**[PREPARER LE LEAP MOTION VISUALIZER]**

**[PREPARER FINGER’S CLONER]**

# Présentation

## Sommaire

* Introduction
* Leap Motion
* Principales fonctionnalités
* Méthodes mises en œuvre
* Démonstration
* Conclusion
* Questions

## Introduction

### Le projet

Créer un logiciel d’apprentissage du langage des signes ou de rééducation de la main.

### Les objectifs

À l’aide d’un capteur de mouvement, interpréter une position de doigts, l’afficher, l’enregistrer la comparer à une position préenregistrée avec un niveau de précision et un affichage couleur pour rendre les différences avec le modèle visuelles.

### Les solutions possibles

Sans passer par un capteur de mouvement tel que le Leap Motion, il serait possible de créer un gant « intelligent » capturant les positions et le mouvement des doigts.

C’est ce que « SignAloud » a fait en récupérant les données des différents capteurs du gant. Les données sont envoyées par Bluetooth, puis elles sont interprétées et enfin traduite sous forme de phrase qu’il est possible de faire lire à haute voix par une voix synthétique.

## Leap Motion

Le Leap Motion est un tracker de main. Grâce à trois LEDs infrarouge et deux cameras, il détecte la paume, les doigts et chacune des phalanges. Par exemple, il peut détecter si les doigts sont pliés ou non, ou encore s’il y a un « pincement » entre deux doigts. Avec les données reçues, et avec son intégration de réalité virtuelle, il localise la position de la main dans l’espace situé au-dessus ou face au capteur.

D’un point de vue logiciel, le service du Leap Motion va analyser les données reçues, puis il ignorera tout ce qui ne concerne pas la main directement, par exemple s’il voit une tête. Une fois ce premier nettoyage effectué, le service va tenter de construire un modèle 3D de la main.

Pour que ses données soient plus précises, il faut lui donner une « Interaction Box », un espace tridimensionnel dans son champ de vision qui va capter toute son attention en ignorant ce qu’il y a en dehors de cet espace. Le centre du cube représente le point {0 ; 0 ; 0} des axes X, Y et Z. Les positions qu’il enverra à l’ordinateur peuvent être normalisées, ainsi, plutôt que d’avoir des coordonnées réelles du Leap Motion, il n’y aura que des coordonnées relatives, situées entre -1 et 1, sur les trois axes.

**[MONTRER LE LEAP MOTION VISUALIZER]**

## Principales fonctionnalités

Le programme doit montrer la main de l’utilisateur sur l’écran. Quand il position ses doigts dans une position souhaitée, il peut l’enregistrer. Une nouvelle fenêtre s’ouvre lui permettant de visualiser sa position au moment où il a cliqué sur « Enregistrer ». Il doit la nommer, la décrire, et il a la possibilité de charger une image, de la position actuelle de préférence, pour rendre le tout plus clair.

Toutes les positions enregistrées sont listées et l’utilisateur peut choisir laquelle il veut tenter de copier. Sur l’affichage, des couleurs (verte, orange, rouge et noir) lui indiquent pour chaque doigt s’il est prêt de la position du même doigt du modèle.

## Méthodes mises en œuvre

### Logiciels utilisés

* Le SDK du Leap Motion est obligatoire pour pouvoir utiliser le Leap Motion, mais il offre d’autres services tels qu’un visualiseur, qui permet de voir ce que filme le capteur et la/les mains qu’il détecte ou encore une page JavaScript qui affiche chaque donnée des doigts et de la main.
* Visual Studio 2017 comme environnement de développement qui permet de créer des programmes « Windows Form » facilement, ce qui permet de ne pas s’attarder sur le visuel. Il assiste les développeurs dans les phases de codage en suggérant les objets, méthodes ou encore fonctions disponibles.
* GitHub m’a servi pour la sauvegarde essentiellement, mais en équipe, il permet un versioning facile et une sécurité quant à l’intégrité du code en général (dans le cas où un développeur ferait une bêtise).
* Balsmiq m’a permis de faire les premières maquettes de mon projet. J’ai pu voir de quel élément j’aurais besoin pour remplir les différentes fonctions dont j’aurais besoin avant même de commencer à coder.

### Classes

#### Main

La classe MyHand stocke toutes les informations d’une main donnée (utilisateur, modèle ou à enregistrer). Elle ne s’appelle pas « Hand », car il s’agit déjà d’un objet du Leap Motion.

Avant la création de cette classe, les données de la main étaient envoyées sous forme de liste. Suite à des erreurs entre chaque élément et à la complexité pour envoyer ces informations, la classe MyHand m’a permis d’unifier la communication de position entre deux affichages ou traitements.

#### Paint

Cette classe a pour seul but de dessiner. Les ronds pour chaque doigt et la paume, les traits reliant les doigts à la paume et leur couleur sont « créés » dans cette classe. La fonction « PaintHand » qui assemble les ronds et les traits reçoit en paramètre une main, les positions sont donc normalisées et doivent être converties en positions réelles, c’est-à-dire par rapport à l’affichage (et non par rapport au Leap Motion). C’est ce que fait la méthode « normToPanelPos ». Les dimensions du panel sont obtenus grâce à la méthode « GetPanelSize » qui est appelée sur la fenêtre principale à chaque fois que celle-ci change de taille.

#### Serialization

Cette classe permet de sauvegarder un objet main dans un fichier serialisé. Il stocke le fichier sous forme de XML dans le dossier source du programme, dans un dossier nommé dans cette même classe, dans le cas présent, il s’agit du dossier « serial ».

Cette classe va aussi récupérer tous les fichiers du dossier pour récupérer leur nom et leurs informations pour désérialiser le tout et ainsi, les afficher dans la classe principale.

**[MONTRER COMMENT MARCHE UN FICHIER XML/SERIALISATION]**

## Démonstration

**[MONTRER FINGER’S CLONER]**

## Conclusion

Ce projet m’a permis de prendre confiance en mes compétences de programmation en C#, mais aussi d’organiser un projet par moi-même et ce, de A à Z. J’ai pu travailler de façon indépendante, de chercher et trouver des solutions qui ne m’ont pas forcément été directement enseignées. Le Leap Motion est le seul appareil externe avec lequel j’ai dû coder en C#, alors j’ai aussi dû apprendre à m’en servir, que ce soit en lisant la documentation ou en programmant des petites fonctionnalités.

Pour que ce projet soit vraiment abouti, il faudrait ajouter les informations restantes sur les doigts, c’est-à-dire les autres phalanges. Cela ne devrait pas être trop compliqué à mettre en place, il suffirait juste d’enregistrer ces informations dans la classe MyHand, et de modifier la classe Paint pour dessiner ces phalanges.