Bachelor of Science HE-ARC in Engineering
Espace de l'Europe 11
CH-2000 Neuchâtel

SENSO GLOVE - DETAILS

Fait par :
Gabriel Griesser

Neuchâtel, HE-ARC//Bachelor, 24 juillet 2019

Table des matières

Chapitre 1

Introduction

Ce document décrit, pour le gant SensoGlove :

— Sa description

— Ses propriétés et caractéristiques

— L'installation, la connexion et la calibration du gant

— La description du SDK et son architecture

— Le mode d'emploi d'intégration de deux gants

— Les remarques et problèmes connus

Chapitre 2

Description

La paire de gants Senso est un dispositif haptique, développé par la firme Senso Device Inc. Les Senso, présentés sous forme de gants traditionnels, sont reliés au PC par une connexion sans fil. Cette connexion est mise en place grâce la clé logicielle fournie avec les gants. Les gants ne dépendent donc d'aucun plugin tier, exceptées les applications client et serveur nécessaires pour la connexion des gants Senso au PC.

Ces gants haptiques, pensés pour la réalité virtuelle et augmentée, permettent un suivi précis des mains et des doigts grâce à 7 capteurs IMU. Les Senso possèdent un moteur de vibrations, placé sur chaque doigt, rendant ainsi l'immersion plus réelle. Aucun système de retour de force n'est disponible ici.

Les fonctionnalités haptiques du gant Senso sont limitées à retour tactile, caractérisé par un moteur de vibrations, disposé au bout de chaque doigt. De ce fait, le Senso ne peut reproduire ni la forme, ni la taille ni la rigidité de l'objet avec lequel il interagit. Par contre, le système de vibration du gant permet de simuler à la fois la rugosité de l'objet, en variant la puissance de la vibration, à la fois les actions tactiles comme le clic sur un bouton ou le ressenti d'une explosion.

Chapitre 3

Propriétés et caractéristiques

- **Type**
  - Gants traditionnels
  - Textile élastique
- **Connexion**
  - Sans fil
  - Connexion au PC par le biais de clé bluetooth (adaptateur BLE)
  - Un adaptateur BLE par gant
  - Drivers à installer
- **Retour de force**
  - Non
- **Retour tactile**
  - Oui, avec un moteur de vibration pour chaque doigt
- **Suivi de mouvement (tracking)**
  - Main : rotation
  - Doigts : rotation et position
  - Poignet : rotation
  - Aucune monture n'est disponible pour les VIVE Tracker. Un bracelet externe doit être ajouté.
  - 7 degrés de liberté
- **Capteurs**
  - 7 Capteurs IMU de 9 axes
  - Gyroscope
  - Accéléromètre
  - Magnétomètre
- **Latence**
  - $<$ 10 millisecondes
- **Batterie**
  - 10 heures
- **SDK**
  - Unity 5.x
  - Unreal Engine 4
  - C++
  - Android
  - Documentation sur le site du développeur : lien du site
- **Suivi de la team**
  - Dernière mise à jour du SDK le 12.02.2017
  - La team Senso Device Inc. n'est pas très réactive aux mails (temps de réponse entre 1 et 2 semaines) et n'hésite pas à répondre uniquement aux questions concernant les gants.

  - Après avoir posé la question d'une possible interaction entre la main virtuelle et les gameobjects, la team n'a donné aucune réponse.
  - Pour positionner correctement la main dans la scène, la team Senso Device Inc. a répondu en 7 jours.

- **Prix**
  - Senso Glove : DK2 (1 paire de Senso) : $599

Chapitre 4

Installation, connexion et calibration

Tous les fichiers à téléchargers se trouve sur la page Dev Center du site senso.me.

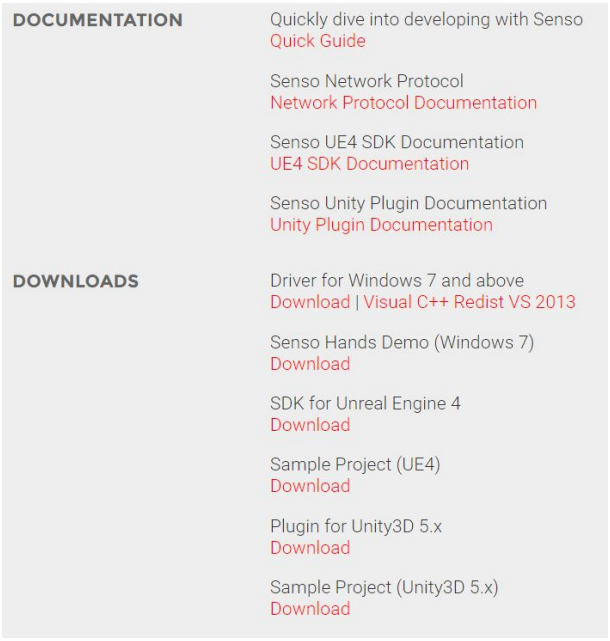

FIGURE 4.1 – Page de téléchargement du site Senso

4.1 Installation du driver

Pour pouvoir connecter les gants Senso au PC, il faut installer le driver de l'adaptateur BLE. Ce dernier est disponible sur le site Senso. La procédure d'installation est décrite ci-dessous.

- Branchez **un seul** adaptateur BLE au PC. L'adaptateur BLE est une petite clé USB sur laquelle est noté **CSR 4.0**.
- Une fois que l'adaptateur a été reconnu par le PC, lancez l'application *Zadig.exe* téléchargée précédemment.
- Sélectionnez **Options** puis **List All Devices**.

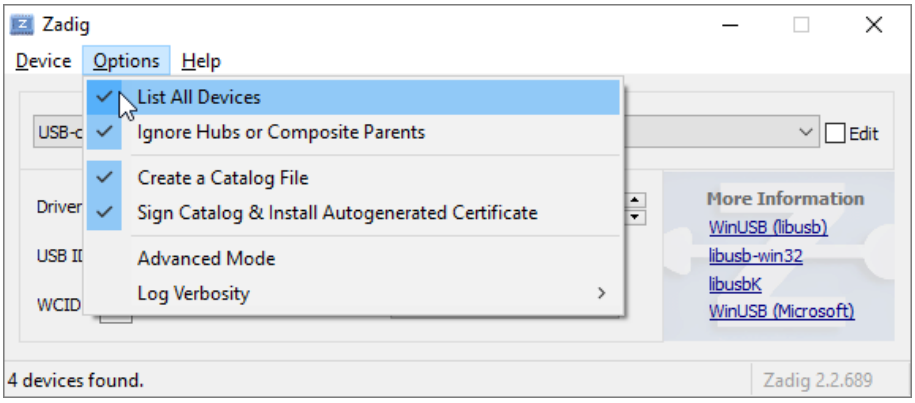

FIGURE 4.2 – *Zadig.exe* : List All Devices

— Si l'adaptateur BLE a bien été reconnu, le nom **CSR8510 A10** devrait être disponible dans la liste déroulante. Sélectionnez-le.

— Cliquez sur le bouton **Replace Driver** puis laissez l'application installer le driver (peut prendre un peu de temps).

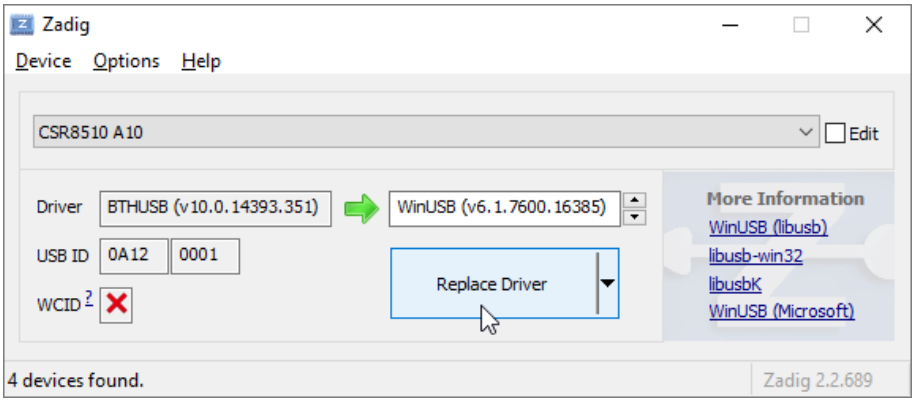

FIGURE 4.3 – *Zadig.exe* : Replace Driver

— Une fois que l'installation est finie, le message **Driver Installation : SUCESS** devrait s'afficher. Le 2ème périphérique USB peut maintenant être branché.

4.2 Connexion aux gants Senso

La connexion aux gants Senso est mise en place grâce à deux applications : le serveur et l'interface. Les deux adaptateurs BLE doivent être branchés au PC pour pouvoir se connecter aux deux gants.

4.2.1 Senso BLE Server

Cette application doit être exécutée pour chaque adaptateur BLE connecté au PC. Par défaut, l'application utilise le port **9872**. Le lancement de deux instances de l'application nécessite la spécification d'un 2ème port, en passant */sport=<port>* comme argument à l'application, via l'invite de commande.

Si l'adaptateur BLE est bien branchée et est reconnu par le serveur, l'icône de l'application Senso BLE Server s'affichera en bleu. Sinon, l'icône reste grise.

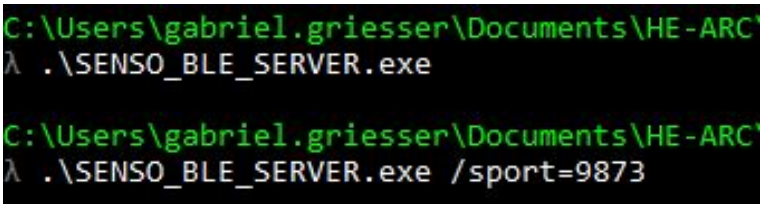

FIGURE 4.4 – Lancement de deux instances du serveur

FIGURE 4.5 – Icône Senso BLE Server. Un seul adaptateur USB branché

FIGURE 4.6 – Icône Senso BLE Server. Deux adaptateurs USB branchés

4.2.2 Senso User Interface

Pour chaque adaptateur BLE branché et connecté, vous devez exécuter une instance de l'application **Senso_UI**. Une fois démarrée, vous devrez spécifier le port de l'application Senso BLE Server et devrez cliquer sur **Connect To Server**.

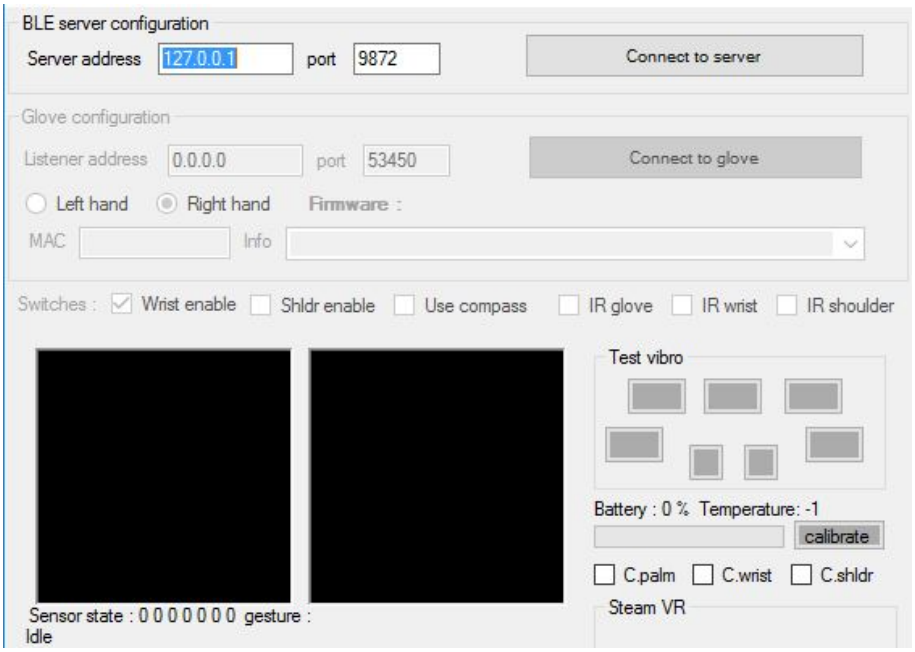

FIGURE 4.7 – Senso UI - Connexion

Vous pouvez maintenant vous connecter au gant en spécifiant les informations de ce dernier.

- La *listener address* doit être laissée à sa valeur par défaut, 0.0.0.0.
- Le port utilisé pour la connexion au gant peut également être laissé à sa valeur par défaut. C'est ce port que qui sera spécifié dans Unity pour relier la scène à l'interface.
- Finalement, sélectionnez le gant Senso dans la liste déroulante puis cliquez sur **Connect to glove**.

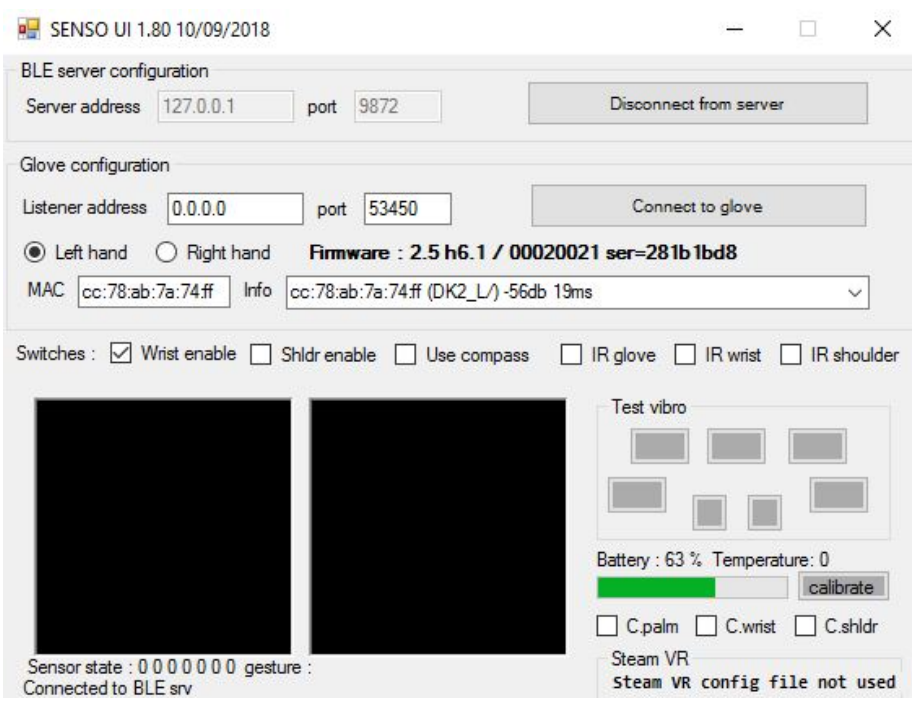

FIGURE 4.8 – Senso UI - Connexion au gant gauche en utilisant le port 53450

Si la connexion est établie entre un gant et l'interface Senso_UI, cette dernière devrait ressembler à la figure ci-dessous.

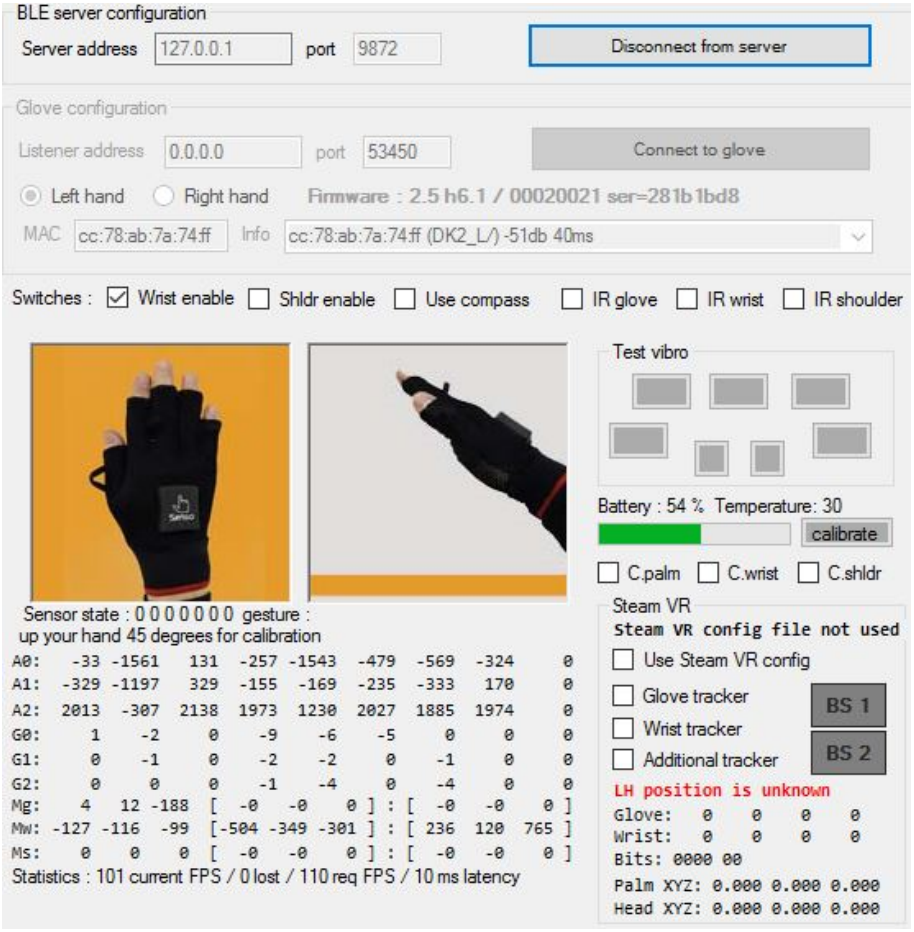

FIGURE 4.9 – Senso UI - Connexion établie

Les gants sont maintenant prêts à être utilisés dans n'importe quelle application. Comme l'étape de connexion est obligatoire avant chaque utilisation des Senso, il est recommandé de lancer simplement le script *run.cmd*, ou *run_lh.cmd* si vous utilisez les gants dans une *lighthouse*. Ces scripts sont réglés pour démarrer le serveur et l'interface avec la configuration nécessaire à la bonne connexion de deux gants Senso.

Le contenu du script *run.cmd* est le suivant :

```
start SENSO_BLE_SERVER.exe /saddr=0.0.0.0 /sport=53452 /channels=ffff0000e0
start SENSO_BLE_SERVER.exe /saddr=0.0.0.0 /sport=53453 /channels=00000fffef

start SENSO_UI.exe /saddr=127.0.0.1 /sport=53452 /caddr=127.0.0.1 /cport=53450
/udpaddr=127.0.0.1 /udpport=53451 /left /reconnect /name=DK2_L
start SENSO_UI.exe /saddr=127.0.0.1 /sport=53453 /caddr=127.0.0.1 /cport=53451
/udpaddr=127.0.0.1 /udpport=53450 /right /reconnect /name=DK2_R
```

Pour plus d'informations concernant l'utilisation des arguments en ligne de commande, veuillez-vous rendre à cette adresse.

4.3 Calibration

Lorsqu'un gant est connecté, il faut suivre la procédure de calibration indiquée par l'application Senso UI. L'utilisateur devra positionner sa main à l'horizontale, puis à 45°. Les figures ci-dessous décrivent la procédure de calibration.

FIGURE 4.10 – Calibration du Senso : main à l'horizontale

FIGURE 4.11 – Calibration du Senso : main à 45°

Une fois la calibration finie, l'application SENSO_UI affichera les axes de rotation du gant.

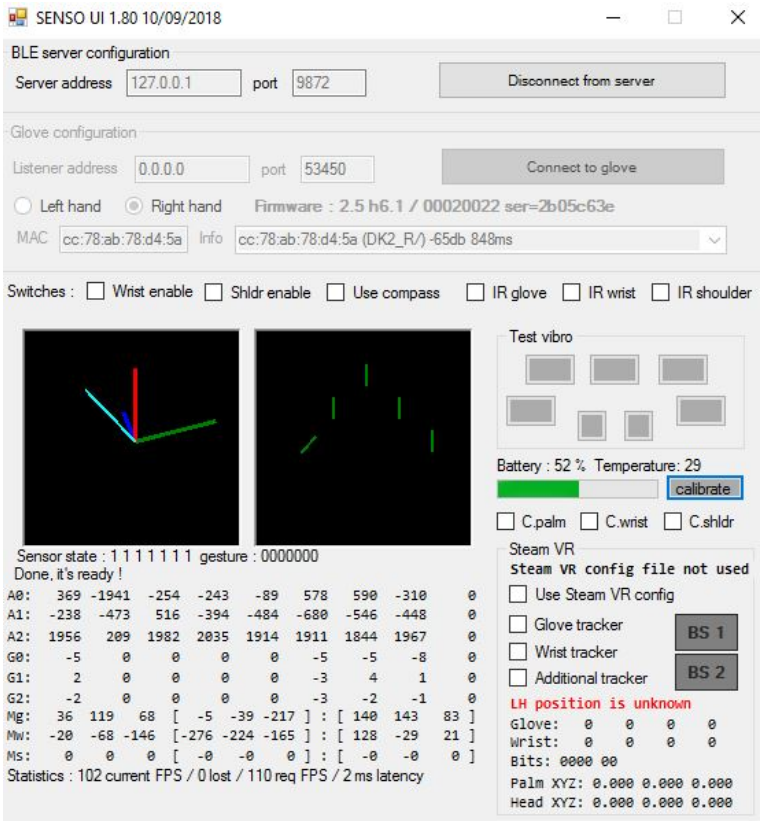

FIGURE 4.12 – Calibration du Senso : calibration du gant droit finie

Afin de vérifier le bon fonctionnement, vous pouvez jongler entre les mouvements de la main et le test de vibration.

Chapitre 5

SDK

Le *Software Development Kit* (SDK) est un kit de développement conçu pour le moteur de jeu Unity version 5.x et Unreal Engine 4. Son utilisation nécessite une connexion aux gants avec les application **SENSO_BLE_SERVER** et **SENSO_UI**. Le SDK est conçu pour fonctionner dans n'importe quel projet 3D Unity, avec ou sans réalité virtuelle. Si la réalité virtuelle devait être intégrée au projet, les fichiers nécessaires comme le plugin SteamVR de Valve, devront être téléchargés et installés séparément.

Le SDK est un package Unity à importer directement dans le projet via **Assets -> Import Package -> Custom Package**.

La documentation du SDK est accessible en ligne.

5.1 Architecture

Cette section décrit comment l'architecture du SDK fonctionne pour créer la connexion aux gants, l'envoi / la réception des données, et les retours haptiques.

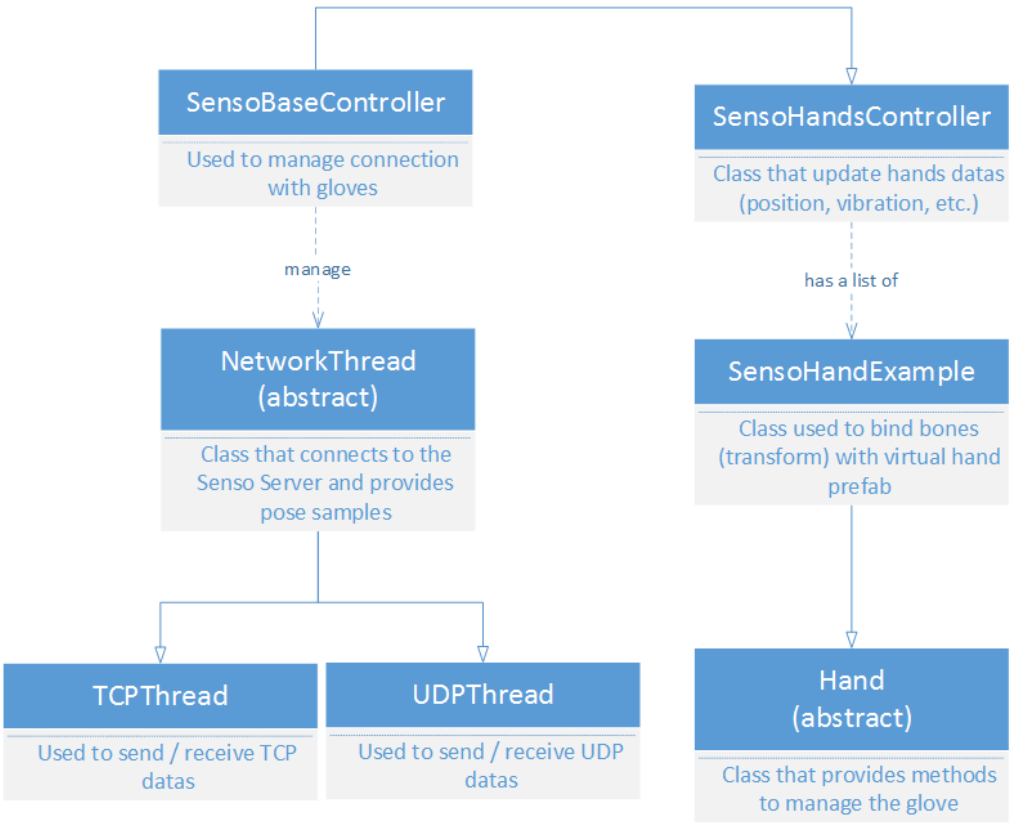

Figure 5.1 – Diagramme UML simplifié du SDK Unity

— **NetworkThread** : une classe abstraite dont le rôle est, à la fois de connecter les gants au serveur, à la fois de définir les méthodes d'envoi et réceptions des données (connexion, position, rotation, vibration et ping).

— **TCPThread et UDPThread** : des classes héritant de NetworkThread. Elles gèrent l'envoi et la réception des paquets au serveur. Ces paquets (connexion, position, rotation et vibration) sont envoyés sous format JSON, via bluetooth, par le protocole TCP ou UDP. Le thread, implémenté dans chacune des classes, s'occupe de la lecture et de l'écriture des données.

— **SensoBaseController** : la classe mère de gestion de connexion des gants. Elle possède les informations de connexion au serveur (hôte, port, protocole utilisé), permettant ainsi une liaison direct au TCP/UDP Thread associé au gant.

— **SensoHandsController** : une classe héritant de la classe SensoBaseController. Son rôle principal est de mettre à jour toutes les données des gants à chaque frame de l'application. L'appel à la fonction d'envoi de vibration se fait par le biais de cette classe.

— **SensoHandExample** : ce script est attaché au gameobject qui représente la main virtuelle. Les liaison avec les transforms se font ici. La fonction de mise à jour de position et de rotation (*SetSensoPose(HandData)*) de la main virtuelle est redéfinie puis utilisée dans cette classe. Cette méthode met à jour les propriétés de la main virtuelle en fonction des valeurs des propriétés du gant Senso, qui sont contenues dans le conteneur **HandData**.

— **Hand** : classe abstraite dont hérite SensoHandExample. Elle définit un gant et, de manière abstraite, sa méthode *SetSensoPose(HandData)*.

— **HandData** (non-affiché sur le diagramme) : classe implémentant un conteneur pour les informations de position et rotation de la paume, des doigts et du poignet de la main sur laquelle est mise le gant Senso. Les informations sont directement reçues de l'application SENSO_UI. Elles seront ensuite traitées par la méthode *SetSensoPose(HandData)*, redéfinie dans la classe SensoHandExample.

Chapitre 6

Intégration

Cette section décrit la mise en place des gants Senso dans un projet Unity configuré pour la réalité virtuelle (utilisation du casque HTC Vive et des contrôleurs VIVE Trackers).

Comme décrit dans le chapitre précédent, les Senso ne dépendent d'aucun plugins tiers (exceptées les applications SENSO_UI et SENSO_BLE_SERVER). Cependant, l'utilisation de la réalité virtuelle nécessite de télécharger, installer et configurer le plugin de Valve : SteamVR, disponible dans l'Asset Store de Unity. Vous pouvez suivre les étapes disponibles à cette adresse pour mettre en place cette configuration.

6.1 Setup de la scène

— Pour faciliter l'architecture du projet, nous ajouterons, dans **[CameraRig]**, un objet vide : **Tracked Devices**. Il contiendra les différents périphériques de suivi utilisés dans l'application. Dans notre cas, nous auront 4 périphériques (2 Vive Tracker et 2 Base Station), nommés **Device1, Device2, Device3** et **Device4**.

FIGURE 6.1 – Architecture du dossier **[CameraRig]**

— Ajoutez le package *SensoPlugin.unitypackage* téléchargé précédemment dans le projet en cliquant sur **Asset -> Import package -> Custom package**. À la suite de ça, le dossier Assets devrait contenir plusieurs dossiers et fichiers comme montré dans la figure ci-dessous.

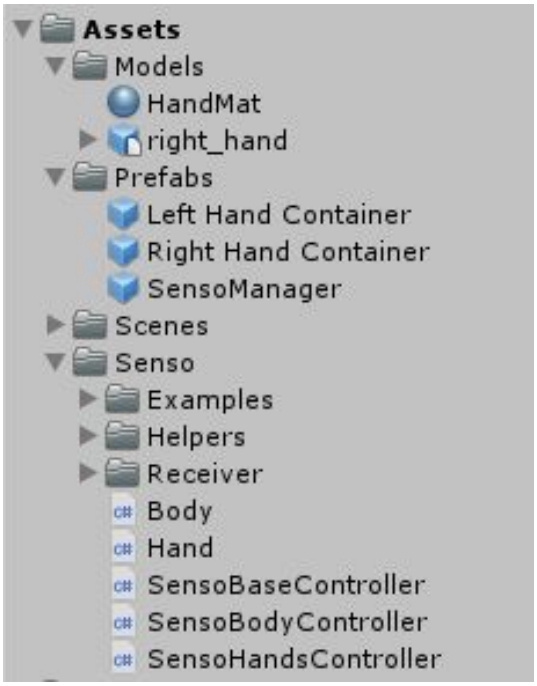

FIGURE 6.2 – Dossiers et fichiers contenus dans le SDK

6.2 Intégration de deux gants Senso

Nous allons maintenant contrôler deux mains virtuelles avec les Senso. Pour ce document, nous utiliserons le modèle de main préfabriqué du SDK Senso.

— Glissez-déposez les prefabs **Left Hand Container** et **Right Hand Container** dans la scène.

— Placez les prefabs en tant qu'enfant des gameobjects qui sont suivis par les systèmes de tracking VIVE Tracker (DeviceX et DeviceY).

— Si nécessaire, ajustez la rotation et la position locale des modèles pour qu'ils correspondent à la façon dont l'objet suivi est relié à la main. Cette modification doit se faire directement dans la méthode *SetSensoPose()* du script *SensoHandExample*.

```
public override void SetSensoPose (Senso.HandData aData)
{

    // Rotation = (90, 0, 0) because we use Tracker rotation (+90 x for adaptation) for palmRotation
    Palm.localRotation = Quaternion.Euler(90, 0, 0); //*(Quaternion.Inverse(wq) */ aData.PalmRotation;

    //Position = 0 (because we use Tracker position for palmPosition)
    Palm.localPosition = Vector3.zero; //aData.PalmPosition;
```

FIGURE 6.3 – Modification de la position et rotation es mains

Les prefabs des mains virtuelles ont été créés et configurés pour fonctionner directement après leurs importations dans la scène.

Le prefab Left / Right Hand Container représente le modèle virtuel de la main gauche / droite du gant Senso. Ce modèle est composé de plusieurs gameobjects, **Bone.0XX**, chacun représentant un os du doigt. Ces gameobjects sont automatiquement liés, via l'inspecteur, au script SensoHandExample qui est attaché au prefab.

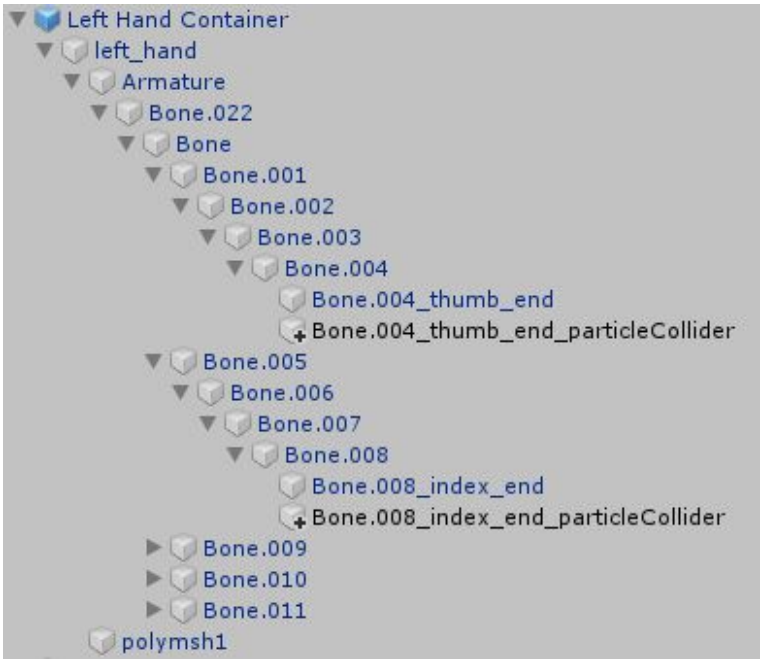

FIGURE 6.4 – Composition du prefab Left Hand Container

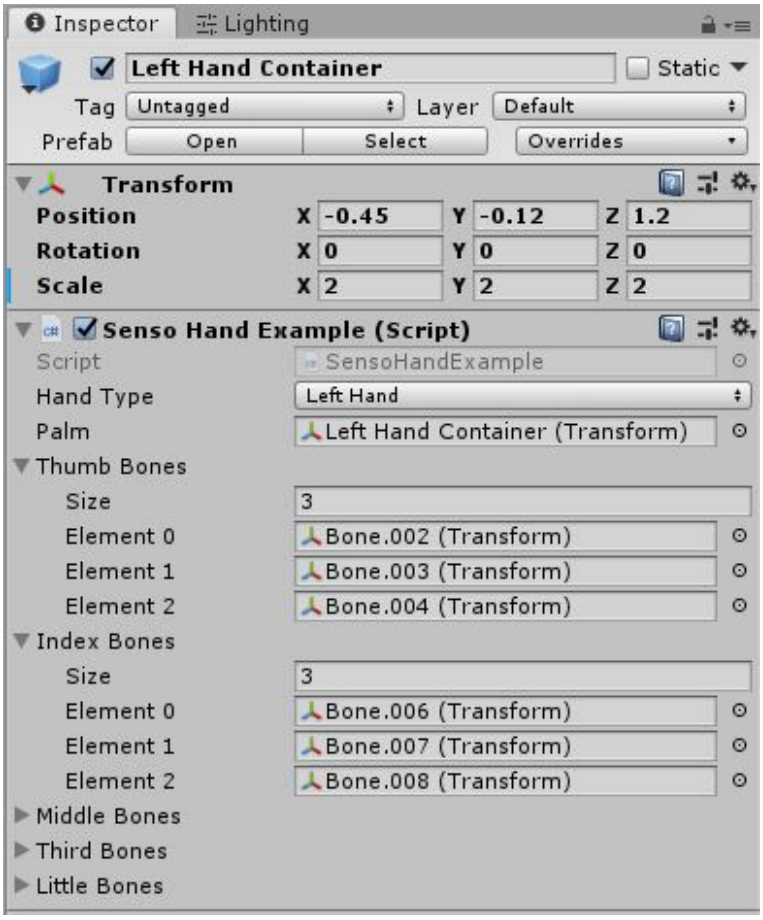

FIGURE 6.5 – Assignation des gameobjects du prefab dans le script SensoHandExample

— Pour finir l'intégration des gants Senso à Unity, il est nécessaire d'ajouter le prefab **Senso-Manager** à la scène. Ce prefab contient le script SensoHandsController, dont le rôle est de gérer le bon fonctionnement des gants durant toute l'instance de l'application. C'est ici que nous définirons l'hôte de connexion, le port, le protocole (TCP ou UDP) et le nombre de gant Senso utilisé. Le port utilisé par défaut est celui défini au lancement du script *run.cmd*.

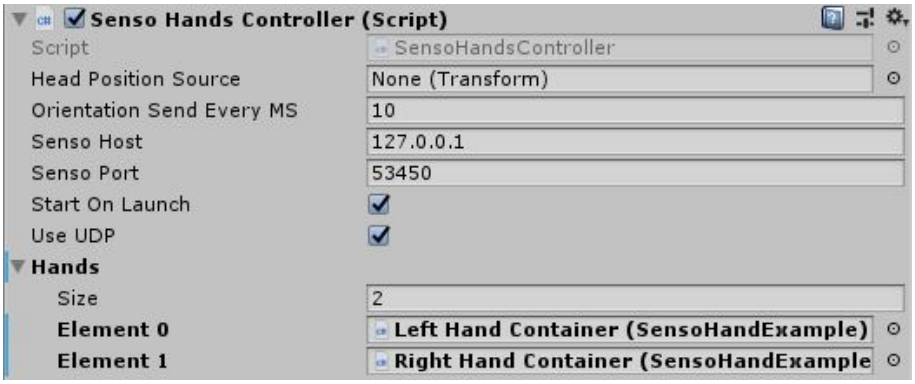

FIGURE 6.6 – Script SensoHandsController

— Comme décrit dans le chapitre 4, la connexion aux gants Senso dépend des application SENSO_BLE_SERVER et SENSO_UI. Exécutez le script *run.cmd* pour connecter les gants.

— Lancez ensuite votre scène Unity. Vérifiez que les mains virtuelles sont bien contrôlées avec les gants Senso et que leurs positions correspondent à ceux de vos Vive Trackers. Si la calibration est imprécise, vous pouvez en relancer une via l'application SENSO_UI.

Chapitre 7

Remarques et problèmes connus

Cette section décrit toutes les remarques et problèmes connus concernant les gants Senso.

— Les gants ne sont pas faits pour les applications lourdes. Trop de vibrations et de mouvements provoque une latence de plusieurs millisecondes, allant même jusqu'à la déconnexion du gant. Cette latence est dû à l'envoi, la réception et le traitement des données entre le serveur, l'interface, et le SDK.

Malgré une étude des trames et des données transmises entre le client et le serveur, aucune solution n'a été trouvée pour résoudre ce problème de latence, excepté la diminution des interactions et des mouvements effectués avec les gants. En effet, les trames transmises entre le client et le serveur contiennent les données de position, de rotation, et d'envoi de vibration. Il n'est donc pas possible de modifier ces dernières afin d'alléger le traitement des données

— Le champ température du gant reste fixe. La société a affirmé utiliser ce champs uniquement pour la compatibilité. Il n'est donc pas possible de changer la température du gant.

— Pour créer les interactions avec les objets, il faut ajouter un collider à chaque bout de doigt avec la propriété **isTrigger** à true. Cela permettra de détecter la collision entre un doigt et un objet ayant un rigidbody. A vous ensuite de créer le script permettant la gestion de cette collision comme sur la figure ci-dessous.

```
/// <summary> Send haptic feedback when trigger
void OnTriggerEnter(Collider other)
{
    if (other.GetComponent<Senso_Material>() && other.GetComponent<Senso_Material>().hapticFeedback)
        SendHapticFeedback(other);
}
```

FIGURE 7.1 – Script lié à chaque bout de doigt, permettant d'envoyer une vibration quand un doigt touche un objet

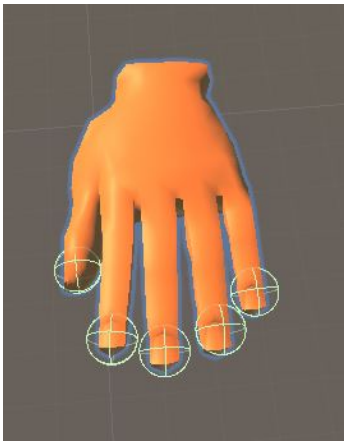

FIGURE 7.2 – Sphere colliders ajoutés à chaque bout de doigt

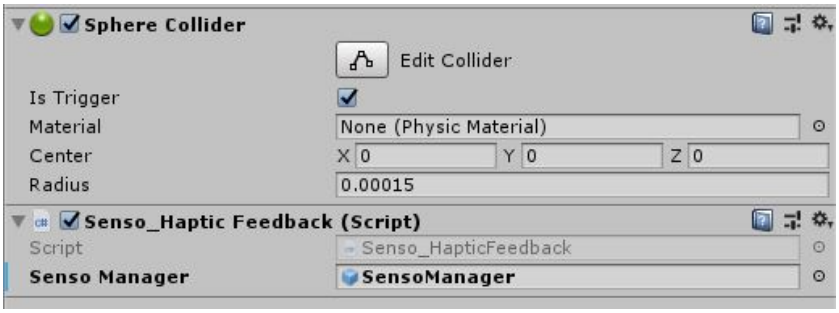

FIGURE 7.3 – Composant du gameobject Bone.004_thumb_end utilisé pour les interactions avec les objets virtuels

— La collision entre les particules et les doigts n'est pas détectée. Comme pour les gants SenseGlove, il faut ajouter, pour chaque doigt, un deuxième gameobject possédant les composants suivants : un sphere collider avec la propriété **isTrigger** décochée et un rigidbody avec la propriété **is Kinematic** cochée.

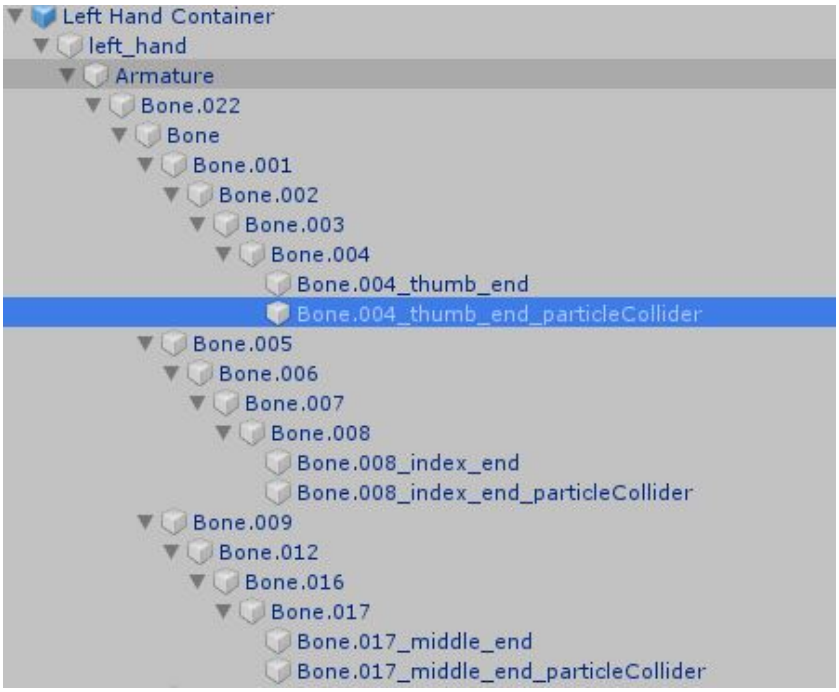

FIGURE 7.4 – Gameobject Bone.004_thumb_end_particleCollider ajouté au bout du pouce

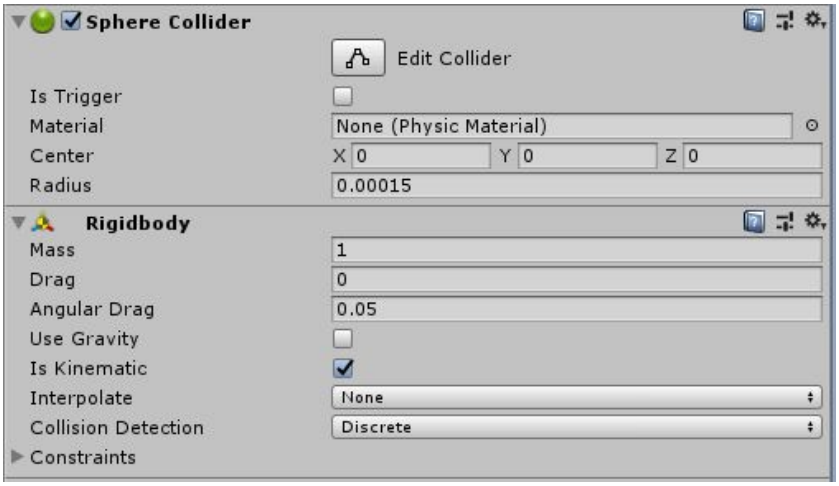

FIGURE 7.5 – Réglage des composants Sphere Collider et Rigidbody du gameobject Bone.004_thumb_end_particleCollider