ANNEE 2014-2015

Rapport de Projet - EI4 AGI

Virtual Platform

Projet réalisé par

Romain COUVET
Antoine JOUET
Audrey LEBRET

Projet encadré par

Paul RICHARD

ISTIA - Université d'Angers

Sommaire

Introduction

L'objectif de ce projet est d'améliorer le projet d'interaction existant, qui propose différentes tâches de manipulation d'objets dans des scènes en 2 ou 3 dimensions, à l'aide d'une ou plusieurs interfaces homme-machine spécifiques. Les fonctionnalités existantes seront revues quand nécessaire et de nouvelles tâches de manipulation d'objets seront proposées.

1. Présentation du projet

1.1. Cahier des charges

L'équipe est composée de trois personnes : Antoine JOUET, Audrey LEBRET et Romain COUVET. Le projet débute le 3 Décembre 2014 et se termine le 15 Avril 2015, avec un total d'environ 90h.

- Le matériel :

Le matériel à notre disposition est composé de : trois ordinateurs, un Razer Hydra (manette), un trackIR (capteur infra-rouge), un Leap Motion (détecteur infra-rouge), un Kinect (caméra) et un Myo (bracelet). Ce dernier est implanté dans le planning mais il sera réalisé uniquement si l'implémentation du Kinect, Leap Motion et les nouveaux exercices seront terminés.

- Le projet existant :

Le projet existant permet de manipuler des objets virtuels avec le Razer Hydra (périphérique composés de 2 manettes à champ magnétique), d'exporter dans une page Excel les données de mouvements (durée de l'exercice, suivi de la position).
Le capteur TrackIR permet de déplacer la caméra. Des réflecteurs sont placés sur une casquette portée par l'utilisateur, l'émetteur est placé sur l'écran en face de l'utilisateur.

- Points clefs :

- Améliorer l'interface du **menu**.
- Ajouter de nouvelles **tâches** de manipulation (clé 3D, suivi de poisson);
- Améliorer le programme du **Track IR**.
- Intégrer le **Leap Motion** au projet (communication avec le programme, manipulation des objets virtuels).
- Intégrer la **Kinect** au projet (communication avec le programme, manipulation des objets virtuels).
- Intégrer le **bracelet Myo** au projet (communication avec le programme, manipulation des objets virtuels).

- En détail :

- Le menu doit être refondu de manière à être plus ergonomique et plus clair, il doit également pouvoir prendre en compte les nouveaux appareils.

- Le Leap Motion doit être intégré pour pouvoir réaliser les tâches proposées suivantes : attraper, déplacer et faire pivoter des objets virtuels sans avoir besoin de manipuler un appareil physique. Il ne permet que de détecter le mouvement

des mains. Il nécessite seulement un petit capteur à poser sous la zone d'action des mains.

- Le Kinect doit être intégré pour pouvoir réaliser les tâches proposées suivantes : attraper, déplacer, faire pivoter des objets virtuels sans avoir besoin de manipuler un appareil physique. S'il permet de détecter l'ensemble du corps, nous allons utiliser un SDK permettant de ne détecter que les mains. Il nécessite une caméra, posée à proximité de l'écran.

- Le Myo, le bracelet connecté mesure l'activité électrique des muscles. De la même façon le Myo devra être intégré pour pouvoir réaliser les tâches proposées suivantes : attraper, déplacer, faire pivoter des objets virtuels sans avoir besoin de manipuler un appareil physique.

- L'implémentation de nouvelles tâches consiste à créer de nouveaux exercices pour l'utilisateur, lui permettant ainsi d'utiliser des mains de façon diverses.

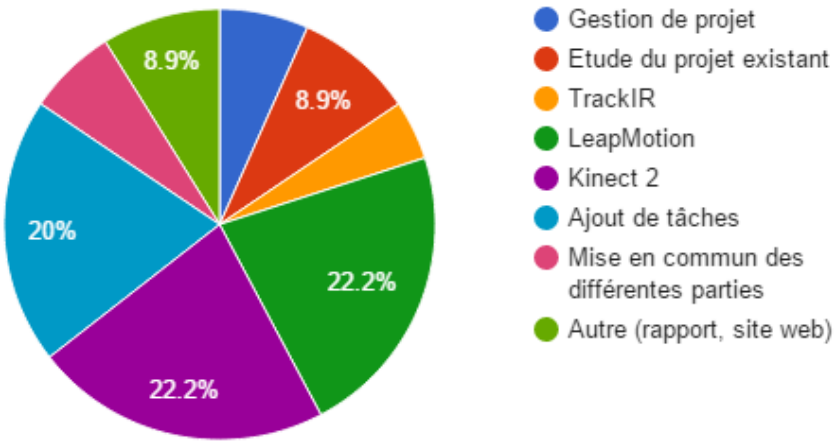
1.2. Gestion du projet

La gestion de projet est une partie importante et indispensable d'un projet. Elle permet de lister les besoins, les différentes parties, d'établir (approximativement) la charge de travail, de vérifier la disponibilité du matériel ...

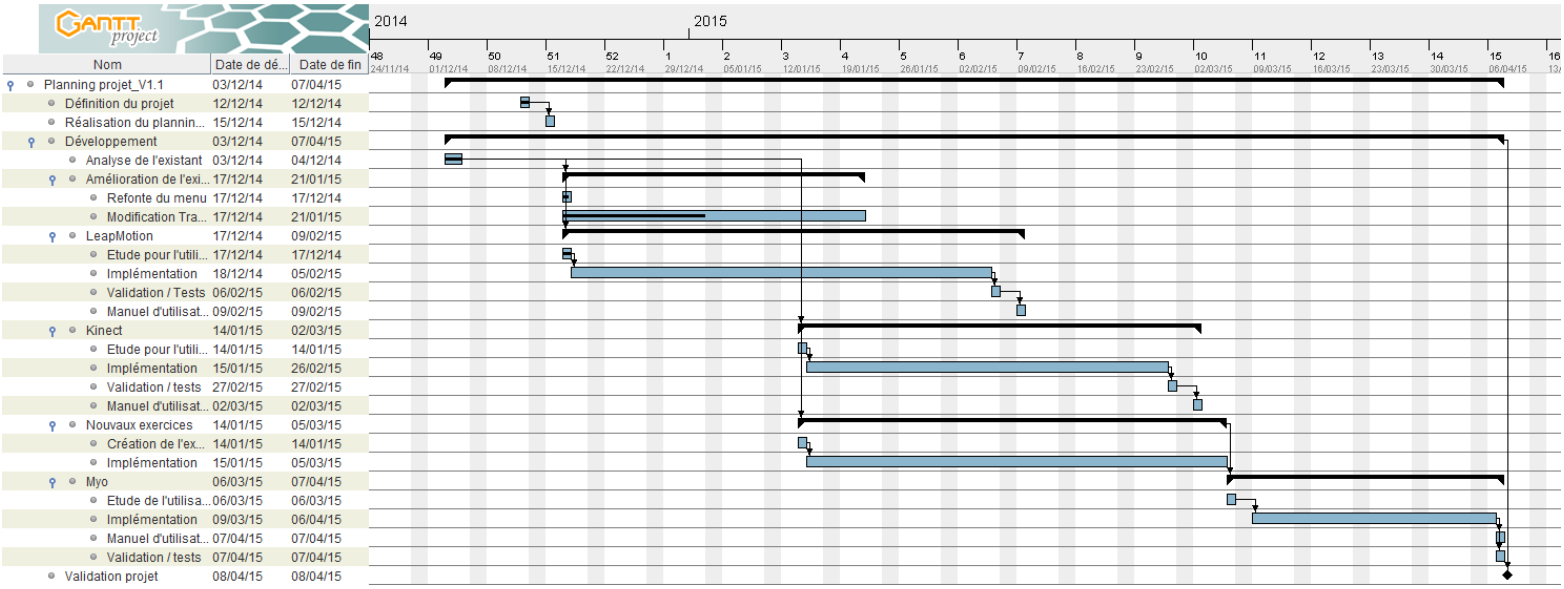
Afin de pouvoir travailler efficacement, nous avons décidés de diviser le projet en 3 parties principales que nous nous sommes partagées. Chacune d'elle étant réalisée en parallèle.
Après avoir pris connaissance de projet déjà existant, l'un d'entre nous s'est occupé de l'implémentation du Kinect2, le second, du Leap Motion et le dernier, de la création de nouvelles tâches.

On trouvera ci-dessous, un diagramme circulaire montrant la répartition des différentes activités ainsi qu'un diagramme de Gantt pour visualiser la planning du projet prévu initialement.

- Répartition du travail

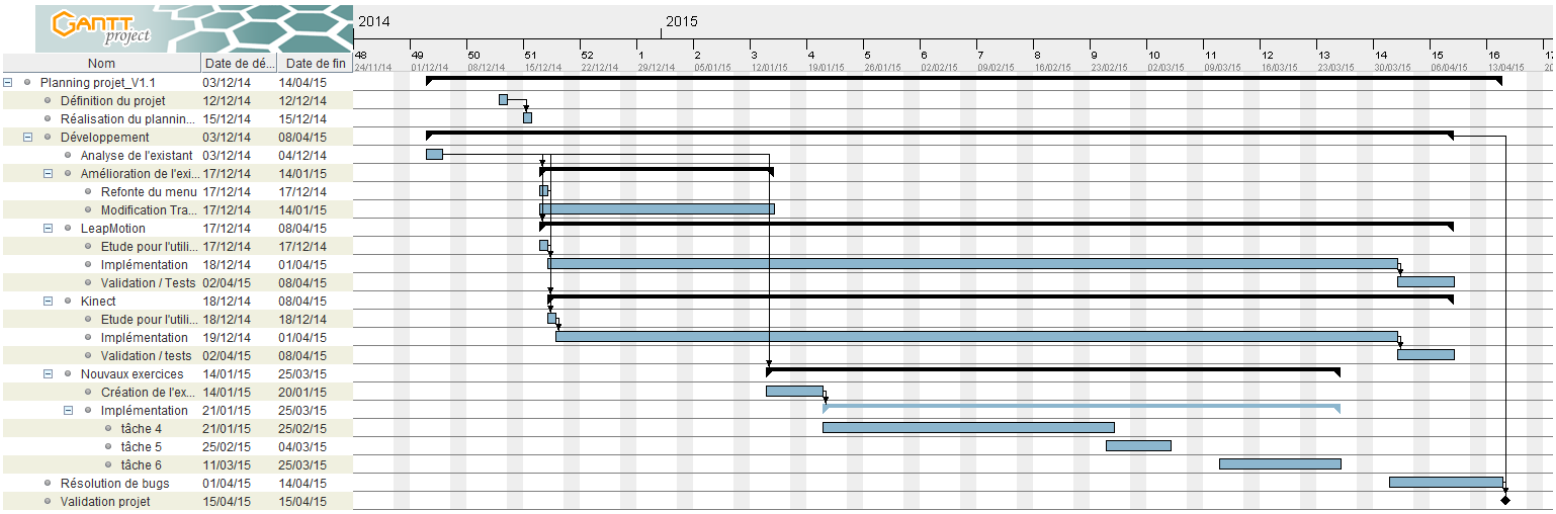


- Planning initial



- Planning réel

Comme nous avons pu le supposer dans le cahier des charges et lors de la gestion de projet, l'implémentation du Myo n'était pas assurée dans le cas où nous ne serions pas capable de développer les premiers objectifs à temps. C'est effectivement ce qui s'est passé.



1.3. Présentation du logiciel Unity

Unity 3D est un logiciel qui permet la création d'un environnement 3D en temps réel et contient un moteur physique utilisé pour la conception de contenus multimédias et interactifs comme les jeux vidéo, les animations…

L'intérêt principal de ce moteur graphique, par rapport à certains de ses concurrents, est sa facilité de prise en main. Le développement tourne autour d'assets (ressources utilisables, comme un objet 3D, une scène ou un script) qui s'utilisent les uns avec les autres.

Le développeur peut alors utiliser les assets déjà créés par la communauté Unity (mise à disposition sur un magasin en ligne) ou alors confectionner ses propres objets ou scripts.

Le contenu multimédia se décompose en scènes. Ces dernières peuvent être agrémentées d'objets 3D, de lumières, de terrains, de ciels,... On peut y ajouter des scripts pour effectuer des actions sur n'importe quel composant de la scène.

2. Les technologies

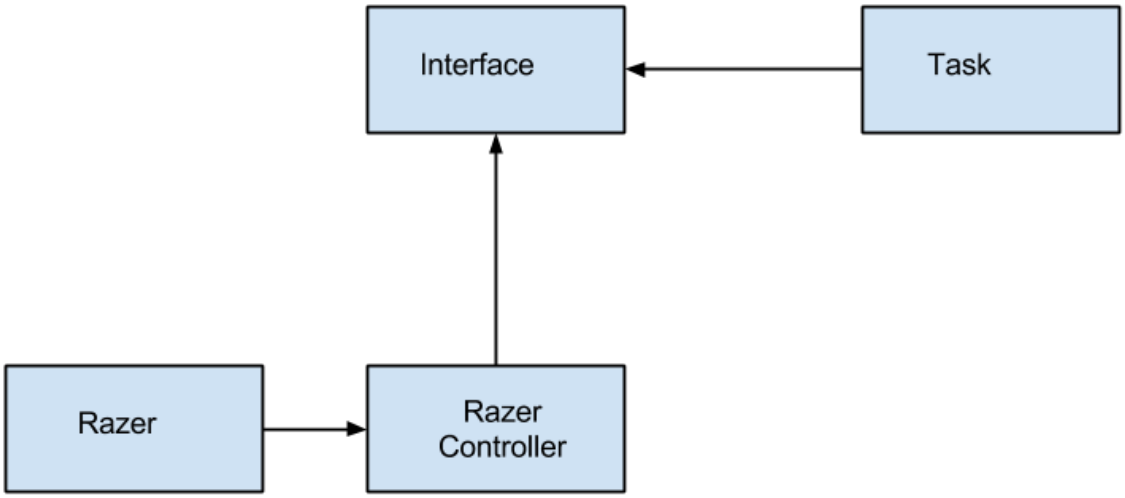
2.1. Le Razer Hydra

2.1.1. Présentation

Le Razer Hydra se présente avec deux manettes dotées de boutons et d'un stick (semblables à un mélange entre les deux contrôleurs de la Wii), reliées par un fil à une base. Celle-ci est capable de détecter la position et la rotation exacte des manettes par mesure d'un champ magnétique. Cette mesure est extrêmement précise (1 mm et 1° annoncé par le constructeur).
Les manettes ne sont toutefois pas équipées de vibreur.

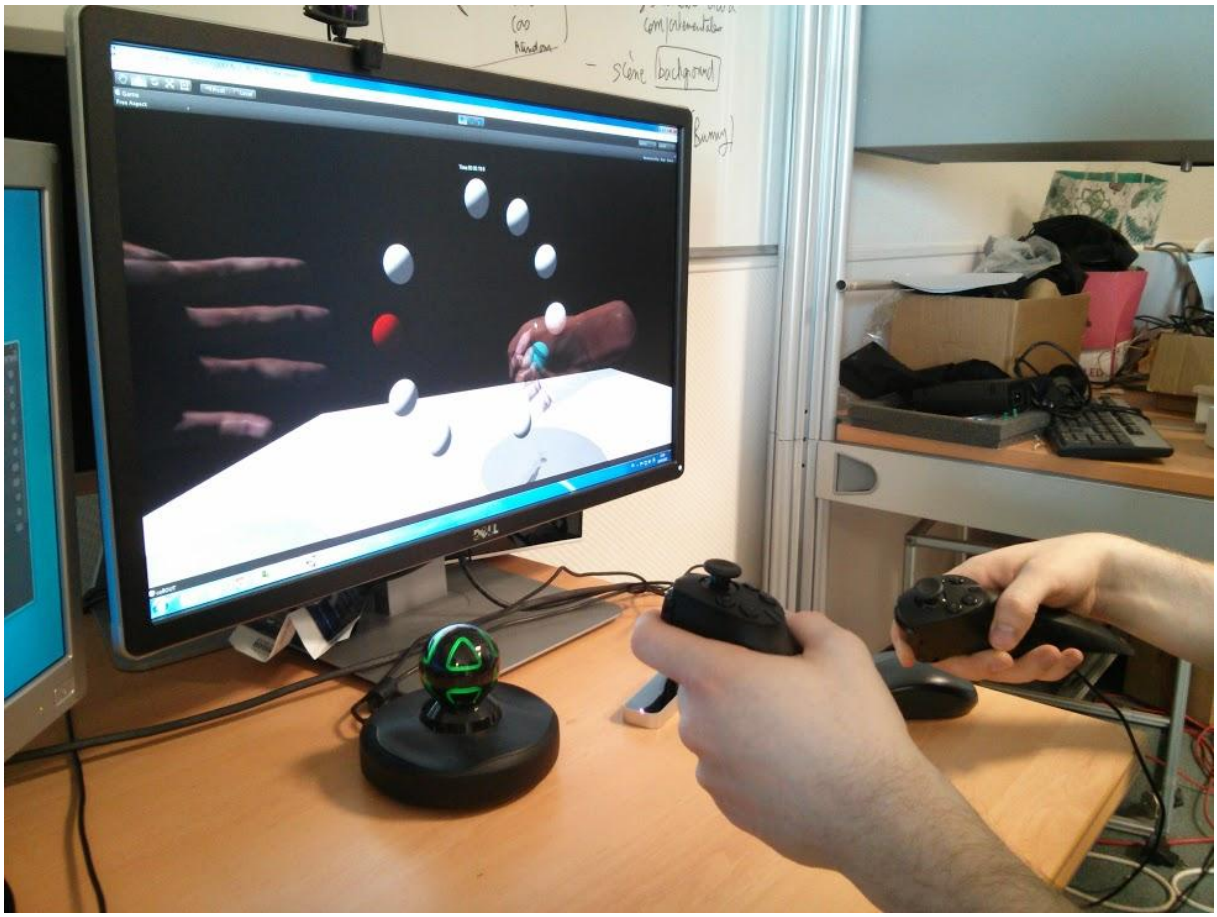
2.1.2. Programmer avec le Razer Hydra - Notre réalisation

L'implémentation du Razer Hydra avait déjà été réalisée l'année dernière par le groupe alors en charge du projet. L'interface est fonctionnelle sur n'importe quelle tâche, y compris sur les nouvelles ajoutées cette année. En effet, les tâches ne sont pas dépendantes de l'interface utilisée. Elles ne communiquent qu'avec une couche (classe) "intermédiaire" *Interface.*cs qui indique la position du ou des curseurs, et l'appui ou non sur les boutons (pour attraper ou détruire un objet par exemple). C'est cette couche intermédiaire qui dialogue avec l'Hydra.



2.1.3. Problèmes rencontrés

Nous n'avons eu aucun véritable souci avec le Razer Hydra. Le code déjà en place était propre et fonctionnel, et l'adaptation aux nouvelles tâches a été facile.



2.2. Le TrackIR

2.2.1. Présentation

Le trackIR, développé par NaturalPoint, est composé d'une caméra émetteur / récepteur infrarouge et de trois réflecteurs. La caméra émet elle-même de la lumière infrarouge, et localise la position et la rotation des réflecteurs qui renvoient cette lumière. Cela permet d'obtenir une gestion de la caméra virtuelle sans occuper les mains de l'utilisateur avec le clavier ou un stick de mannette par exemple.

2.2.2. Programmer avec le TrackIR - Notre réalisation

Cette fois encore, le trackIR avait été implémenté l'année dernière. Ainsi, l'utilisateur était en mesure de déplacer sa tête durant les tâches afin de jauger la distance des différentes pièces par exemple. En revanche, la rotation de la tête n'était pas gérée. Nous avons donc modifié le code originel afin de prendre en charge la rotation de la tête de l'utilisateur, sur les trois axes. L'implémentation du trackIR est relativement simple, un seul script *TrackIRCamera.cs* s'appuyant sur une dll suffit. Celui-ci se contente de lire les données envoyées par la caméra infrarouge et d'agir sur la caméra virtuelle en retour.

L'activation ou non du trackIR est possible via le menu de configuration des tâches. De plus, le lancement du logiciel nécessaire au fonctionnement du trackIR a été automatisé, ainsi que son extinction en fin de programme. Ainsi, l'utilisateur n'a pas à manuellement le démarrer avant de commencer l'utilisation.

2.2.3. Problèmes rencontrés

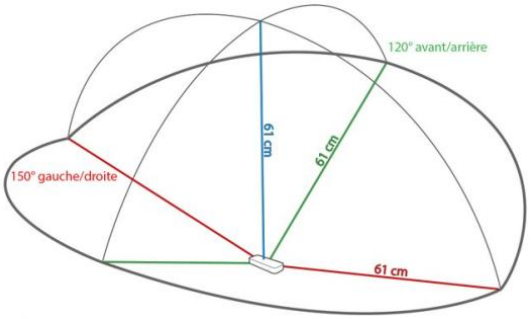
Dans l'ensemble, le trackIR fonctionne plutôt bien. On peut simplement noter que la caméra perd rapidement le contact avec la tête de l'utilisateur si celui-ci s'écarte un peu trop de l'écran. De plus, malgré plusieurs tentatives, il n'a pas été possible d'éteindre complètement la caméra lors de la fermeture du logiciel de pilotage. Celle-ci reste alors dans un état d'attente jusqu'à ce que l'utilisateur ouvre puis ferme à nouveau de logiciel manuellement, ou jusqu'à ce que l'ordinateur soit éteint. Dans tous les cas, cette situation ne pose aucun problème pratique.

2.3. Le Leap Motion

2.3.1. Présentation

Le Leap Motion est un appareil USB qui repère les mouvements des mains et des doigts dans l'espace avec une certaine précision. Il représente une manière différente d'interagir avec l'ordinateur, et permet de s'affranchir de tout contact physique entre l'homme et la machine. D'après ses concepteurs, il est 200 fois plus précis que n'importe quelle technologie similaire existante sur le marché. Le Leap Motion est également capable de déterminer la position des doigts au centième de millimètre et est capable de rafraîchir ses données à la fréquence de 200 Hz (contre 30 Hz pour Kinect).

Il est maintenant possible de contrôler un ordinateur en trois dimensions avec les mouvements de des mains et des doigts.



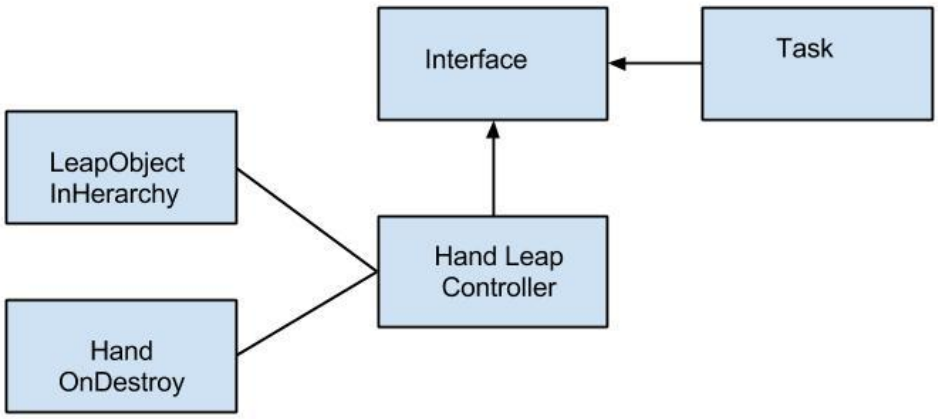
2.3.2. Programmer avec le Leap Motion - Notre réalisation

Le Leap Motion est une toute nouvelle interface dans ce projet. Elle était prévue pas les étudiants qui ont travaillé dessus l'an dernier et à été mise en place dans le menu mais n'a pas été implémentée.

Dans un premier temps, il était nécessaire d'installer le SDK pour pouvoir utiliser les librairies. A partir de là, on peut commencer à intégrer le Leap Motion au sein du logiciel Unity. Pour cela, on va se servir d'un asset Leap Motion présent sur l'Asset Store (www.assetstore.unity3d.com). A partir de ce dernier, on récupère les scripts, scènes et préfabs nécessaire à l'utilisation du projet.
L'implémentation du Leap Motion est principalement réalisée grâce à un script qui gère l'intégralité des "mains" et leur fonctionnement. Il permet de gérer l'aspect physique des mains (robot, humaine ...), différencier la main gauche et la main droite. C'est également dans ce script que l'on définit l'action sélection, lorsqu'on ferme la main.

Deux autres scripts on été ajoutés:
Le premier permet la création des mains sur la scène lorsque le Leap Motion est utilisé, mais aussi le déclenchement d'une "alerte" si l'appareil n'est pas détecté par l'ordinateur.
Le second permet de gérer la disparition puis la réapparition des mains dans la scène notamment avec l'utilisation de la fonction *OnDestroy()*.

Lors de l'utilisation du Leap Motion, 4 "objets" sont créés pour la scène : le *controller* des mains qui prend en paramètre leur modèle physique, la main en elle-même qui sera utilisée par le *controller* et "l'alerte" qui vérifie que le Leap Motion est bien détecté.

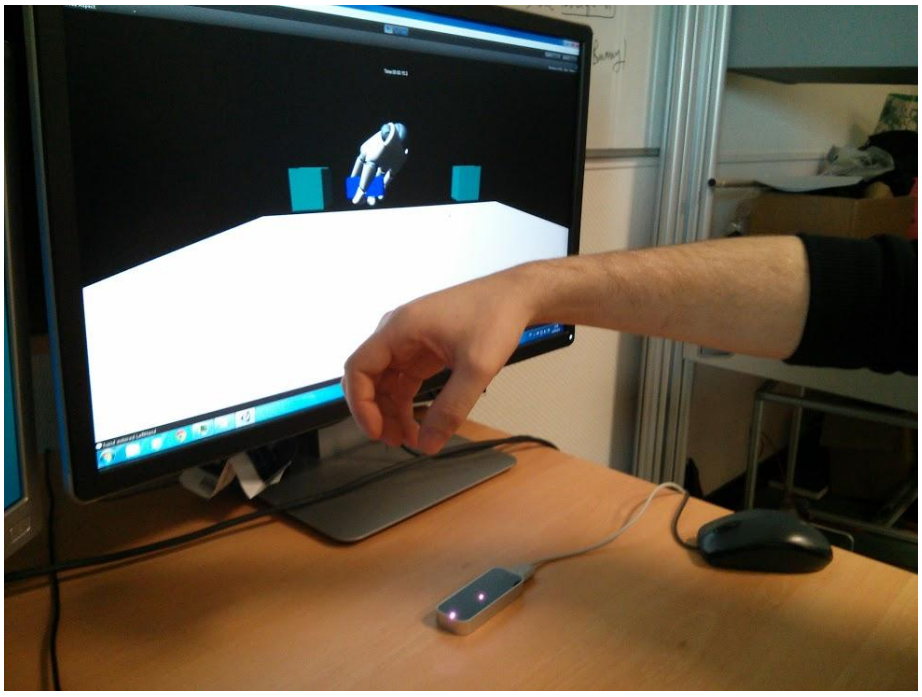


2.3.3. Problèmes rencontrés

L'installation et la prise en main du Leap Motion ont étés relativement facile à réaliser. Le principal problème rencontré a été de faire correspondre les mains apportées par le Leap Motion avec les fonctionnalités déjà présentent dans le projet. En effet, le projet était conçu de manière à pouvoir ajouter de nouvelles technologies tout en gardant les mêmes fonctions pour ne pas avoir besoin de trop modifier le code de base.

En début de projet, nous avons voulu remplacer les objets de base qui servaient de "mains" par les modèles de mains apportés par le Leap Motion, cependant beaucoup de problèmes de compatibilités sont apparus. Après plusieurs jours de recherche et grâce à un travail collaboratif, nous avons finalement trouvé une solution consistant à ne pas supprimer les objets "mains" mais à remplacer leurs coordonnées par celles des mains apportées par le Leap Motion.

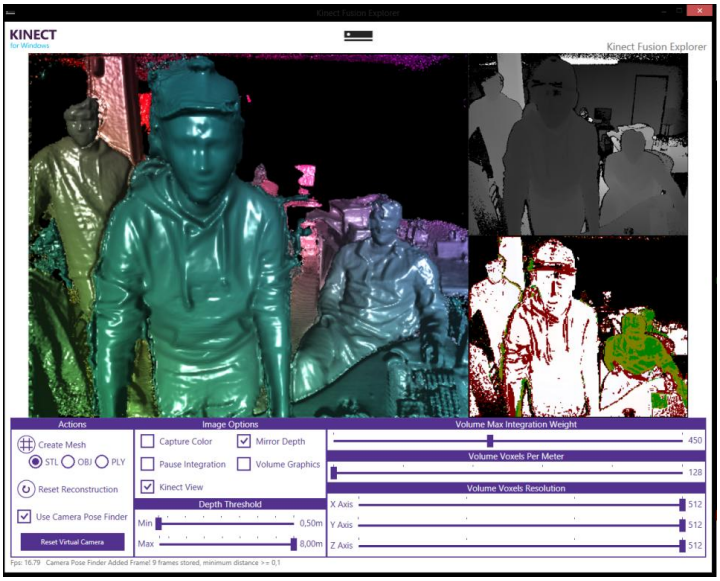
Au final, le Leap Motion est utilisable dans toutes les scènes et permet de réaliser les objectifs, même si son imprécision rend certaines tâches compliquées.



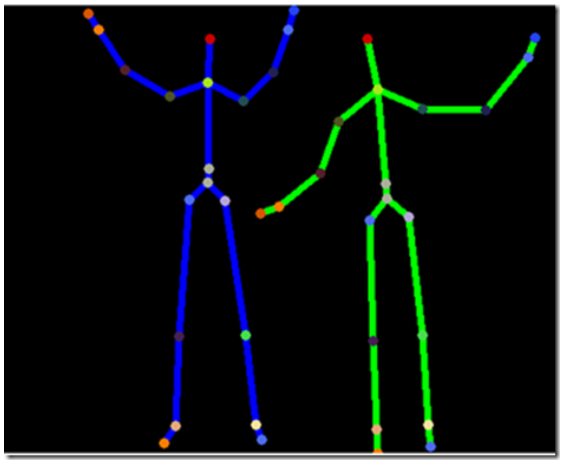
2.4. Le Kinect2

2.4.1. Présentation

Kinect2 désigne la deuxième génération de la caméra Kinect de Microsoft, destiné à la console de salon Xbox One.



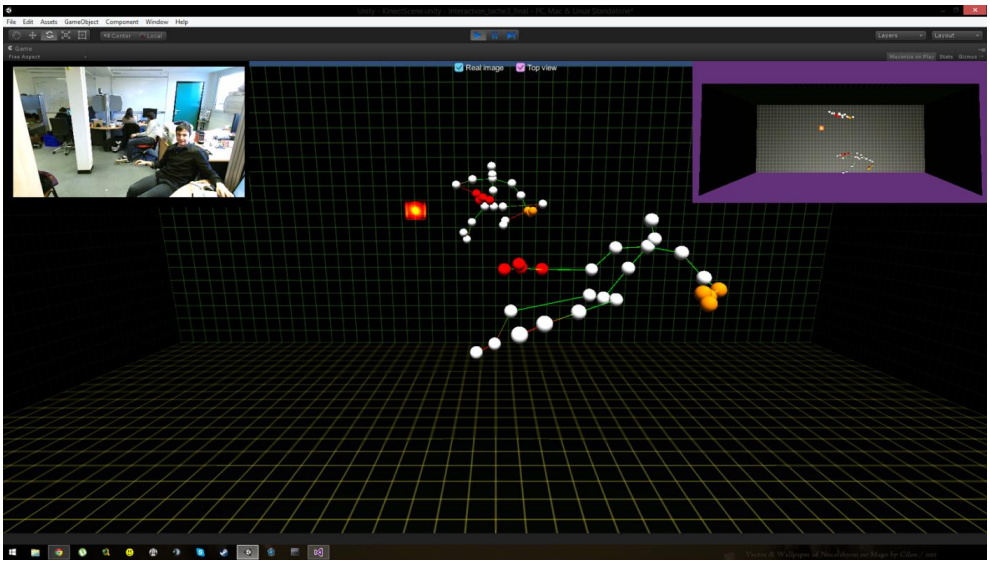
Kinect2 possède un puissant système de traitement d'image en temps réel. Image haute définition, caméra infrarouge, reconnaissance faciale, tracking de tête et même reconnaissance vocale sont embarqués dans cette barre horizontale de 25 centimètres. Principalement destiné à fonctionner avec sa console, Microsoft à tout de même développé un moyen de connecter l'appareil à un ordinateur et la communauté a rapidement produit une conséquente bibliothèque d'applications.



2.4.2. Programmer avec le Kinect - Notre réalisation

Nous avons commencé à travailler avec le Kinect 1, en utilisant le SDK 1 de Microsoft pour connecter le Kinect à l'ordinateur. Nous avons testé quelques fonctionnalités de base, puis il nous a été proposé de travailler avec le Kinect2, plus performant. Kinect2 est uniquement compatible avec le système d'exploitation Windows 8. Nous avons téléchargé le package Kinect2 for Unity Pro, sur le site de Microsoft. Ainsi nous avions

une scène Unity3D “exemple”, qui nous a servi de base pour comprendre le mécanisme. Le premier défi était donc de faire apparaitre ce que filmait la caméra dans une scène Unity. Ensuite, nous avions besoin de reconnaitre les mains du joueur. Le traitement d’image de Kinect2 est très performant et permet d’afficher ce qu’il appelle des jointures. La première scène créée a été complétement indépendante de notre projet Virtual Platform. Elle était nécessaire pour se familiariser avec les scripts et les classes spécifiques à la Kinect (langage C#).



Le squelette est respecté et nous avons ajouté une coloration aux éléments des mains (paume, poignet et index). La difficulté dans ce projet est que nous avons travaillé sur une base de code réalisé par un autre groupe. Il est donc judicieux de réutiliser les classes et les fonctions mises à notre disposition plutôt que de créer nous-même ce dont nous avions besoin pour assurer le fonctionnement de nos appareils respectifs. Dans Unity il faut utiliser les tags “LeftHand” et “RightHand” et les associer aux “mains” détectées par Kinect.

2.4.3. Problèmes rencontrés

Un problème qui n’est pas résolu est que lorsqu’un joueur détecté sort du champ de vision de la caméra, il n’est pas repéré lorsqu’il revient. Il faut réinitialiser la caméra en obstruant l’objectif quelques secondes.

Une difficulté nous a occupé se trouvait dans l’amplitude des mouvements et la position relative des mains virtuelles par rapport à la taille de la scène de la tâche. Il s’agissait en réalité d’une particularité d’Unity lorsque des gameobjects enfants se déplacent relativement à la position du gameobject parent (ici les mains sont objets enfants du corps, qui lui aussi se déplace).

Un autre problème qui a été réglé, était par rapport à l’*animator* des mains. Avec le Razer Hydra, fermer la main entraine une animation. Avec la Kinect, il est possible de détecter la fermeture de la main. Le problème était que l’animation était réinitialisée avant d’arriver à sa position de fermeture. La raison à cela était que la méthode était appelée dans 2 scripts. Il a suffi de supprimer les doublons pour résoudre ce problème.

Un autre point, nous n'avons trouvé aucun moyen de gérer la rotation des objets attrapés. Cela devrait être possible en utilisant l'alignement des doigts avec la main.

Enfin, Kinect2 est uniquement compatible avec Windows 8. Nous avons ajouté au menu une vérification qui permet de griser l'option Kinect si notre application est exécutée sur une version antérieure de Windows.

Tous les éléments nécessaire à l'utilisation sont générés et gérés par de script. Donc lors de l'ajout de nouvelles scènes, il est directement possible de réaliser l'exercice avec le Kinect (excepté un exercice nécessitant une rotation d'un objet).

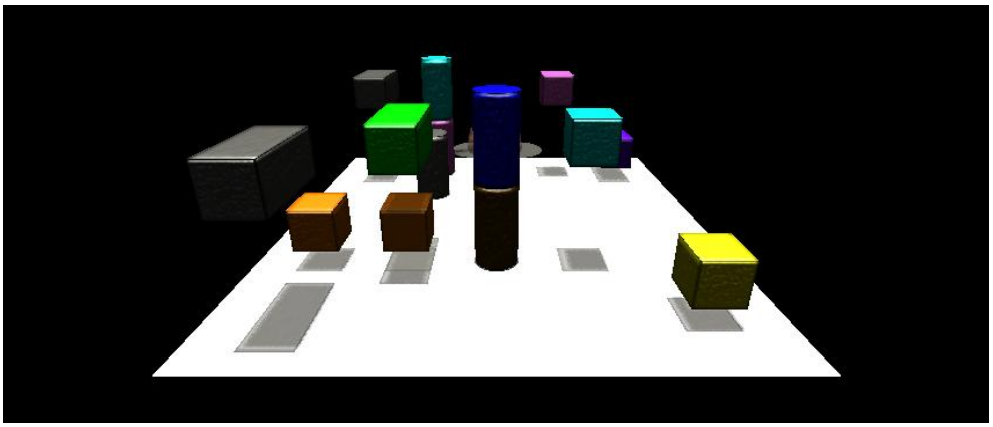
3. Réalisation

3.1. Menu

Nous avons conservé le menu réalisé par l'équipe de l'an dernier mais nous y avons apporté quelques modifications esthétiques.

En effet, nous avons amélioré la lecture du menu avec une meilleure visibilité des boutons lorsqu'ils sont sélectionnés. Nous avons également fait disparaitre certaine option lorsqu'elles ne sont pas utilisables avec une technologie.

3.2. Tâche 1



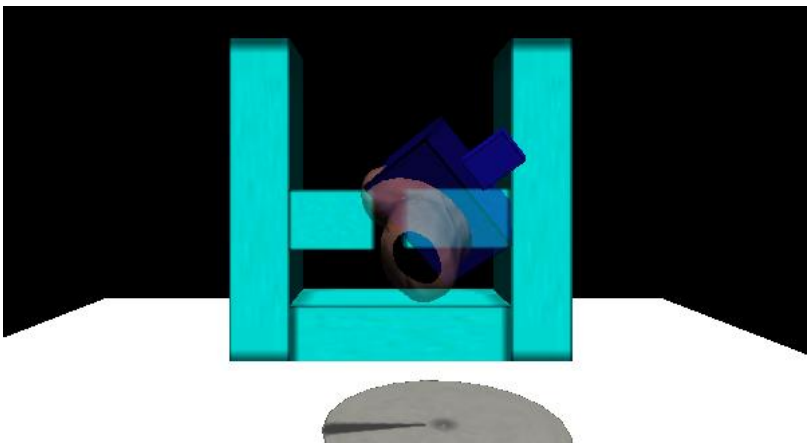
La tâche 1, implémentée l'année dernière, met en scène différents objets (parallélépipèdes, sphères) que l'utilisateur doit toucher à l'aide de son interface (Hydra, Leap Motion, Kinect) et détruire, dans un environnement en trois dimensions. L'épreuve met donc en application le déplacement dans une scène 3D, avec toutes les implications de problème de perspective pour l'utilisateur, qui peuvent être compensés par l'utilisation du trackIR. La destruction peut-être automatique dès le contact, ou bien déclenchée par l'appui sur une touche (Hydra) ou l'exécution d'un mouvement particulier comme serrer le poing (Leap Motion et Kinect). Cette option, ainsi que le nombre et le type de pièces souhaité, sont paramétrables via le menu de la tâche.

3.3. Tâche 2 & 3

Les tâches 2 et 3 dont nous avons hérité sont incomplètes voir presque pas développées. Nous les avons conservées dans le projet, mais dans l'état elles sont inexploitables.

3.4. Tâche 4

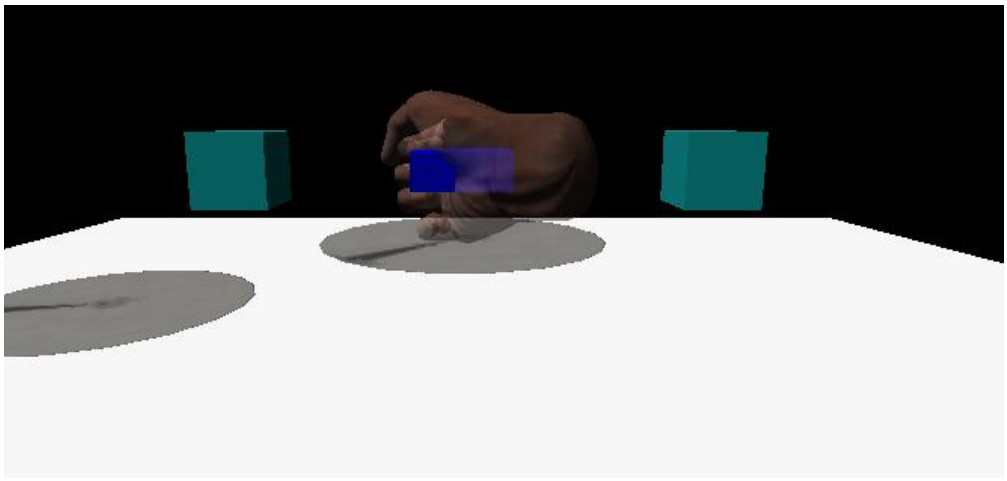
Cette quatrième tâche est conçue pour tester les capacités de visualisation, de déplacement et de rotation d'une pièce dans un environnement en trois dimensions. La scène est composée d'une pièce simple, la "clef", ainsi que d'une "serrure", un emplacement dans lequel il faut placer la clef, dans le bon sens.



En raison de la nature même de la tâche, il n'est pas possible d'utiliser le Kinect 2, qui n'est aujourd'hui pas capable de gérer la rotation de la pièce (cette limitation est liée à notre code et non pas aux capacités intrinsèques du système). De plus, il n'y a à l'heure actuelle qu'une seule clef et une seule serrure disponible. Il serait tout à fait possible d'ajouter des possibilités afin de varier les manipulations.

3.5. Tâche 5

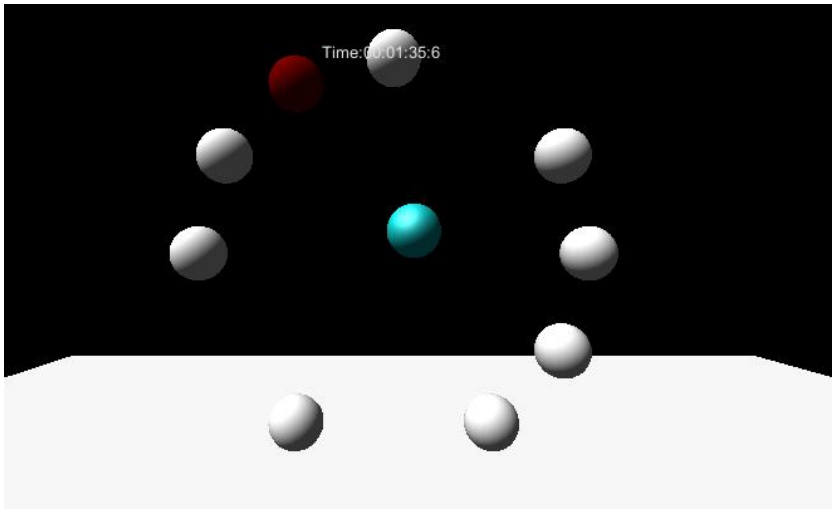
La tâche n°5 est une tâche basique centrée sur le déplacement d'une pièce sur une seule dimension.



Le but est simplement d'attraper la pièce, placée au point de départ, et de la faire translater jusqu'au point d'arrivée. Le mouvement n'est possible que le long de l'axe x, les axes y et z étant bloqués, tout comme les rotations. Une fois la pièce en contact avec la zone d'arrivée, la tâche se termine automatiquement.
Il est possible, via le menu de configuration de la tâche, de paramétrer la distance entre le point de départ et le point d'arrivée, afin d'étendre ou de réduire l'amplitude du mouvement nécessaire. Comme pour toutes les tâches, les mouvements de chaque mains sont enregistrés dans un fichier CSV, permettant l'étude à posteriori de l'épreuve.

3.6. Tâche 6

Dernière tâche implémentée, la tâche 6 est la suite logique de la tâche 5 : il s'agit cette fois de demander à l'utilisateur un mouvement de translation d'un point de départ à un point d'arrivée, mais cette fois sur deux dimensions.



La scène se compose d'un certain nombre de sphères placé en cercle, avec une dernière au centre de ce cercle. Cette sphère bleue est l'objet à manipuler par l'utilisateur. L'épreuve consiste à attraper la sphère centrale et à la déplacer jusqu'à la sphère cible indiquée en rouge, ce qui la détruira. Une fois ceci fait, une nouvelle sphère sera marquée en rouge, et ainsi de suite jusqu'à ce que toutes les sphères cibles aient étés détruites. La tâche est faite de telle manière qu'à chaque fois qu'une sphère est détruite, la prochaine à être marquée sera le plus possible à l'opposé, obligeant l'utilisateur à effectuer de grands mouvements.
Le nombre de sphères cible est paramétrable simplement via le menu, ainsi que leur taille.

4. Informations techniques complémentaires

4.1. Présentation du fichier Excel

L'exécution de l'application génère un fichier .CSV contenant les paramètres de la scène et les statistiques du joueur. La position des mains est récupérée toutes les 100ms. D'autres informations sont enregistrées comme le nom et le numéro du patient, le nombre d'objets virtuels de la tâche, l'appareil utilisé et la main utilisée.

Task1				
	Number Of Objects In Scene			8
	Time Before End Of Task			00h00m38s8
		Position	Position	Position
	Time(s)	X	Y	Z
	0	-0.07200002	0.05	0.1
	0.1	-0.05485715	0.03398364	0.09592617
	0.2	-0.04965296	0.03973608	0.09590232
	0.3	-0.05106563	0.05507802	0.09544694
	0.4	-0.04742187	0.05766099	0.09176362
	0.5	-0.05150887	0.06115012	0.08731401
	0.6	-0.07436104	0.07800178	0.08372223
	0.7	-0.1140852	0.105249	0.0847944
	0.8	-0.1775948	0.1558239	0.08746135
	0.9	-0.2163543	0.1873742	0.09089482
	1	-0.2415801	0.2009496	0.09413755

4.2. Comment ajouter une interface

Afin de pouvoir ajouter une interface, il faut tout d'abord ajouter son SDK que l'on peut télécharger sur le site du produit. C'est grâce au SDK que l'on va pouvoir utiliser les librairies dans le projet.

Une nouvelle interface homme-machine doit venir s'intercaler dans le système de couches existant : Une couche d'interface *Interface.cs* qui communique les actions effectuées par l'utilisateur aux scènes (déplacer ou attraper un objet, etc.), et une couche qui récupère les données brutes de l'appareil utilisé (Kinect, Hydra, Leap Motion) afin de les traiter et d'informer la couche d'interface des actions réalisées.
Ainsi, la scène n'a pas à se préoccuper de l'appareil utilisé, car tout ce qu'elle demande est la position des mains et l'action en cours (neutre, sélectionner, valider, récupérer la position).

Les fonctions sont les suivantes :

- setInterfaceInit(bInterfaceInit : bool) : void
  Permet d'initialiser l'interface.

- setSelectingLeft(bSelectingLeft : bool) : void
  Indique qu'une sélection (attraper un objet) a été réalisée de la main gauche.

- setSelectingRight(bSelectingRight : bool) : void
  Indique qu'une sélection (attraper un objet) a été réalisée de la main droite.

- setValidateRight(bValidateRight : bool) : void
  Indique qu'une validation (détruire un objet) a été réalisée de la main droite.

- setValidateLeft(bValidateLeft : bool) : void
  Indique qu'une validation (détruire un objet) a été réalisée de la main gauche.

- setPosition(v3Position : Vector3) : void
  Définit la position de l'interface.

4.3. Comment ajouter une tâche

Dans l'optique d'ajouter de nouvelles tâches à l'application, il faut bien comprendre l'architecture et la suite d'évènements qui conduisent l'utilisateur du menu principal aux différentes épreuves.

La première étape consiste à ajouter une tâche au menu principal. Pour ce faire, il suffit d'augmenter la capacité de la liste de tâches via l'inspecteur Unity. On peut ensuite ajouter une texture représentant la tâche.

Il faut alors éditer le script de configuration : celui-ci gère le menu de paramétrage que l'on retrouve avant le début de chaque tâche. Il est indispensable d'implémenter les éventuels paramètres nécessaires au bon fonctionnement de l'exercice que l'on souhaite ajouter (nombre de pièce, distance entre le début et la fin…), afin qu'ils apparaissent dans le menu. Il faut également indiquer au script de configuration quelle scène Unity il doit charger lorsque la nouvelle tâche est demandée.

La scène Unity, justement, doit comporter au moins une caméra "Main Camera", contenant un script lié à la tâche et à ses objectifs. Ce script, héritant du script *Task.cs*, initialise les objets à insérer dans la scène s'il y a lieu, et gère tout ce que l'utilisateur peut faire durant l'exercice (déplacer ou détruire un objet…). Il gère également les conditions de fin de tâche et entraine ensuite le menu principal.

Dans tous les cas, la scène sera automatiquement équipée des objets et scripts nécessaires au fonctionnement de l'interface utilisée (Hydra, Leap Motion, Kinect, trackIR). Le script de la tâche n'a pas à se préoccuper de l'outil utilisé, puisqu'il ne dialogue qu'avec une classe *Interface.cs*, qui lui transmet la position des différentes mains ou encore l'état des boutons. C'est la raison pour laquelle une tâche fonctionnelle sera directement opérationnelle avec n'importe quelle interface.

5. Conclusion

Nous avons dû très vite nous plonger dans un projet déjà existant et comprendre le fonctionnement de l'ensemble et des nombreuses classes en présence. Le fait que le projet ait été complexe, avec de nombreuses relations d'héritage entre les classes et une vraie organisation a été vraiment formateur.

Nous avons eu l'occasion d'améliorer la prise en charge du trackIR en ajoutant la gestion de la rotation de la tête. Nous nous sommes ensuite concentré sur l'ajout de nouvelles tâches au projet, ce qui nous à une nouvelle fois obligé à comprendre toute l'architecture du code, et a développé nos compétences relatives à Unity 3D, notamment concernant la gestion des collisions entre objet. Le fait que le Razer Hydra ait déjà été intégré nous a évidemment facilité la tâche puisque cette interface "traditionnelle" (manettes et boutons) était parfaitement stable.

L'utilisation du Leap Motion est très facile, mais son implémentation l'est beaucoup moins. Ce projet nous a donc permis de pouvoir mettre en œuvre cette nouvelle technologie. Dans toutes les parties et programmes qui compose ces "mains", il a fallu retrouver les bons scripts et les parties indispensable à leur fonctionnement. Une fois trouvées, il faut savoir les utiliser aux bons endroits et de la bonne manière avec les fonctions déjà existante du projet, ce qui n'est pas toujours simple. Savoir adapter une nouvelle technologie comme celle-ci à un projet déjà réalisé est une bonne expérience.

Pour intégrer le Kinect au projet nous avons utilisé le package Kinect for Unity distribué par Microsoft. Cette étape nous a permis d'établir une connexion entre la caméra et le logiciel Unity3D. Le Kinect étant capable de reconnaitre tout le squelette, il a fallu masquer les jointures qui ne nous intéressaient pas pour ne garder que les mains. Finalement les bases du projet étaient bien faites et le déplacement des mains virtuelles de la scène en fonction des mains réelles a été plutôt simple. Un problème restait cependant, dans l'amplitude des mouvements et la position relative des mains virtuelles par rapport à la taille de la scène de la tâche.

Au final, tout ceci nous a permis d'apprendre à analyser le projet et son environnement, l'utiliser et y ajouter de nouvelles fonctions. C'est un projet formateur qui devra sans doute être poursuivi dans les prochaines années, et qui a su nous apporter de nouvelles compétences et de la pratique bienvenue.

6. Résumé / Abstract

Le projet Virtual Platform propose 6 exercices de manipulations d'objets virtuels. Le projet initial (3 tâches et les interfaces Razer Hydra et trackIR) a été étendu afin de pouvoir utiliser 2 appareils supplémentaires: Leap Motion, Kinect2, ainsi que 3 nouvelles tâches.
Le Razer Hydra est une interface homme machine composé de 2 manettes dont la position en 3 dimensions est calculée avec précision grâce à un champ magnétique. Le Leap Motion est un dispositif capable de repérer une ou plusieurs mains et de représenter leur orientation dans l'espace par diffusion de lumière infrarouge. Le Kinect2 est une caméra équipé d'un système performant de traitement d'image capable d'identifier une personne et de faire coïncider ses mouvements à ceux d'un personnage virtuel. Enfin le TrackIR est un système composé d'un émetteur récepteur et de réflecteurs fixés sur une casquette utilisé pour obtenir la position de la tête de l'utilisateur et orienter la caméra virtuelle dans la direction correspondante.

Les données des 6 tâches (position des mains dans le temps) sont enregistrées dans un fichier CSV, ces résultats permettront de comparer les performances des sujets dans chaque exercice de manipulation ainsi que la facilité d'utilisation de ces interfaces homme machine en fonction de la situation.

The Virtual Platform project features 6 virtual objects manipulation exercises. The initial project (3 tasks and the devices Razer Hydra and trackIR) has been extended in order to use 2 new devices: Leap Motion, Kinect2, and 3 new tasks.

Razer Hydra is a human machine interface made of 2 controllers for which the virtual position is calculated with great accuracy thanks to a magnetic field. Leap Motion is a device able to spot a hand and represent its spatial orientation with infrared light. Kinect2 is a video camera equipped with a powerful image processing system that can identify a person and match his movement with the ones of a virtual avatar. Finally, TrackIR is an emitting / reflecting / receiving system which can detect the user's head position and adjust the virtual camera to the corresponding direction.

The results of the 6 tasks (position of the hands in time) are saved as a CSV file, those results will be used to compare the performances of the players in each exercise as well as the handiness of all these human machine interfaces according to the situation.

7. Bibliographie - Webographie

- https://www.leapmotion.com/
- http://www.clubic.com/technologies-d-avenir/article-575170-1-leap-motion-test.html
- http://docs.unity3d.com/
- http://www.microsoft.com/en-us/kinectforwindows/develop/downloads-docs.aspx
- http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft_Kinect_-_Microsoft_SDK

8. Annexes

Manuel d'utilisation

Installation du Razer Hydra :

Afin d'utiliser le Razer Hydra dans l'application, il suffit simplement de brancher l'appareil à l'ordinateur et de se laisser guider par Windows. Au besoin, il est possible de télécharger le driver spécifique sur le site de Razer :

http://drivers.razersupport.com//index.php?_m=downloads&_a=viewdownload&downloaditemid=678&nav=0,166,181

Installation du trackIR :

Le trackIR nécessite un logiciel propriétaire pour fonctionner : celui-ci est disponible sur le site de NaturalPoint :

https://www.naturalpoint.com/trackir/06-support/support-download-software-and-manuals.html

Dès le lancement d'une tâche, le logiciel trackIR sera automatiquement démarré afin de permettre le fonctionnement de l'appareil.

Installation du Leap Motion :

L'utilisation du Leap Motion nécessite l'installation du logiciel propriétaire afin de pouvoir fonctionner correctement. Il est disponible sur le site de Leap Motion à l'adresse suivante :

https://www.leapmotion.com/setup?lang=fr

En plus d'installer les éléments nécessaire au fonctionnement du Leap Motion, le logiciel vous proposera quelque jeux afin d'apprivoiser l'outil.

Installation du Kinect2 :

L'installation du Kinect2 n'est possible que sur les ordinateurs fonctionnants sous le système d'exploitation Windows8 ou plus recent (Windows 8.1 et Windows 10). Les pilotes se trouvent dans le SDK disponible à l'adresse suivantes:

http://www.microsoft.com/en-us/kinectforwindows/develop/downloads-docs.aspx

Une fois installé, il faut exécuter Kinect Configuration Verifier, pour s'assurer que le Kienct2 est prêt à être utilisé.