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

FALCON - DETAILS

Fait par :
Gabriel Griesser

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

Table des matières

Chapitre 1

Introduction

Ce document décrit, pour le gant SenseGlove :

- Sa description
- Ses propriétés et caractéristiques
- L'installation du driver
- La description du SDK et son architecture
- Le mode d'emploi d'intégration de deux joysticks Falcon
- Le mode d'emploi d'interaction
- Les remarques et problèmes connus

Chapitre 2

Description

Le joystick Falcon est un appareil à retour haptique développé par la firme Novint Technologies Inc. Ce périphérique haptique, branché en USB, possède un système de retour de force qui permet de ressentir la texture et la résistance des objets. Une autre propriété des Falcons est le ressenti du poids des objets.

Son utilisation passe par une boule, reliée au support par 3 tiges. C'est cette boule qui transmet à la main de l'utilisateur les efforts des moteurs.

Avec un temps de réponse rapide, un contrôle sur 3 degrés de liberté et des propriétés haptiques multiples, les Novint Falcons permettent de ressentir la taille, le poids, la forme, la rigidité et la texture des objets. Cependant, la forme du Falcon offre un ressenti et une immersion différente des gants haptiques.

Chapitre 3

Propriétés et caractéristiques

- **Type**
  - Joystick
- **Connexion**
  - Câblée
  - Driver à installer
- **Retour de force**
  - Oui, jusqu'à 8.9 Newton par joystick
  - Permet de simuler la texture d'un objet
- **Retour tactile**
  - Non
- **Suivi de mouvement (tracking)**
  - Boule : position
  - 3 degrés de liberté
- **Capteurs**
  - Aucun
- **Latence**
  - Aucune
- **Batterie**
  - Illimitée
- **SDK**
  - SDK Windows disponible
  - Aucun SDK Unity
  - Plusieurs librairies Unity sont disponibles sur internet. Ces dernières ont été développées par des utilisateurs.
- **Suivi de la team**
  - Aucun
- **Prix**
  - Joystick Falcon (1 joystick) : $200

Chapitre 4

Installation du driver

Le site web www.novint.com a fermé ses portes. Le driver du Novint Falcon est encore disponible à cette adresse.

Il faut installer le driver **setup.Falcon.v4.0.28.0_100707.exe** et le SDK pour Windows **HDAL_SDK_3.0.0_RC16_SPECIAL_setup.exe** contenus dans l'archive téléchargée.

Le document **Falcon_Quick_Start_Guide_R7.pdf** contient les démarches à suivre pour l'installation du driver et la première connexion au Falcon.

Afin que le plugin du chapitre suivant puisse être utilisé, la variable d'environnement *PATH* doit être modifié par l'installateur. Il faut donc cliquer sur le bouton *Yes (Recommended) Please make those changes for me* afin de mettre à jour la variable d'environnement *PATH* automatiquement.

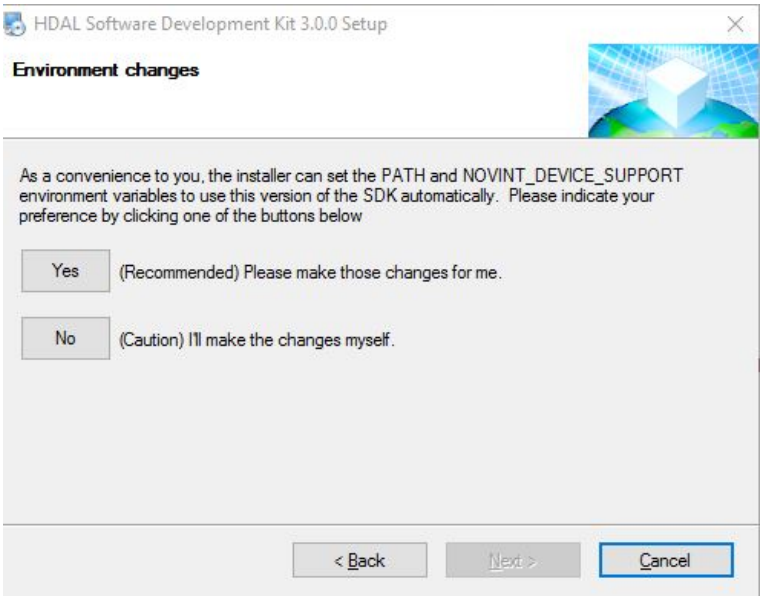

Figure 4.1 – Modification de la variable d'environnement *PATH*

Chapitre 5

SDK

Le joystick Falcon ne dispose d'aucun SDK officiel pour le moteur de jeu Unity. Cependant, une librairie permet la manipulation des objets 3D avec le moteur de jeu Unity. Ce plugin, développé par **Kenneth Bogert**, est disponible à cette adresse.

5.1 Architecture

Cette librairie utilise une connexion client-serveur pour reconnaître et utiliser les joysticks Falcons. Cette connexion doit être démarrée chaque fois que la scène Unity qui utilise le joystick Falcon est lancée. Le rôle de ce serveur est de créer un monde virtuel 3D (non visible) contenant les Falcon et les objets de la scène Unity avec lesquels les joysticks peuvent interagir.

Le monde virtuel du serveur est identique au monde virtuel de la scène Unity. Le premier monde virtuel permet de détecter les collisions entre les Falcons virtuels et les objets virtuels.

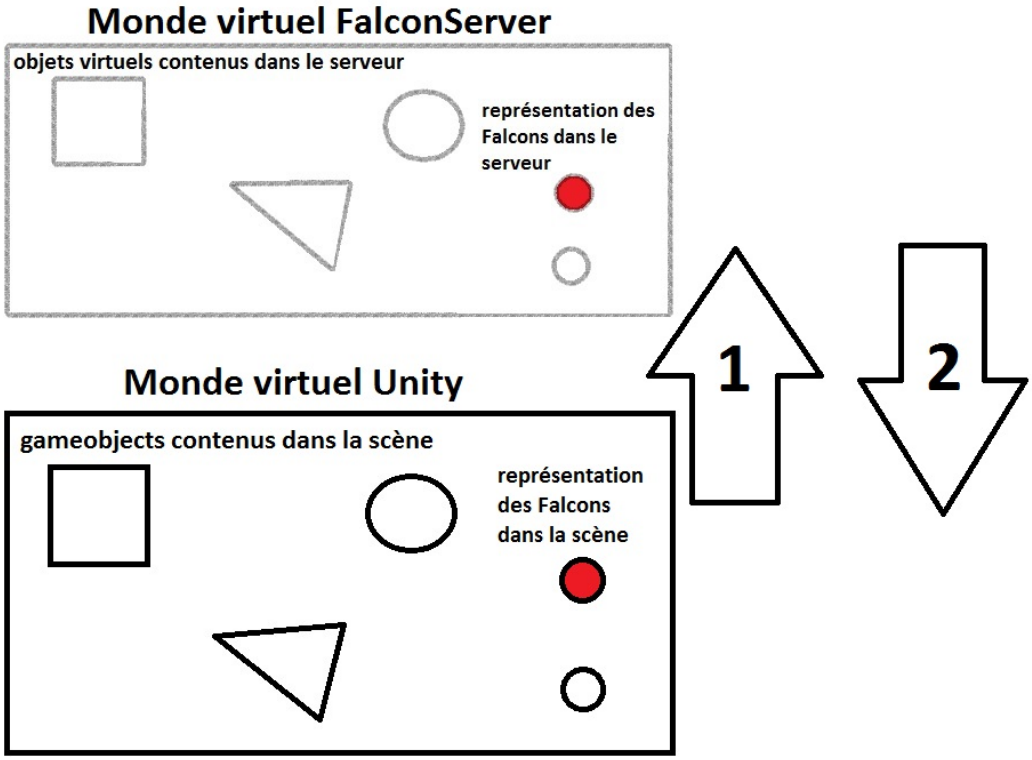

FIGURE 5.1 – Architecture simplifiée de la connexion client-serveur

— **Monde virtuel Unity** : création et instanciation des gameobjects et des Falcons de la scène Unity.

— **Flèche 1** : envoi des propriétés des gameobjects et des Falcons du monde virtuel de la scène Unity au monde virtuel du FalconServer. Les propriétés envoyées des gameobjects sont la taille, la masse, la friction, la position, la rotation et la vélocité. Les propriétés envoyées des Falcons sont la masse du Falcon et sa position dans le monde virtuel Unity.

— **Monde virtuel FalconServer** : création d'un monde virtuel 3D invisible, identique au monde virtuel de la scène Unity. Les propriétés des objets virtuels du FalconServer sont automatiquement mises à jour en fonction des propriétés envoyées des gameobjects contenus dans la scène Unity.

— **Flèche 2** : envoi d'un message lorsqu'un Falcon virtuel, du FalconServer, entre en collision avec un objet virtuel de ce même serveur. Ce message est alors récupéré par Unity, puis traité.

Chapitre 6

Intégration

Ce chapitre décrit la mise en place du joystick Falcon dans un projet Unity.

6.1 Setup de la scène

— Extrayez les fichiers, téléchargés sur le dépôt GitHub, dans le même dossier.

— Importez le package *FalconUnity.unitypackage* dans la scène Unity en cliquant sur **Asset ->** **Import package -> Custom package**.
À la suite de ça, le dossier Assets devraient contenir un dossier SenseGlove comme montré dans la figure ci-dessous.Le dossier Assets du projet doit être similaire à la figure ci-dessous :

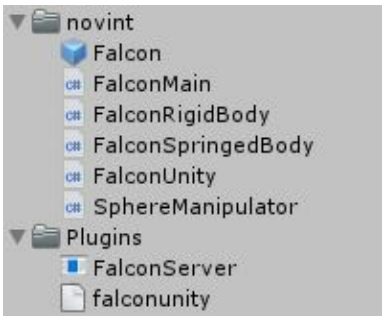

Figure 6.1 – Dossiers et fichiers contenus dans le SDK

6.2 Intégration du Falcon

— Glissez-déposez le prefab **Falcon** dans la scène pour chaque appareil utilisé.

— Démarrez l'application **FalconServer.exe**, puis lancez la scène.

— Si le serveur a bien été connecté au plugin Unity, qu'il a reconnu les Falcon utilisés, il devrait afficher ces lignes :

Toutes les méthodes de gestion des Falcons (connexion au serveur, mise à jour des objets virtuels, gestion des forces, etc.) sont définies dans la classe statique *FalconUnity*. Chacun des autres scripts s'occupe d'appeler les méthodes statiques de la classe *FalconUnity*, en adéquation avec le rôle du script appelant.

Par exemple : L'envoi des propriétés des objets virtuels de la scène Unity, au serveur FalconServer, se fait dans le script *FalconRigidBody* qui appelle la méthode
FalconUnity.updateDynamicShape(property A, property B, property C, etc.).

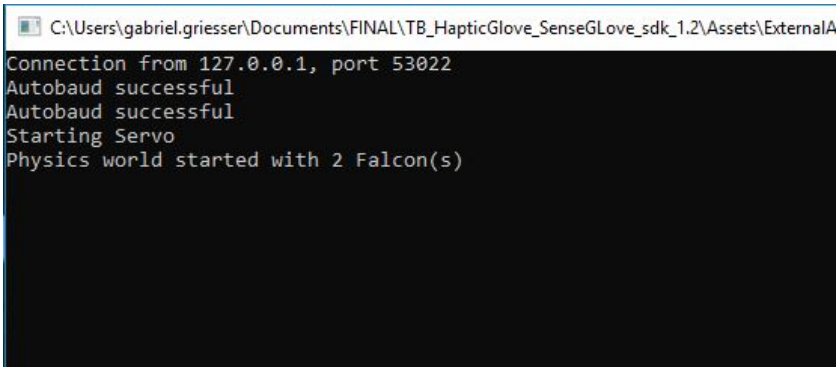

FIGURE 6.2 – Deux joysticks Falcon sont connectés au serveur FalconServer

Le script *FalconMain.cs* s'occupe d'instancier la connexion entre le serveur FalconServer et l'application Unity.

— Le script *FalconMain* doit être placé une seule fois dans la scène.

— Si vous utilisez 2 Falcons, il vous faudra alors modifier le prefab **Falcon** pour lui supprimer le script *FalconMain*. Attachez ensuite le script *FalconMain* à un objet de votre scène, évitant ainsi une duplication du script.

— Pour appliquer la force de gravité dans le monde virtuel, il vous faut ajouter la valeur -9.81 au champ Y de la propriété Gravity.

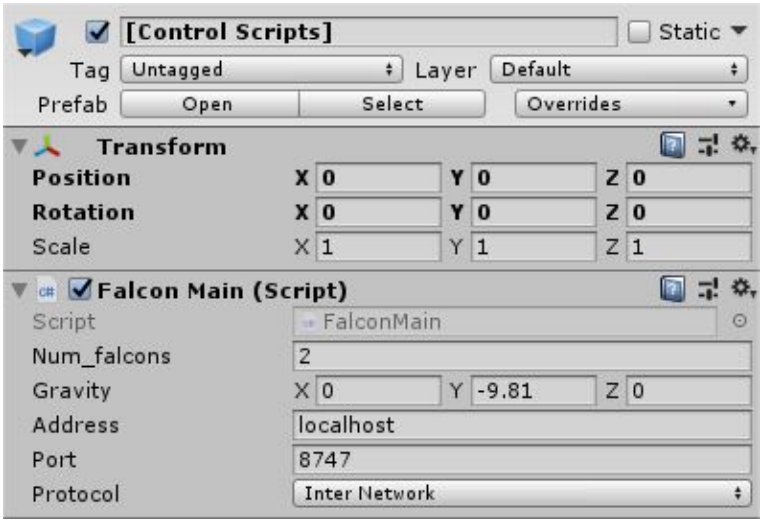

FIGURE 6.3 – Script *FalconMain* placé dans un gameobject **[Control Scripts]**

La figure suivante représente le prefab **Falcon**. Ce dernier a été modifié pour en faire un nouveau prefab : **Falcon_Left**. En effet, et comme décrit ci-dessus, pour utiliser 2 joysticks Falcons, il est nécessaire de modifier le prefab original **Falcon** pour lui enlever le script *FalconMain* afin d'éviter une duplication de ce dernier.

FIGURE 6.4 – Prefab Falcon gauche

Toutes les propriétés du script *SphereManipulator*, rattaché au prefab **Falcon**, sont à laisser par défaut pour éviter de casser l'immersion ressentie, exceptée la propriété **Falcon_num**. Le champ **Falcon_num** défini le numéro d'instanciation du Falcon. Si vous utilisez deux joysticks, le champ **Falcon_num** du premier prefab doit avoir la valeur 0. Le champ **Falcon_num** du deuxième prefab aura alors la valeur 1.

Chapitre 7

Interactions

Les interactions se font uniquement si l'objet virtuel est créé dans le monde virtuel mis en place par le FalconServer. Pour créer un objet virtuel dans le FalconServer, vous devrez lui attacher le script *FalconRigidBody* dans l'application Unity. C'est par le biais de ce dernier que les propriétés haptiques et physiques des objets virtuels de la scène Unity seront envoyées au serveur.

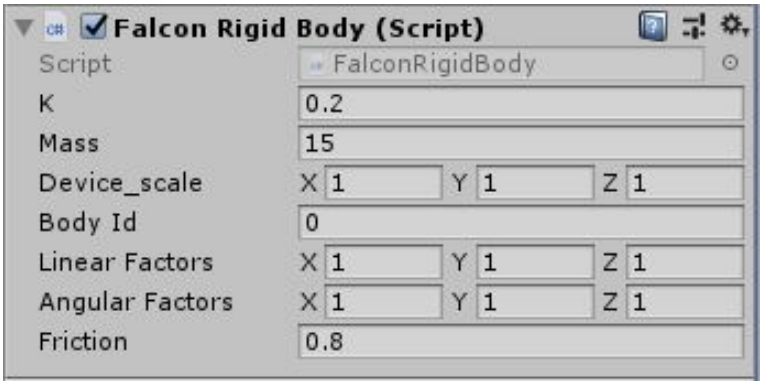

FIGURE 7.1 – Script *FalconRigidbody*

FalconRigidBody a pour rôle de lier, et de mettre à jour en temps réel, l'objet virtuel de la scène Unity auquel le script est rattaché, à l'objet virtuel du FalconServer.

Comme décrit dans la section 5.1, les interactions entre les Falcons et les objets virtuels de la scène Unity sont détectées dans le monde virtuel créé par le FalconServer. Ce dernier envoi l'information de collision à l'application Unity. L'information sera alors traitée puis reportée en temps réel sur le joystick Falcon.

Par exemple : Le Falcon virtuel pousse un objet virtuel de la scène Unity. Cette collision est donc détectée dans le FalconServer car il contient les mêmes objets, placés au même endroit. L'interaction est alors envoyée à Unity, qui déclenchera un retour de force sur le joystick Falcon en fonction des propriétés de l'objet virtuel touché (taille, forme, masse, friction). La méthode d'envoi du retour de force est définie dans la classe *FalconUnity*.

Chapitre 8

Remarques et problèmes connus

Cette section décrit toutes les remarques et problèmes connus concernant les joysticks Falcon.

— Le plugin utilisé ne permet pas d'interagir avec un objet virtuel composé de plusieurs gameobjects, placés en tant qu'enfants. En effet, le script *FalconRigidbody* requiert un mesh filter pour la forme complète de l'objet. Or, dans ce cas, l'objet virtuel est composé d'autant de mesh filter qu'il n'est composé de gameobjects, ce qui rend l'interaction impossible.

— Le prefab représentant le Falcon doit toujours avoir un scale de 1 / 1 / 1. Pour le redimensionner, il faut redimensionner ses gameobjects enfants **Tip** et **GodObject**.

— Il arrive parfois que le monde virtuel du FalconServer ne soit pas correctement initialisé. Une mauvaise instanciation du monde virtuel du serveur est caractérisée par :

  — Un joystick Falcon non connecté au serveur. Dans ce cas, la sphère blanche/rouge reliée au dispositif haptique non connecté ne bougera pas.

  — Ou alors un joystick Falcon connecté, mais un monde virtuel vide. Cela signifie que le monde virtuel de la scène Unity n'a pas été transmis correctement au FalconServer. Dans ce cas, les Falcons virtuels sont contrôlées par les joysticks Falcons, mais ni les forces, ni les collisions ne sont détectées.

  Si ce problème devait arriver, la seule solution disponible est de redémarrer l'application et le serveur FalconServer.

— Le plugin utilisé ne permet pas de détecter la pression et la libération du bouton en même temps. Le script *SphereManipulator* ne peut gérer qu'une seule action. Si le clic du bouton doit être détecter, il faut utiliser la fonction *ButtonPressed()* dans le premier test. Au contraire, pour détecter la libération du bouton, c'est la méthode *ButtonReleased()* qui doit être appelée en premier.

```
//go through the buttons, seeing which are pressed
for (int i = 0; i < 4; i++)
{
    if (button_states[i] && button_states[i] != curr_buttons[i])
    {
        ButtonPressed(i);
    }
    else if (button_states[i] && button_states[i] != curr_buttons[i])
    {
        ButtonReleased(i);
    }
    button_states[i] = curr_buttons[i];
}
```

FIGURE 8.1 – Ici, la pression d'un bouton sera détectée car la méthode *ButtonPressed()* est placée avant la méthode *ButtonReleased()*