Tutorial: Running Bluetooth with Unity (PC and Android)

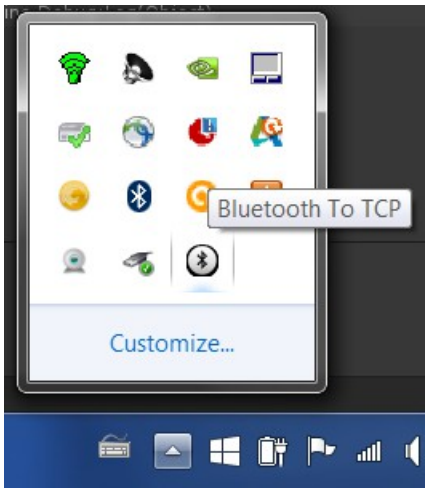
requirements:
PC with built-in Bluetooth card or Bluetooth dongle
An Android phone with Bluetooth

Step 1 - Set up Bluetooth on devices
Confirm that Bluetooth is working properly on the PC and on the phone.

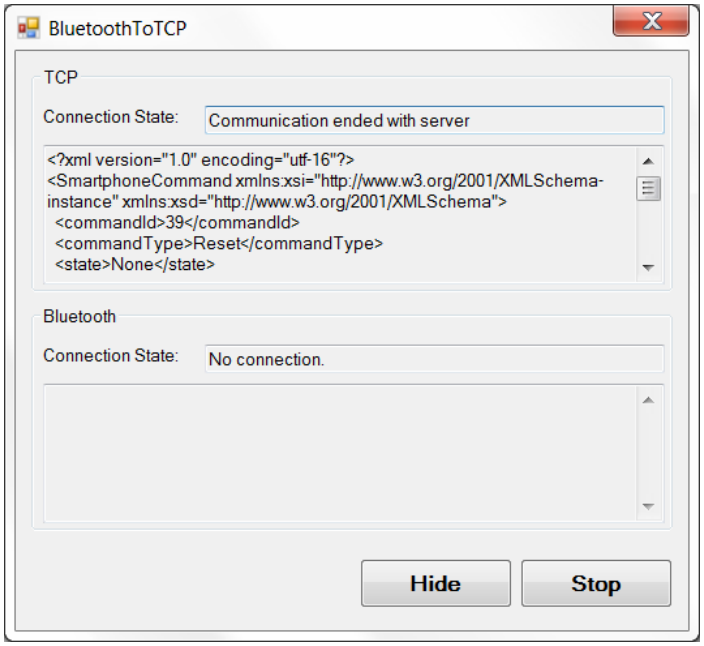
Step 2 - Pair the phone and the PC
On the Android phone, add the PC to the paired device list.
The PC should also be the first device (and possibly the only one).
The included code does not contain any device search: it directly uses the first connected device.

Step 3 - Import the package for PC
In the Unity project, import "PC \ BluetoothForPC.unitypackage" (Assets / Import Package / Custom Package ...).
Add the Prefab "Assets / Scripts / Smartphone / Smartphone" to the stage.
In this gameobject, check the debug for Smartphone External Tool.

Step 4 - Test on PC
Launch the scene and, if necessary, accept that the BluetoothToTCP application accesses the network.
The console should announce that a server (TCP) has been launched at port 13000 and that the BluetoothToTCP application has been launched.
Check that BluetoothToTCP is in the taskbar:



Double-click on the icon to open the application debugger:



This application aims to make a "man-in-the-middle" between the TCP server used in the Unity code and the Bluetooth client of the Android phone. The state of the connections is indicated, in addition to the last transmission received on each side.

This application does not have to be closed between each start of the Unity program: the code itself checks whether the application is running and launches it as needed. BluetoothToTCP supports disconnections /

reconnections, ie the client (phone) or the server (Unity) can be restarted and the connection will be recovered.

> BluetoothToTCP can still be stopped with the "Stop" button or with a right-click on the icon in the taskbar.

Step 5 - Import the package on Android
In the Unity project, import "Android \ BluetoothForAndroid.unitypackage" (Assets / Import Package / Custom Package ...).
Add the Prefab "Assets / Scripts / PC / PC" to the scene.
In this gameobject, check the debug for Command Handler.

Step 6 - Test on Android
Compile the project and launch it on an Android phone.
Launch the PC application and display the BluetoothToTCP debugger.
The connection should be between the devices and a "Reset" command should be received and returned by the PC.
Both projects are now ready to exchange information and orders.

Use :

On PC, a command can be sent to the phone with SmartphoneExternalTool.sendCommand (...). To be informed when you receive an order, simply use SmartphoneExternalTool.registerOnCommand with parameters of the order of interest and a callback.
On Android, the operation for sending commands is the same with CommandHandler, but there are no callbacks for receiving orders. Instead, the last received is kept in currentState and the class is responsible for updating the rest of the system, for example with changeMenu ();
On both platforms, the SmartphoneCommand class is shared. The file should be copied from one project to another at each modification. The class contains a list of all possible commands, states, and modes for the system and is used to store all the information in a command. It is directly this class which is encoded when sending a command and it is therefore SmartphoneCommand objects that must be manipulated. The class can also

be used to define the structure of a menu on the phone, with populateListOfCommands ().

Notes

The reason we need a "man-in-the-middle" to convert TCP to Bluetooth and vice versa is that Mono C #, as compiled by Unity, does not support Bluetooth. It is therefore necessary to make use of BluetoothToTCP (written in C # .NET) on PC and part of Java native code on Android: Note that since the PC application only interprets TCP, it is possible to remove the man-in-the-middle and specify on the phone that it is desired to use TCP (PC / DataClient / TransmissionType) for communication WiFi instead, if both devices are on the same network. There is also no need to specify an IP address since the code searches the network. This approach is sometimes unreliable and the use of Wifi causes latency and loss of connection. Because of this, Bluetooth is recommended.

Other projects:

"PC \ BluetoothToTCP" contains the Visual Studio 2012 project solution for compiling the "man-in-the-middle". If you want to modify the code, the compiled executable must be placed in "Assets \ StreamingAssets \ BluetoothToTCP" with the required libraries:
Dolphins.Salaam.dll
InTheHand.Net.Personal.dll
"Android \ BluetoothPlugin" contains the Eclipse project to compile the native Java code for communicating with Bluetooth in a Unity application. The .jar produced by the code must be placed, with the Manifest, in the folder "Assets \ Plugins \ Android".

References : Dolphins Salaam (scanning a network for IP search): https://salaam.codeplex.com/ 32feet.NET (Bluetooth for C #): In The Hand Ltd. & Alan J. McFarlane The rest of the code, including BluetoothToTCP and BluetoothPlugin,

except that specified in the comments of the code: Alexandre Millette This tutorial: Alexandre Millette