

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

SENSE GLOVE - DETAILS

Fait par :

Gabriel Griesser



Table des matières

1	Introduction	1
2	Description	3
3	Propriétés et caractéristiques	5
4	SDK	7
4.1	Architecture	7
5	Calibration	11
5.1	Calibration du poignet	11
5.2	Calibration de la main	11
5.3	Méthodes de calibration	12
6	Intégration	13
6.1	Setup de la scène	13
6.2	Intégration de deux SenseGlove	14
7	Interactions	17
7.1	SenseGlove_Material	17
7.2	SenseGlove_Grabable	18
8	Remarques et problèmes connus	21

Chapitre 1

Introduction

Ce document décrit, pour le gant SenseGlove :

- Sa description
- Ses propriétés et caractéristiques
- La description du SDK et son architecture
- Le processus de calibration
- Le mode d'emploi d'intégration de deux gants
- Le mode d'emploi des interactions
- Les remarques et problèmes connus

Pour faciliter la lecture, les scripts dont la structure du nom est présentée comme SenseGlove_**ScriptName** seront réutilisés seulement par leurs noms **_ScriptName**.

Chapitre 2

Description

La première paire de gants, nommée SenseGlove, est un dispositif haptique exo-squelettique développé par la société SenseGlove Inc. Ils sont reliés par câbles à un moteur central, lui-même relié au PC par deux câbles (voir schéma section suivante).

Étant donné leur structure exo-squelettique, ces gants possèdent un système de retour de force allant jusqu'à 1,8 Newton par doigt. Ce retour de force est appliqué à chaque doigt et permet ainsi le ressenti de la forme, la taille et la rigidité de l'objet avec lequel nous interagissons.

Avec un moteur de vibration présent aussi à chaque doigt, les SenseGlove peuvent produire des vibrations pour des signaux interactifs comme le clic sur un bouton. Ces vibrations peuvent également être utilisés pour simuler des textures rugueuses. Grâce à leurs 24 degrés de liberté, les SenseGlove reproduisent, en temps réel, et avec précision, les doigts, la main et le poignet par tous les gestes possibles.

Chapitre 3

Propriétés et caractéristiques

- **Type**
 - Exosquelette
- **Connexion**
 - Câblée
 - Liaison au PC pas câbles avec un passage par le moteur central
 - La liaison peut se faire avec un ou deux gants

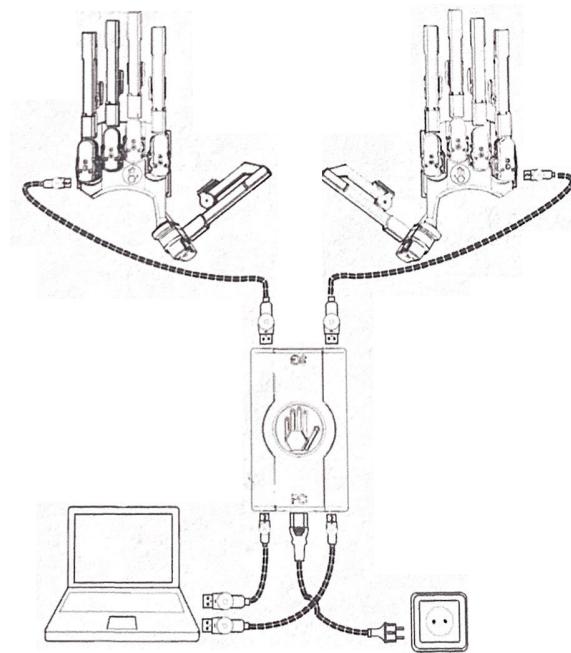


FIGURE 3.1 – Connexion SenseGlove - PC

- **Retour de force**
 - Oui, jusqu'à 1.8 Newton par doigt
- **Retour tactile**
 - Oui, avec un système de vibration pour chaque doigt
- **Suivi de mouvement (tracking)**
 - Main : rotation
 - Doigts : rotation et position
 - Poignet : rotation
 - Deux montures pour les VIVE Tracker sont livrées avec les gants permettant ainsi de récupérer la position de la main (et donc du poignet) dans le monde virtuel
 - 24 degrés de liberté
- **Capteurs**
 - Capteurs IMUs (nombre inconnu)
 - Gyroscope

- Accéléromètre
- Magnétomètre
- **Latence**
- Aucune
- **Batterie**
- Illimitée
- **SDK**
 - SDK Unity
 - Documentation complète sur le dépôt GitHub : [Wiki du dépôt](#)
- **Suivi de la team**
 - Dernière mise à jour du SDK le 18.06.2019
 - La team SenseGlove est réactive aux mails (temps de réponse entre 2 et 3 jours ouvrables) et n'hésite pas à proposer diverses solutions si un problème est rencontré.
- **Prix**
 - *Developers Pack* (1 paire de SenseGlove) : \$2500
 - *Professional Bundle* (1 paire de SenseGlove + intégration au projet + support personnel et maintenance) : \$7500

Chapitre 4

SDK

Le *Software Development Kit* (SDK) est un kit de développement conçu uniquement pour le moteur de jeu Unity. Son utilisation ne repose sur aucun plugin tiers et est conçue 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, disponible à [cette adresse](#), est un package Unity à importer directement dans le projet via **Assets -> Import Package -> Custom Package**.

La documentation du SDK est accessible en ligne sur le [wiki du dépôt](#).

4.1 Architecture

Cette section décrit comment l'architecture des différents scripts fonctionne pour créer la connexion, les interactions et les retours haptiques des gants SenseGlove.

Pour mieux comprendre les interactions, il est nécessaire de se familiariser avec le système de colliders de Unity. Toute interaction entre le SenseGlove et un objet virtuel signifie que les colliders auront été déclenchés. Je vous invite à lire [cette page](#) pour plus d'informations.

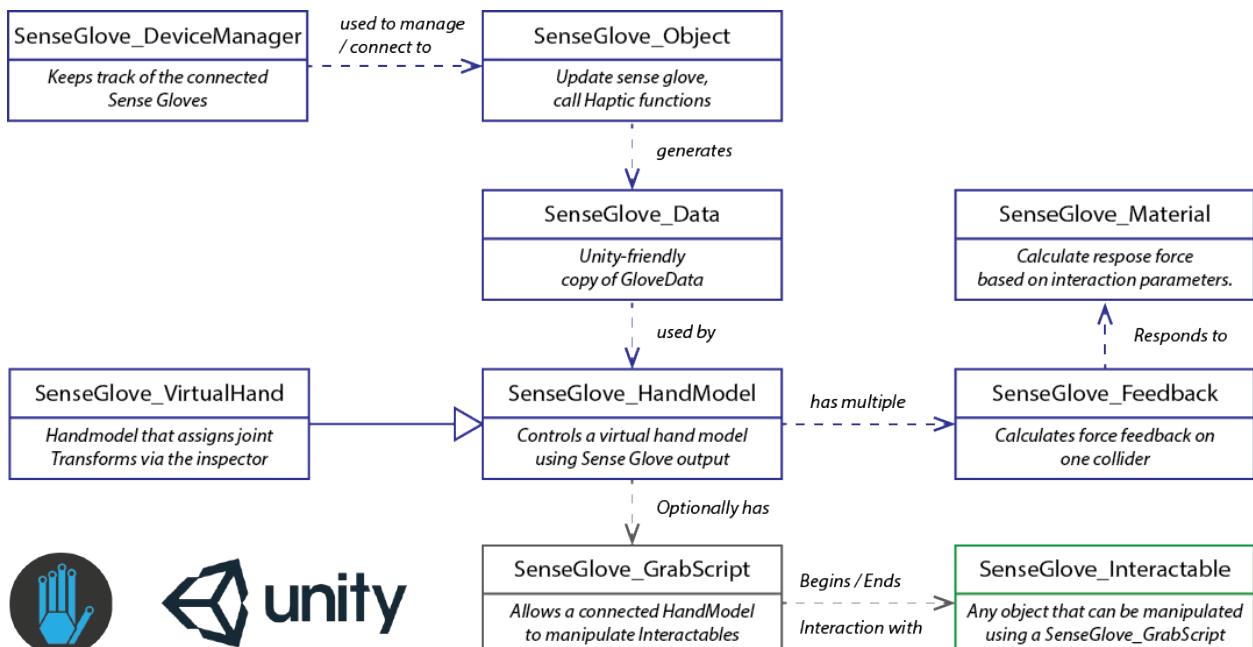


FIGURE 4.1 – Architecture du SDK SenseGlove

La figure ci-dessus représente l'architecture principale du SenseGlove et comment les scripts sont liés pour créer les interactions et ressentis avec l'environnement virtuel.

- **SenseGlove_HandModel** : une classe abstraite générique pouvant être étendue pour fonctionner avec la plupart des modèles de main. Elle possède de nombreuses méthodes virtuelles, et est utilisée pour assigner les jointures d'articulation calculées par le SenseGlove, de l'avant-bras jusqu'au bout des doigts. La classe *_HandModel* est également responsable à la fois de la récupération des éléments de ses scripts *_Feedback* et de leurs envois au gant SenseGlove, à la fois de la calibration du poignet grâce à la méthode *CalibrateWrist()*.

Documentation complète de la classe.

- **SenseGlove_VirtualHand** : *_VirtualHand* hérite de la classe abstraite *_HandModel*, et collecte automatiquement les données d'un *_Object* pour créer un modèle de main virtuelle. En plus des propriétés et méthodes héritées de sa classe parente *_HandModel*, le script *_VirtualHand* possède une liste de transforms représentant les articulations de la main virtuelle. Ces derniers peuvent être assignées sous Unity par l'intermédiaire de l'inspecteur, ce qui permet une intégration plus facile.

Documentation complète de la classe.

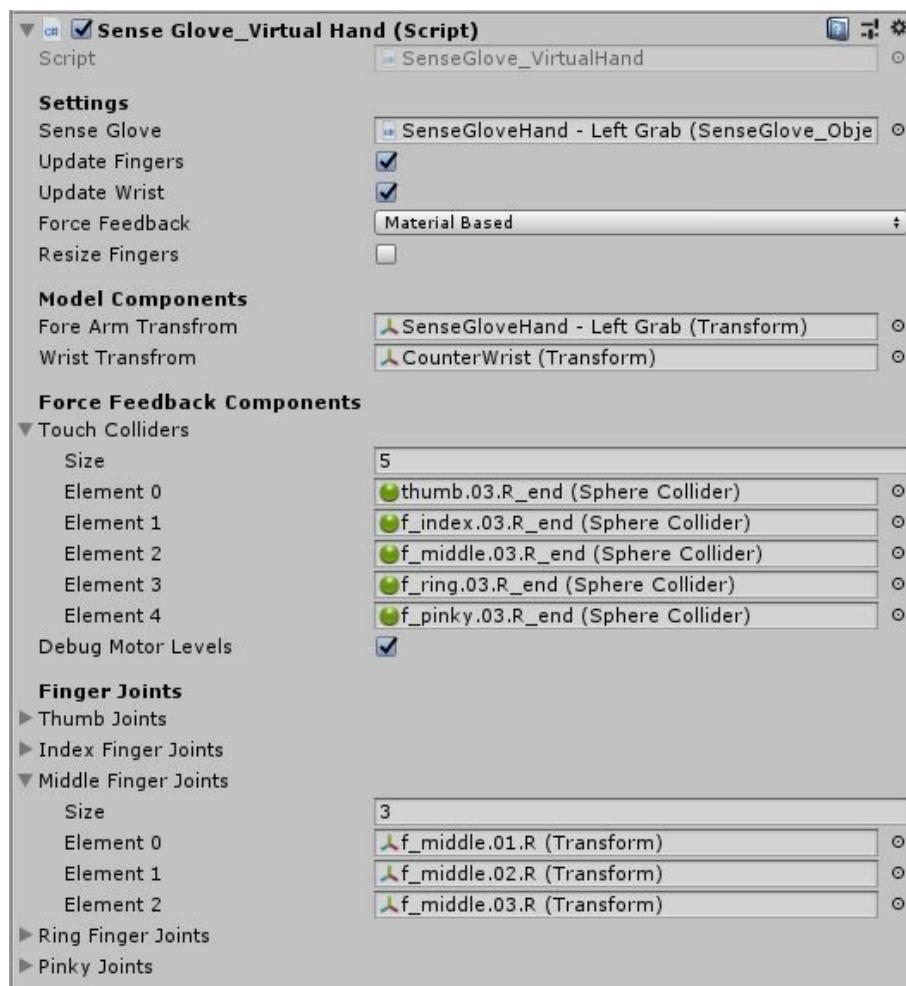


FIGURE 4.2 – Assignation des gameobjects des doigts au script *_VirtualHand*



FIGURE 4.3 – Liaison au script `_VirtualHand`

- **SenseGlove_Data** : Ce script contient toutes les données liées au SenseGlove. N’importe quel script ayant accès à l’objet `_HandModel` pourra récupérer les informations liées au gant afin d’en faire le traitement qu’il souhaite. Parmi les données des SenseGlove, nous retrouvons :
 - Les propriétés générales comme la version du gant, le type de gant (droite ou gauche), la version du firmware, etc.
 - Les informations sur l’objet SenseGlove telles que la position et l’orientation de la main et du poignet, la position et la longueur de chaque segment du gant, etc.
 - Les informations relatives aux capteurs (axes IMU)Documentation complète de la classe.
- **SenseGlove_DeviceManager** : est un script relié à un gameobject, généralement invisible. Le `_DeviceManager` est responsable de la bonne instanciation des gants. Par instanciation nous définissons la détection de connexion et de déconnexion des SenseGlove, suivi de leurs liaisons au script `_Object` en fonction des paramètres de connexion.
Documentation complète de la classe.
- **SenseGlove_Object** : est attaché en tant que script à la fois au modèle de la main, et à la fois au script `_DeviceManager` pour lier correctement le gant à l’application. Ainsi, le script `_Object` permet de connecter et mettre à jour les données, `_Data`, du SenseGlove à chaque frame de l’application.
Documentation complète de la classe.

- **SenseGlove_GrabScript** : est une classe abstraite qui contient toutes les méthodes et propriétés partagées nécessaire aux interactions avec les objets de la scène. Ce script utilise un certain nombre de données du SenseGlove, les *_Data*, pour démarrer et terminer des interactions. Grâce à cette classe, les systèmes gestuels, physiques et même les systèmes personnalisés peuvent interagir avec les objets ayant le script *_Interactable* rattaché
[Documentation complète de la classe.](#)
- **SenseGlove_PhysGrab**(non-affiché sur le diagramme) : hérite de la classe abstraite *_GrabScript*. Ce script met en place un système d’interaction basé sur la physique. *_PhysGrab* utilise les collisions entre la main et les objets plutôt que les gestes pour déterminer quand il faut ramasser un objet *_Interactable*. Il est directement attaché au gameobject représentant le gant (qui a le script *_Object* comme composant) et hérite des propriétés et méthodes de sa classe parente, *_GrabScript*.
[Documentation complète de la classe.](#)
- **SenseGlove_Interactable** : contient une classe abstraite représentant un objet avec lequel le script *_Grabscript* peut interagir. Cette classe est étendue par la plupart des scripts d’interaction.
[Documentation complète de la classe.](#)
- **SenseGlove_Grabable** (non-affiché sur le diagramme) : ce script a pour classe parente *_Interactable*. Un gameobject, auquel le script *_Grabable* est rattaché, pourra être ramassé et déposé par le SenseGlove.
[Documentation complète de la classe.](#)
- **SenseGlove_Feedback** : c'est ici que le retour de force et le retour tactile est calculé en fonction de la méthode choisie et des propriétés du script *_Material* de l'objet touché. Le script *_Feedback* est attaché à chacun des gameobjects représentant le bout des doigts. Dans la figure 4.3, c'est le gameobject **f_middle.03.R_end**, le bout du majeur, qui aura le script *_Feedback* attaché. Le retour de force est uniquement déclenché si le script *_Material* est rattaché à l'objet touché.
[Documentation complète de la classe.](#)
- **SenseGlove_Material** : contient les propriétés du matériau assigné à un objet. Tout gameobject ayant le script *_Material* rattaché déclenchera les systèmes de retour de force et de retour tactile du gant par le biais du script *_Feedback*.
[Documentation complète de la classe.](#)
- **SenseGlove_MeshDeform** (non-affiché sur le diagramme) : une classe, qui se connecte à un *_Material*, dont la tâche est de déformer le maillage (mesh) de l'objet afin de créer un retour visuel.
[Documentation complète de la classe.](#)
- **SenseGlove_Breakable** (non-affiché sur le diagramme) : ce script supprime un gameobject de la scène lorsque son *_Material* se casse. Nous pouvons éventuellement remplacer l'objet par une version cassée, et jouer un son ou un système de particule.
[Documentation complète de la classe.](#)

Chapitre 5

Calibration

Le système de calibration est composé de 2 méthodes différentes : la calibration du poignet et celle des doigts. La calibration du poignet est appelée par la fonction `_HandModel.CalibrateWrist()`, alors que la calibration de la main / doigts est appelée par la fonction `_Object.StartCalibration(variableToCalibrate, collectionMethod)`. C'est le paramètre `variableToCalibrate` qui définit quelle calibration nous souhaitons effectuer (calibration des mains, des doigts, par longueur ou flexion, etc.).

5.1 Calibration du poignet

Pour le cas du poignet, il faut utiliser la fonction `_HandModel.CalibrateWrist()`. Cette méthode, qui se sert de l'orientation de l'avant bras si ce dernier est disponible, est automatiquement appelée à la connexion d'un SenseGlove. Cette étape n'est pas nécessaire si vous utilisez un tracker monté directement sur les SenseGlove ou si vous n'utilisez pas le mouvement du poignet. Dans le cadre du projet **TMS => Gants Haptiques**, cette calibration n'est pas utilisée car le projet utilise les VIVE Tracker.

5.2 Calibration de la main

Pour mettre en place la calibration de la main, l'équipe de SenseGlove a développé un algorithme qui détermine la longueur des doigts et la position de ces derniers en se basant sur trois *snapshots*. Plus les points d'étalonnage sont éloignés, plus la calibration de la main est précise.

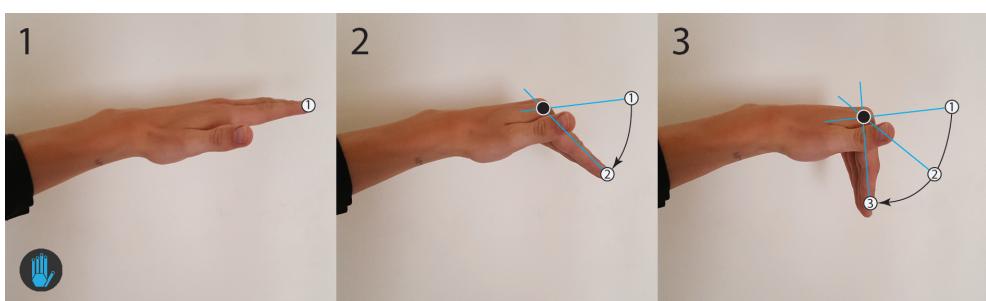


FIGURE 5.1 – Calibration de la main en utilisant trois *snapshots*

Le pouce est calibré de la même manière, en utilisant la flexion de ce dernier comme le montre la figure ci-dessous.

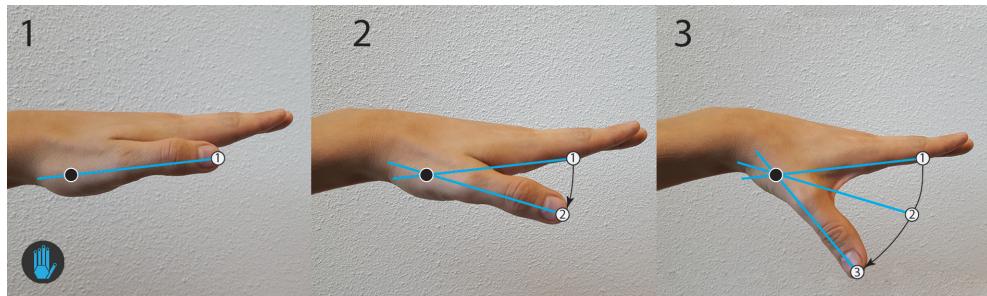


FIGURE 5.2 – Calibration du pouce en utilisant trois *snapshots*

5.3 Méthodes de calibration

Le SDK des SenseGlove offre 2 façons différentes de collecter les *snapshots* : calibration manuelle et semi-automatique.

Le choix de la récupération des *snapshots* se fait via le paramètre `collectionMethod` passé à la fonction `_Object.StartCalibration(variableToCalibrate, collectionMethod)`. C'est cette dernière qui se chargera de calibrer la variable souhaitée (main, doigts, etc.), en utilisant la méthode de collection de *snapshots* souhaitée. Chacune de ces méthodes a un impact différent sur l'application. L'utilisation en puissance de calcul, en mémoire, et en effort utilisateur change entre les deux méthodes, comme démontré dans la figure ci-dessous.

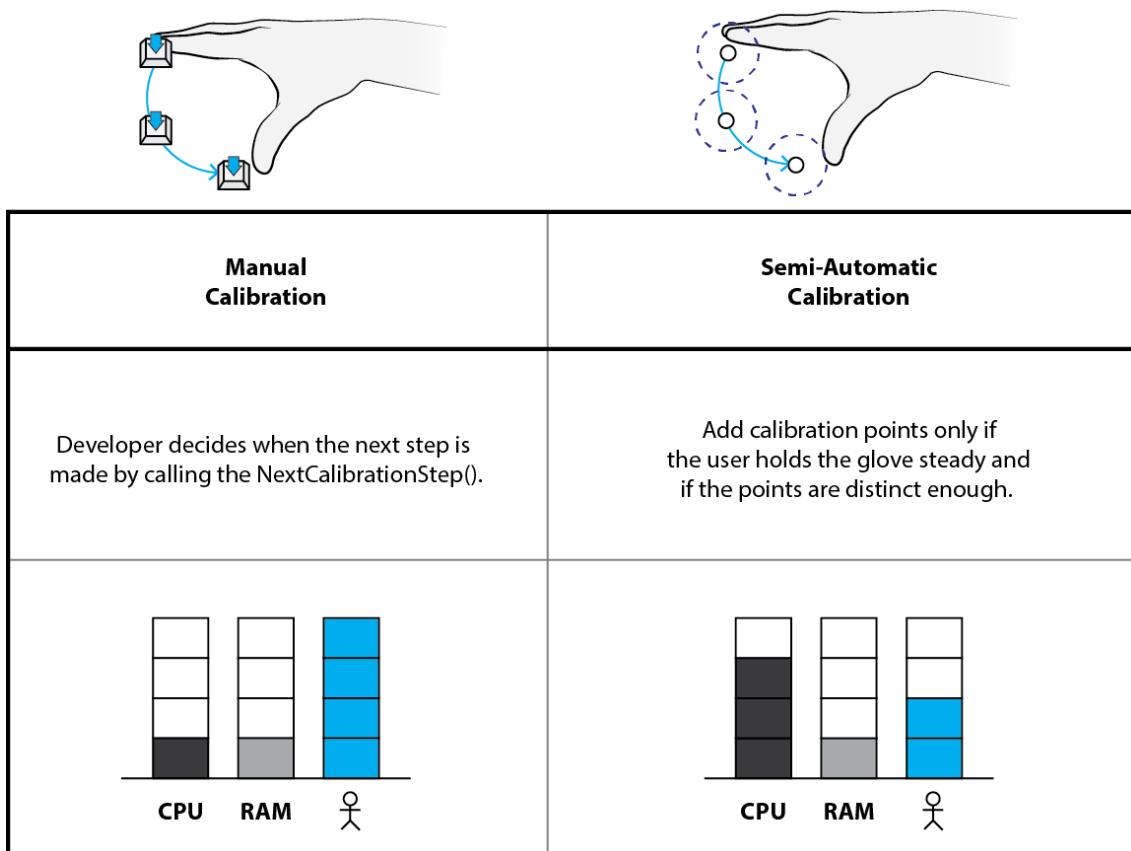


FIGURE 5.3 – Méthodes de calibration

Pour plus d'informations sur la calibration, veuillez-vous rendre sur [cette page](#).

Chapitre 6

Intégration

Ce chapitre décrit la mise en place des SenseGlove dans un projet Unity configuré pour la réalité virtuelle (utilisation du casque HTC Vive et des contrôleurs VIVE Trackers).

Comme décrit dans le chapitre 4, les SenseGlove ne dépendent d'aucun plugins tiers. 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 *SenseGloveSDK v1_0.unitypackage* téléchargé précédemment dans le projet en cliquant sur **Asset -> Import package -> Custom package**. À la suite de ça, le dossier Assets devrait contenir un dossier SenseGlove comme montré dans la figure ci-dessous.

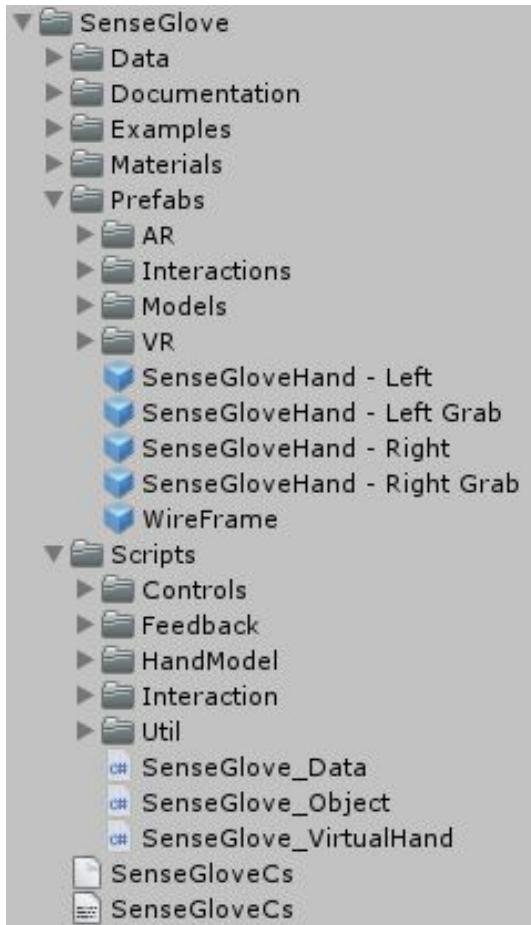


FIGURE 6.2 – Dossiers et fichiers contenus dans le SDK

6.2 Intégration de deux SenseGlove

Nous allons maintenant contrôler deux mains virtuelles avec les SenseGlove. Si vous souhaitez mettre en place votre propre modèle de main, veuillez suivre les démarches à [cette adresse](#). Pour ce document, nous utiliserons le modèle de main préfabriqué du SDK SenseGlove.

- Glissez-déposez les prefabs **SenseGloveHand – Left Grab** et **SenseGloveHand – Right Grab** 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 le VIVE Tracker est monté sur le gant.

Les prefabs ajoutés ont été créés et configurés pour fonctionner directement après leurs importations dans la scène. Nous retrouvons les scripts *_Object*, *_VirtualHand* et *_PhysGrab* qui sont ici pour la bonne connexion et intégration des gants au logiciel Unity.

En plus des scripts cités précédemment, le prefab contient le script *SenseGlove_KeyBinds*. Ce dernier est une classe qui permet la gestion du gant avec les touches du clavier. On y retrouve notamment les paramètres de calibration (variable et méthode de collection des *snapshots*).

La documentation du script *_KeyBinds* est disponible à [cette adresse](#).

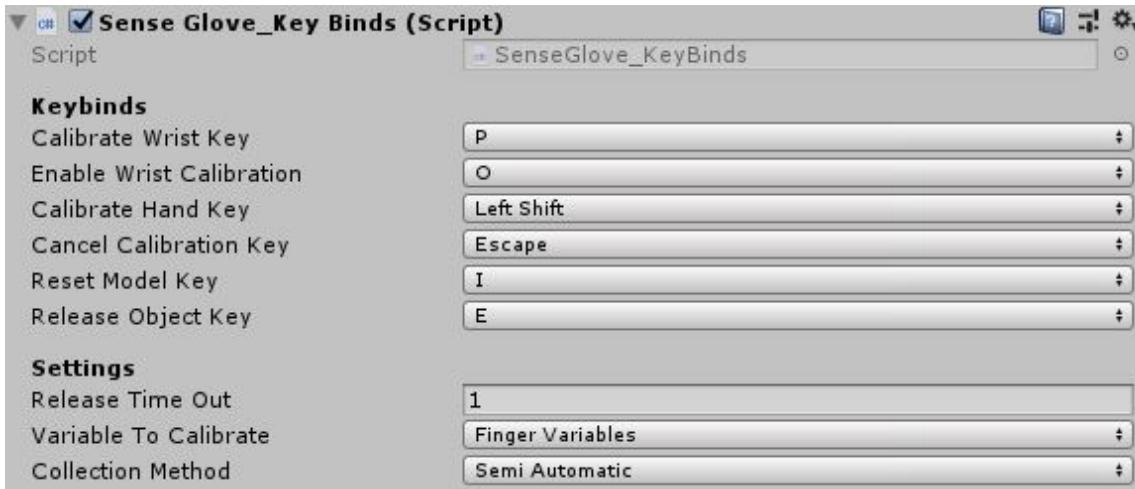


FIGURE 6.3 – Script *SenseGlove_KeyBinds*

- Pour finir l'intégration des gants SenseGlove, ajoutez le script *SenseGlove_DeviceManager* à la scène. Ce dernier peut être placé en tant que composant de n'importe quel objet de la scène. *_DeviceManager* ne demande aucune configuration, il détecte automatiquement les gants connectés à l'application.
- Une fois tous ces objets et scripts mis en place, lancez votre scène Unity.

Vérifiez que les mains virtuelles sont bien contrôlées avec les SenseGlove et que leurs positions correspondent à ceux de vos VIVE Trackers. Si la calibration du poignet et des doigts est imprécise, vous pouvez changer les paramètres de calibration des gants via le script *_KeyBinds*. Utilisez ensuite les touches de claviers assignées aux différentes méthodes de calibration pour configurer vos gants de manière optimale. N'hésitez-pas à ajouter plusieurs commandes *Debug.Log()* dans le script *_KeyBinds* pour afficher le nombre de *snapshots* correctement effectués.

Chapitre 7

Interactions

Ce chapitre décrit l'utilisation des scripts nécessaires aux interactions entre les objets virtuels et les gants SenseGlove. Les interactions sont créées en attachant des scripts aux objets concernés. Les scripts utilisés sont les suivants :

- **SenseGlove_Material**
- **SenseGlove_Grabable**

Comme décrit plus haut, ces script doivent être reliés à un gameobject pour mettre en place les interactions entre ce dernier et les gants SenseGlove.

7.1 SenseGlove_Material

Ce premier script permet de configurer les propriétés du matériau que nous souhaitons définir au gameobject auquel le script est attaché. Il se présente sous cette forme :

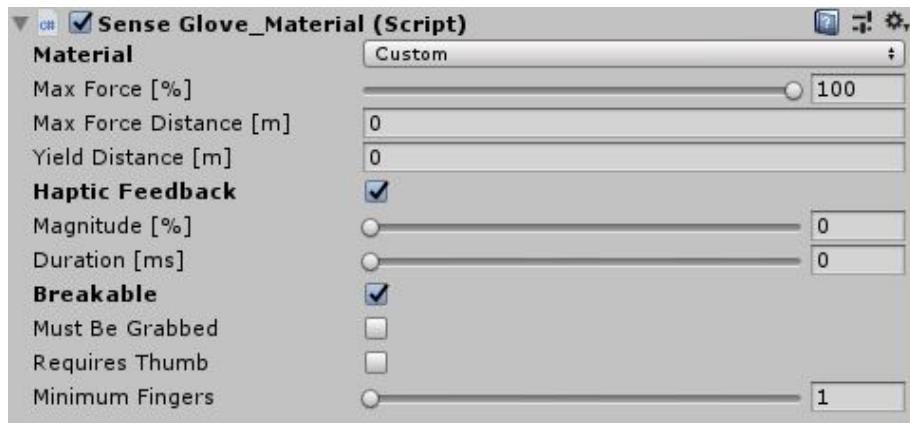


FIGURE 7.1 – Présentation du script *SenseGlove_Material*

Le premier paramètre défini le matériau à utiliser. Les valeurs possibles sont les suivantes :

- **Custom** : les propriétés du matériau sont assignées via l'inspecteur de Unity.
- **FomDataBase** : les propriétés du matériau sont chargées depuis un fichier texte.
- **Steel** : assigne les propriétés du matériau le plus dur.
- **Rubber** : assigne les propriétés d'un matériau souple (mou).
- **Egg** : assigne les propriétés d'un matériau souple que l'on peut casser.

Dans la plupart des cas, c'est la valeur custom qui est choisie car permet d'assigner des valeurs personnalisées aux propriétés du matériau dans le but d'obtenir un ressenti s'approchant le plus fidèlement possible de la réalité.

Les autres paramètres du script `_Material` sont décrits ci-dessous :

- **Max Force [%]** : la valeur maximale, de 0 à 100, du retour de force appliqué aux doigts qui sont à *MaxForce Distance* dans l'objet.
- **Max Force Distance [m]** : la distance, en mètre, à laquelle le retour de force de puissance *MaxForce* sera appliquée. Une distance de 0.01 signifie qu'un retour de force de valeur *MaxForce* sera appliquée aux doigts qui sont à 0.01 mètres à l'intérieur de l'objet. Cela permet de simuler les matériaux mous.
- **Yield Distance [m]** : cette propriété est la distance, en mètre, à laquelle le matériau se casse. Une modification de cette valeur n'est nécessaire que si la case *Breakable* de ce script est cochée.
- **Haptic Feedback** : si la case est cochée, le retour tactile des SenseGlove s'active pour les doigts entrant en collision avec l'objet. Il est possible de régler le l'ampleur et la durée de ce retour tactile, caractérisé par des vibrations.
- **Breakable** : défini si l'objet en question peut se casser ou pas.
- **Must Be Grabbed** : l'objet peut se casser uniquement s'il est ramassé (voir script `_Grabable`)
- **Requires Thumb** : nécessite l'utilisation du pouce pour casser l'objet.
- **Minimum Fingers** : le nombre de doigts requis pour casser l'objet

Les propriétés du script `_Material` doivent être réglées de façon à optimiser le ressenti de l'interaction avec l'objet et donc, maximiser l'immersion dans le monde virtuel.

7.2 SenseGlove_Grabable

Ce deuxième script permet à l'objet auquel il est attaché d'être ramassé par un script `_GrabScript`. Ce dernier est instancié par le biais de la classe `_PhysGrab`, elle-même liée à chaque SenseGlove.

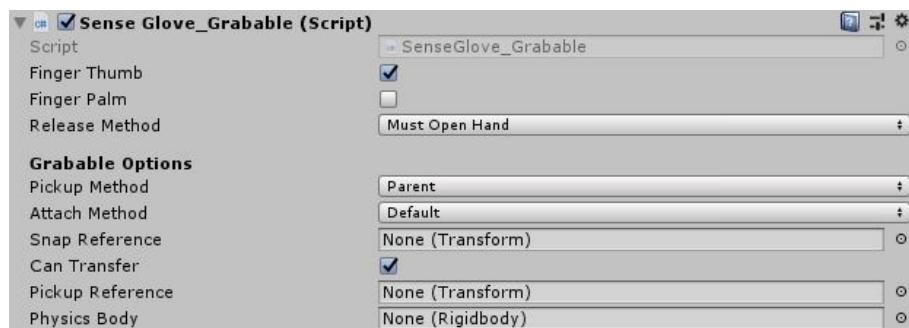


FIGURE 7.2 – Présentation du script `SenseGlove_Grabable`

Comme pour le script précédent, `_Grabable` possède différentes options de personnalisation.

- **Finger Thumb** : si activée, l'objet peut être ramassé entre le pouce et n'importe quel autre doigt.
- **Finger Palm** : l'objet peut être ramassé entre la paume de la main et n'importe quel autre doigt, y compris le pouce.
- **Release Method** : détermine la condition à remplir sous laquelle cet objet sera relâché
- **Pickup Method** : la façon dont l'objet est ramassé.
- **Attach Method** : la façon dont l'objet se lie à la main. Nous pouvons ici choisir une référence, *Snap Reference*, qui sera utilisée comme point d'ancrage de l'objet si ce dernier est ramassé. Si aucune référence n'est spécifiée, c'est le composant transform de l'objet qui sera utilisé.

- **Can Transfer** : si cette option est activée, l'objet peut être pris par un autre *_GrabScript* alors qu'il est déjà tenu.
- **Pickup Reference** : le composant transform qui sera utilisé par le *_GrabScript*. La valeur par défaut de cette option est le composant transform de l'objet lui-même.
- **PhysicBody** : cette option correspond au rigidbody auquel la vitesse et la gravité seront appliquées. Par défaut, le script essaiera d'attacher le rigidbody de l'objet en question.

Une fois ces deux scripts ajoutés puis configurés, l'interaction entre les mains virtuelles, contrôlées par les SenseGlove, et les objets virtuels en question sera mise en place.

Il existe de multiples autres scripts inclus dans le SDK, chacun ayant sa propre fonction. Plus d'informations à [cette adresse](#).

Chapitre 8

Remarques et problèmes connus

Cette section décrit toutes les remarques et problèmes connus concernant les SenseGlove.

- Si les scripts `_MeshDeform` et `_Grabable` sont attachés au même gameobject, la valeur de la propriété **Pickup Method** du script `_Grabable` doit être réglée sur **follow**. En effet, une autre valeur entraîne une déformation abusive de la texture et casse l'immersion.

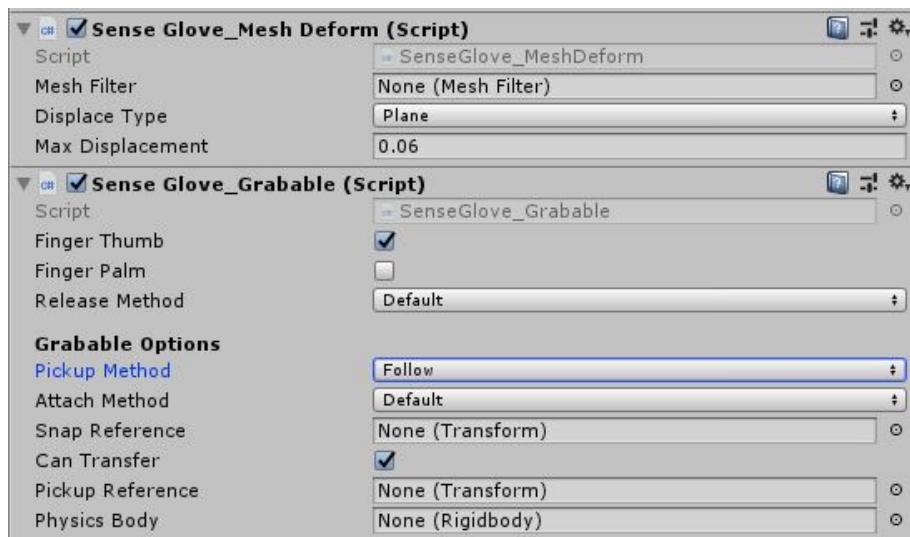


FIGURE 8.1 – Configuration du script `SenseGlove_Grabable`

- La calibration par la méthode **FingerLength** ne fonctionne pas. Lorsque les trois *snapshots* ont été capturés, rien ne se passe. La team SenseGlove a confirmé travailler sur ce problème. Il existe une autre méthode de calibration, toute aussi précise. Cette dernière nécessite la valeur **Interpolation_Flexion** pour calibrer en 2 étapes : main ouverte et poing fermée.

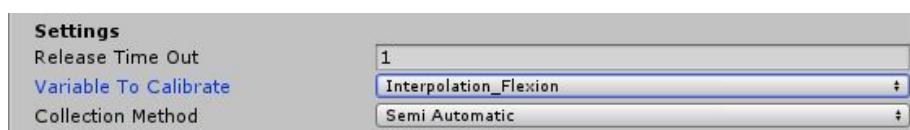


FIGURE 8.2 – Utilisation de la variable `Interpolation_Flexion` pour la calibration



FIGURE 8.3 – Premier *snapshot* de la calibration : mains ouvertes



FIGURE 8.4 – Deuxième *snapshot* de la calibration : poings fermés

- Par défaut, les particles system de Unity n’entrent pas en collision avec les doigts des prefabs SenseGlove. Il faut, à chaque bout de doigt, ajouter un 2ème gameobject qui possède les composants suivants :
 - Un collider (de type quelconque) avec la case **Is Trigger** décochée.
 - Un rigidbody avec la case **Is Kinematic** cochée.

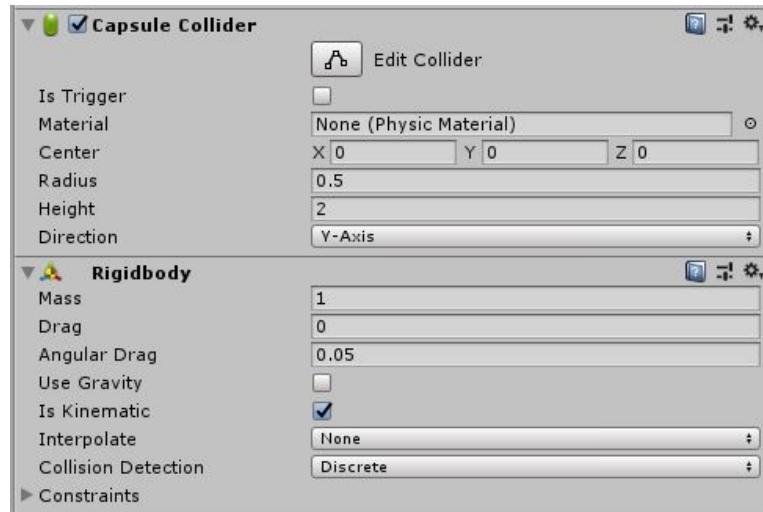


FIGURE 8.5 – Composants Capsule Collider et Rigidbody, utilisés pour la détection de collision avec les particules.

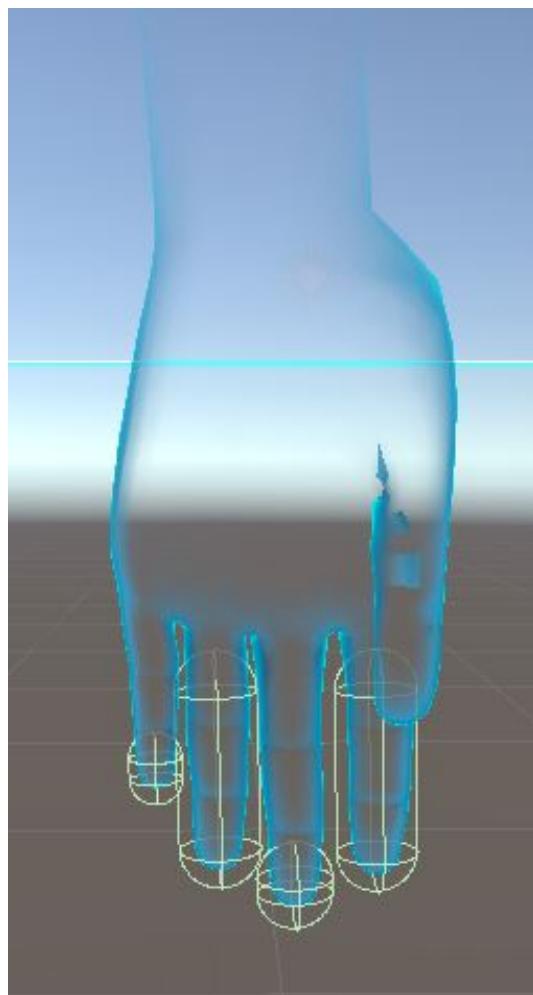


FIGURE 8.6 – Les colliders utilisés pour les interactions avec les gameobjects sont des sphères. Les colliders utilisés pour la collision avec les particules sont des capsules.

- Le chargement de la même scène SenseGlove dans une unique instance de l'application Unity provoque une erreur. En effet, le script *_DeviceManager* ne ferme pas correctement la connexion aux gants SenseGlove. La team SenseGlove a affirmé être en connaissance du problème. Ils sont actuellement en train de le corriger. Un patch, bientôt disponible en ligne, a été déployé sur l'application **TMS => Gants Haptiques**.
- Pour relâcher un objet, il faut ouvrir complètement la main. L'objet sera lâché lorsque tous les doigts seront en dehors du collider de l'objet. En d'autres termes, aucun doigt ne doit toucher l'objet pour que ce dernier soit libéré de la main. Ce problème entraîne un mouvement de lancer non naturel et difficile à prendre en main, surtout pour les grands objets. Les figures ci-dessous représentent les gestes à suivre pour effectuer un mouvement de lancer correct.



FIGURE 8.7 – Début du mouvement de lancer, l'objet est encore dans la main

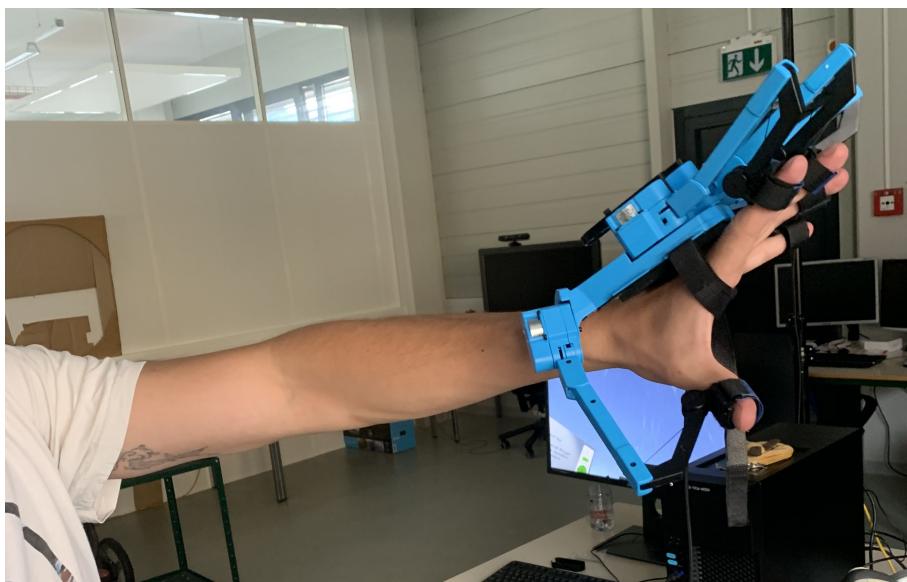


FIGURE 8.8 – Fin du mouvement de lancer, l'objet est alors relâché car les doigts ne sont plus en contact avec ce dernier