you want to control via OSC. In our example we assume that you would like to control the position of the Game Object, via mouse movements in your Processing app.

1. See the C# file "OSCExample1".
   Put it inside the "scripts" subdirectory of "Assets".
2. Then introduce some Unity 3D Game Object, and
3. Add "OSCExample1" as a Component, using the "Inspector" panel for your 3D object.
4. On the Processing side, open the Processing sketch "ProcessingOscExample1"
5. Run both the Processing script, and the Unity level. If you click the mouse in the Processing panel, your Unity object should move along. Also have a look at the Unity Console.
6. In Unity you also have an extra "Window" called "OSC Helper", that you can use to watch communications. Especially useful if you are debugging your application.

In a nutshell:

- Inside the "Start" method, Call OSC.ReceiverPort() for the port you want to use for your OSC channel. This must match the same channel on the Processing side, obviously.

- Also in “Start” use OSC.Onreceive() to register your “handler” methods. Call it once for every handler. Note that you can specify different handler methods for different OSC addresses. So, for example, in the example, we only used OSC address “processing/mouse/”, but it is easy to use other addresses for other types of events.
- The List<OSCArgument> data parameter is a C# List:
    - Data.Count is the length, i.e. the number of data elements
    - Data[0], data[1] etcetera use “array subscript” notation for accessing individual list elements.
    - A single data element has type OSCArgument. This is either an int, a float, or a string.
- Define your handler methods. They can have any name, but must have the typing:
  public void MyHandlerXYZ(List<OSCArgument> data) {….}

Sending data from Unity to Processing

TBD. See OSCExample2.

Dealing with multiple remote applications:

See OSCExample3. It is possible to receive via various ports, and it is possible to send to different IP addresses and/or ports.

The example relies on two variations of the `Processing script, using different posts, that are run simultaneously. Seems to work on the Mac. (Not on Windows, because one of the processing apps does not react)