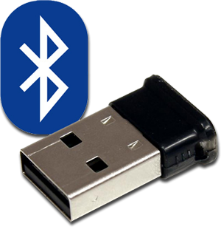
Tutoriel : Faire fonctionner Bluetooth avec Unity (PC et Android)

**Requis :**

* Un PC muni d’une carte Bluetooth intégrée ou d’un « Bluetooth dongle »
* Un téléphone Android avec Bluetooth

**Étape 1 – Configurer Bluetooth sur les dispositifs**

Valider que Bluetooth fonctionne correctement sur le PC et sur le téléphone.

**Étape 2 – Appareiller le téléphone et le PC**

Sur le téléphone Android, ajoutez le PC dans la liste de dispositifs appareillés.

Le PC devrait aussi être le premier dispositif (et possiblement le seul).

Le code inclus ne contient aucune recherche de dispositifs : il utilise directement le premier dispositif connecté.

**Étape 3 – Importer le package pour PC**

Dans le projet Unity, importez « PC\BluetoothForPC.unitypackage » (Assets/Import Package/Custom Package…).

Ajoutez le Prefab « Assets/Scripts/Smartphone/Smartphone » à la scène.

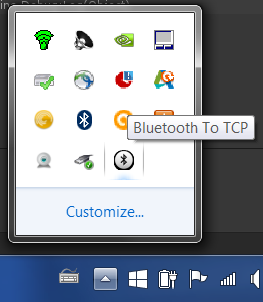
Dans ce gameobject, cochez le debug pour Smartphone External Tool.

**Étape 4 – Tester sur PC**

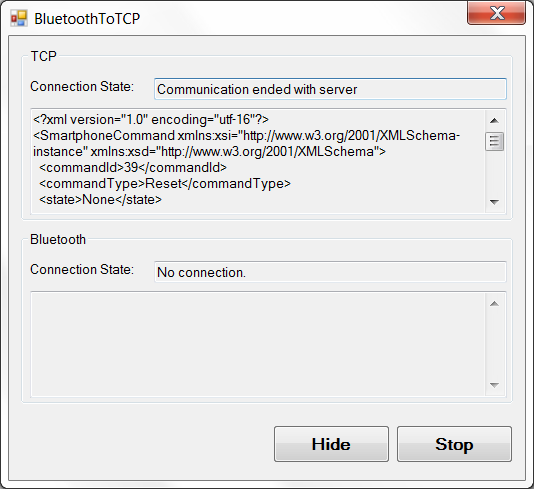
Lancez la scène et, au besoin, acceptez que l’application BluetoothToTCP accède au réseau.

La console devrait annoncer qu’un serveur (TCP) a été lancé au port 13000 et que l’application BluetoothToTCP a été lancée.

Vérifiez que BluetoothToTCP est bien dans la barre des tâches :



Double-cliquez sur l’icône pour ouvrir le debugger de l’application :



Cette application a pour but de faire un « man-in-the-middle » entre le serveur TCP utilisé dans le code Unity et le client Bluetooth du téléphone Android. L’état des connexions est indiqué, en plus de la dernière transmission reçue de chaque côté.

Cette application n’a pas à être fermée entre chaque démarrage du programme Unity : le code vérifie lui-même si l’application roule et la lance au besoin. BluetoothToTCP supporte les déconnexions/reconnexions, c’est-à-dire que le client (téléphone) ou le serveur (Unity) peut être relancé et la connexion sera récupérée.

BluetoothToTCP peut tout de même être arrêté avec le bouton « Stop » ou avec un clic-droit sur l’icône dans la barre des tâches.

**Étape 5 – Importer le package sur Android**

Dans le projet Unity, importez « Android\BluetoothForAndroid.unitypackage » (Assets/Import Package/Custom Package…).

Ajoutez le Prefab « Assets/Scripts/PC/PC » à la scène.

Dans ce gameobject, cochez le debug pour Command Handler.

**Étape 6 – Tester sur Android**

Compilez le projet et lancez-le sur un téléphone Android.

Lancez-aussi l’application PC et affichez le debugger de BluetoothToTCP.

La connexion devrait se faire entre les dispositifs et une commande « Reset » devrait être reçue et renvoyée par le PC.

Les deux projets sont maintenant prêts à échanger des informations et des commandes.

**Usage :**

Sur PC, une commande peut être envoyée au téléphone avec SmartphoneExternalTool.sendCommand(…). Pour être informé lors de la réception d’une commande, il suffit d’utiliser SmartphoneExternalTool.registerOnCommand avec en paramètres la commande d’intérêt et un callback.

Sur Android, le fonctionnement pour l’envoi de commandes est le même avec CommandHandler, mais il n’y a pas de callbacks pour la réception de commandes. À la place, la dernière reçue est conservée dans *currentState* et c’est la classe qui se charge de mettre à jour le reste du système, par exemple avec changeMenu();

Sur les deux plateformes, la classe SmartphoneCommand est partagée. Le fichier devrait être copié d’un projet à l’autre lors de chaque modification. La classe contient la liste de toutes les commandes, états et modes possibles pour le système et est utilisée pour stocker toutes les informations d’une commande. C’est directement cette classe qui est encodée lors de l’envoi d’une commande et c’est donc des objets SmartphoneCommand qu’il faut manipuler. La classe peut aussi être utilisée pour définir la structure d’un menu sur le téléphone, avec populateListOfCommands().

**Notes :**

La raison pour laquelle on a besoin d’un « man-in-the-middle » pour convertir du TCP en Bluetooth et vice versa est que Mono C#, tel que compilé par Unity, ne supporte pas Bluetooth. Il faut donc faire usage de BluetoothToTCP (écrit en C# .NET) sur PC et d’une partie de code natif Java sur Android :



À noter que comme l’application PC n’interprète que du TCP, il est possible de retirer le man-in-the-middle et de spécifier au téléphone qu’on désire utiliser TCP (PC/DataClient/TransmissionType) pour que la communication se fasse par Wifi à la place, si les deux dispositifs sont sur le même réseau. Il n’y a pas non plus besoin de spécifier une adresse IP puisque le code effectue une recherche sur le réseau. Cette approche est parfois peu fiable et l’utilisation du Wifi cause de la latence et des pertes de connexion. De ce fait, Bluetooth est recommandé.

**Projets autres :**

« PC\**BluetoothToTCP** » contient la solution du projet Visual Studio 2012 pour compiler le « man-in-the-middle ». Si on désire modifier le code, l’exécutable compilé doit être placé dans « Assets\StreamingAssets\BluetoothToTCP » avec les librairies requises :

* Dolphins.Salaam.dll
* InTheHand.Net.Personal.dll

« Android\**BluetoothPlugin** » contient le projet Eclipse pour compiler le code natif Java permettant de communiquer avec Bluetooth dans une application Unity. Le .jar produit par le code doit être placé, avec le Manifest, dans le dossier « Assets\Plugins\Android ».

**Références :**

* Dolphins Salaam (balayage d’un réseau pour recherche d’IP) : <https://salaam.codeplex.com/>
* 32feet.NET (Bluetooth pour C#) : In The Hand Ltd. & Alan J. McFarlane
* Le reste du code, incluant BluetoothToTCP et BluetoothPlugin, sauf que précisé dans les commentaires du code : Alexandre Millette
* Ce tutoriel : Alexandre Millette