2017-2018

Loïc Dubas

I.FA-P3B – CFPT-I

Manuel technique

TPI – Juin 2018

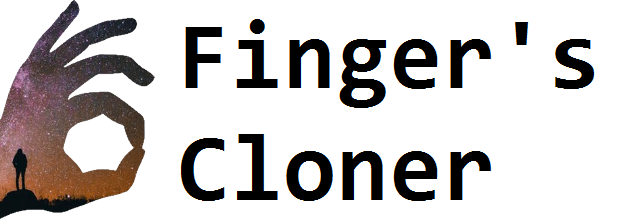


Table des matières

[1. Analyse préliminaire 2](#_Toc516581718)

[1.1 Introduction 2](#_Toc516581719)

[1.1.1 Présentation de l’application 2](#_Toc516581720)

[1.1.2 Fonctionnement 2](#_Toc516581721)

[1.1.3 Pourquoi ce projet ? 2](#_Toc516581722)

[1.2 Organisation 2](#_Toc516581723)

[1.3 Objectifs 2](#_Toc516581724)

[1.4 Planification initiale 3](#_Toc516581725)

[2. Analyse 4](#_Toc516581726)

[2.1 Analyse concurrentielles 4](#_Toc516581727)

[2.2 Cahier des charges détaillé 4](#_Toc516581728)

[2.2.1 Définition de l’audience 4](#_Toc516581729)

[2.2.2 Définition du contenu et de fonctionnalités 4](#_Toc516581730)

[2.2.3 Maquette préliminaire 4](#_Toc516581731)

[3. Conception 6](#_Toc516581732)

[3.1 Analyse de l’environnement 6](#_Toc516581733)

[3.2 Détermination des différentes parties 6](#_Toc516581734)

[3.3 Maquette graphique 6](#_Toc516581735)

[Comment utiliser *Finger’s Cloner* 7](#_Toc516581736)

[Enregistrer une nouvelle position 7](#_Toc516581737)

# Analyse préliminaire

## Introduction

### Présentation de l’application

*Finger’s cloner* est un programme de reconnaissance de position de main destiné à des fins d’apprentissage. Il permet à l’utilisateur de créer des positions de doigts ou de tester ses connaissances en tentant de copier une position donnée.

### Fonctionnement

Grâce au capteur du « *Leap Motion* », cette application affiche la position de votre main en temps réel. Elle affiche aussi une position de main qui correspond au modèle que vous devrez tenter de copier au mieux.

### Pourquoi ce projet ?

J’ai choisis de faire ce programme car je voulais trouver un projet avec une réelle utilité. L’apprentissage du langage des signes, par exemple, peut être bénéfique à tous, et ce projet offre une première version d’une application pouvant s’adapter facilement à cette apprentissage. Dans le domaine de la rééducation aussi, on peut imaginer une évolution du logiciel où le médecin pourrait préenregistrer des positions de main adaptée à son patient.

## Organisation

Dès le 4 juin 2018, ma séance de TPI débutera officiellement. Dès lors, je n’aurais que 10 jours pour tenter de mettre au point mon projet. Avec 8 heures par jour, cela me laisse 80 heures au total pour mener ce travail à bien.

Pour garder une sauvegarde de mon projet de façon sécurisée et simple, j’utilise GitHub. Cela permet, en quelques clics, de faire des copies de mon projet en ligne, ainsi je ne crains pas de panne logicielle ou matérielle. GitHub permet aussi de revenir à d’anciennes versions si besoin est.

Pour démarrer sur une base concrète, les maquettes sont la première chose que j’ai faite. Cela m’a aidé à visualiser ce dont j’avais besoin pour mener à bien mon programme.

## Objectifs

L’objectif principal de mon logiciel est de mettre au point un programme de reconnaissance de position de main. Il doit permettre à l’utilisateur de sauvegarder une position personnalisée avec un nom, une description et une image. La fenêtre principale affiche sa main et le modèle à recopier, avec son nom, sa description et son image, bien entendu. Un code couleur (rouge, orange et vert) indique à l’utilisateur si sa position s’approche de celle du modèle.

D’un point de vue personnel, ce logiciel me permettre de confirmer mes compétences en programmation C#, mais aussi ma compétence à apprendre par moi-même. Ainsi, je vais devoir prendre en main le Leap Motion et ses fonctions dans sa version la plus récente, mais aussi à calculer des positions relatives à la fenêtre en se basant sur les positions relatives au Leap Motion.

## Planification initiale

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Jour 1 | Jour 2 | Jour 3 | Jour 4 | Jour 5 | Jour 6 | Jour 7 | Jour 8 | Jour 9 | Jour 10 |
| Installation du Leap Motion |  |  |  |  |  |  |  |  |  |  |
| Paramétrage du Leap Motion |  |  |  |  |  |  |  |  |  |  |
| Création de l’interface |  |  |  |  |  |  |  |  |  |  |
| Capture de la main + affichage |  |  |  |  |  |  |  |  |  |  |
| Création de la partie « *Appreneur* » |  |  |  |  |  |  |  |  |  |  |
| Création de la partie « *Apprentissage* » |  |  |  |  |  |  |  |  |  |  |
| Finissions |  |  |  |  |  |  |  |  |  |  |
| Documentation |  |  |  |  |  |  |  |  |  |  |
| Manuel technique |  |  |  |  |  |  |  |  |  |  |

Ci-dessus, j’ai tenté de planifier mon projet en essayant de m’imaginer les différentes étapes et leurs durées de réalisation. L’installation et le paramétrage du Leap Motion ne me prenne que peu de temps, car j’ai pris le temps de m’informer sur l’appareil en général avant de commencer ce travail.

La création de l’interface ne consiste qu’à poser les éléments sur mes différentes fenêtres.

La capture de la main et son affichage devrait être assez simple. Le Leap Motion renvoie directement les coordonnées de chacun des doigts, de façon normalisée si nécessaire. Son affichage ne devrait consister qu’à une mise à l’échelle par rapport à mon élément d’affichage.

La partie « appreneur » gère la sauvegarde des positions de la main. Les coordonnées de chaque doigt et de la paume doivent être enregistré. Je ne sais pas vraiment comment sérialiser des objets multiples, alors je me réserve quelques jours pour tenter de réaliser par moi-même l’enregistrement.

La partie « apprentissage » compare la position actuelle de l’utilisateur au modèle sélectionner. Je ne sais pas encore quelle logique je vais appliquer pour ce faire. Je me garde suffisamment de jour pour cela.

Les finissions consisteraient à « nettoyer » tous les bouts de codes qui peuvent être soient supprimés, soient simplifiés, soit unifiés pour s’adapter à différentes situations.

La documentation et le manuel technique doivent évoluer quotidiennement en fonction des avancées du projet.

# Analyse

## Analyse concurrentielles

Plusieurs personnes se sont déjà penchées sur un programme de reconnaissance de signe de main avec le Leap Motion, mais aucune application commercialisable n’existe. Ces personnes ont pour la plupart codé en Python et elles ne proposent que peu d’interface.

De plus en plus d’établissement s’intéresse à mêler détection de mouvements et traduction en temps réel en langage des signes. Pour l’instant, il n’existe que des ébauches de projet.

## Cahier des charges détaillé

### Définition de l’audience

Mon projet s’intéresse en premier lieu aux personnes voulant apprendre le langage des signes et celles en rééducation de la main. Grâce à sa simplicité d’utilisation, l’application peut convenir à tous ceux intéressés à apprendre une nouvelle façon de s’exprimer, et ce, peu importe l’âge.

### Définition du contenu et de fonctionnalités

* Capture de la main
* Affichage de la main
* Enregistrement :
  + De la position de la main
  + D’un nom
  + D’une description
  + D’une illustration
* Affichage de la position enregistrée
* Modification des informations de la position (nom, description, illustration)
* Sérialisation de plusieurs positions
* Récupération des positions enregistrées
* Affichage de toutes les positions dans une liste déroulante

### Maquette préliminaire

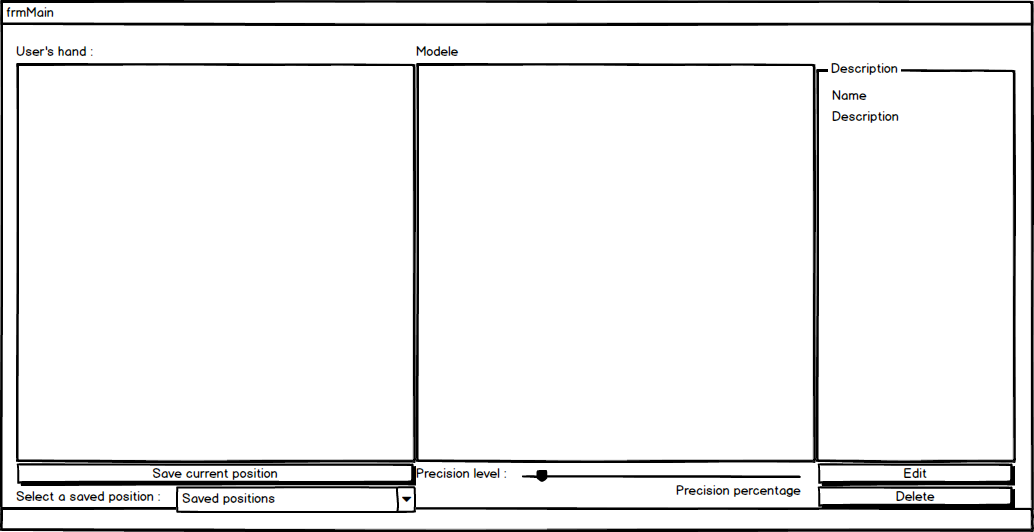


Figure . Fenêtre principale

|  |  |
| --- | --- |
| Figure . Fenêtre Nouveau Modèle | Figure . Fenêtre Commentaire |

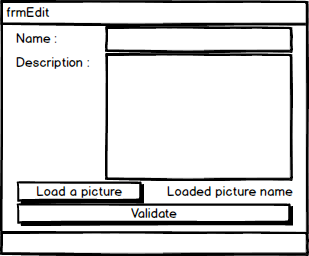


Figure . Modifier une position

# Conception

## Analyse de l’environnement

Pour m’aider dans la conception de mon projet, j’utilise plusieurs softwares :

* **Visual Studio**, en tant qu’environnement de développement C#, qui me permet de faire un programme avec des composants visuels faciles à mettre en place.
* **GitHub**, en tant que gestionnaire de version, qui me permet de garder une trace de l’évolution du logiciel et qui me permet de faire des sauvegardes facilement.
* **LeapMotion SDK** est obligatoire pour pouvoir utiliser le capteur de mouvement *Leap Motion*, mais il offre aussi des samples permettant de visualiser facilement les informations reçues par l’appareil.
* **Word** comme éditeur de texte ce qui me permet d’écrire les manuels nécessaires.
* **Balsamiq** me permet de créer des maquettes de mon programme.

## Détermination des différentes parties

La fenêtre principale (frmMain) contiendra deux panel affichant respectivement la main de l’utilisateur et le modèle à copier. Sur le bord droit de la fenêtre, le nom, la description et l’image de la position seront affichés. En dessous de cela, deux boutons permettront de modifier et de supprimer la position en cours. En bas à gauche, une liste déroulante proposera toutes les positions préenregistrées. Un bouton en bas de cette liste permettra d’enregistrer la position actuelle de l’utilisateur. En bas à droite, un slider permettra de choisir le niveau de précision requis pour que la position de l’utilisateur soit considérée comme correcte par rapport au modèle.

Une seconde fenêtre (frmNewModel) s’ouvrira lorsque l’utilisateur cliquera sur le bouton « Enregistrer la position ». L’utilisateur pourra choisir un nom et une image pour la position. En cliquant sur « Enregistrer », une autre fenêtre (frmComment) s’ouvrira pour que l’utilisateur propose une description à sa position.

Une autre dernière fenêtre (frmEdit) est la fenêtre qui s’ouvrira quand l’utilisateur voudra modifier la position en cours.

## Maquette graphique

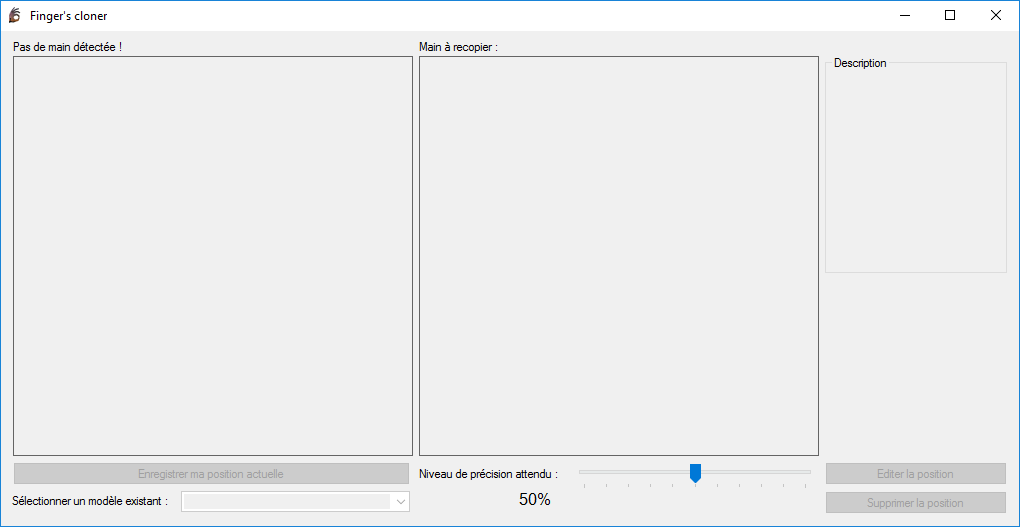


Figure . Fenêtre principale

|  |  |
| --- | --- |
| Figure . Fenêtre de création de modèle | Figure . Fenêtre d'édition de position |

## Conception du code

Mon code se sépare en plusieurs classes :

La première est celle qui permet de contrôler le Leap Motion. Nommée ***LeapController.cs***, elle récupère les informations de la main que le capteur visualise. Ces informations sont stockées grâce à la classe *MyHand.cs*.

***MyHand.cs***est la classe qui me permet de stocker les informations d’une main, que ce soit celle de l’utilisateur, celle qui s’affiche lorsque l’utilisateur veut enregistrer sa position ou encore celle qui s’affiche en tant que modèle. Son premier constructeur ne demande que la position des doigts et de la paume. Ce constructeur concerne directement la main de l’utilisateur. Un second constructeur demande les mêmes informations et un nom ainsi qu’une description en plus. Cela concerne la position à enregistrer et le modèle. Un troisième constructeur demande une illustration en plus. Ainsi, si l’utilisateur le souhaite, il peut proposer une photo à afficher avec une position donnée.

**Serialization.cs** ne sert qu’à sérialiser et déserialiser le fichier XML permettant la sauvegarde des positions. Il reçoit une main en paramètre, une fois celle-ci reçu, il va déserialiser le fichier XML et ajouter toutes les mains stockées dans une liste. Il stocke la main reçue en paramètre dans la liste, puis resérialise le tout.

**Paint.cs** gère le dessin. À l’initialisation, il requiert la largeur et la hauteur du panel dans lequel il va devoir dessiner. Trois fonctions dessinent respectivement un rond, un trait et la dernière fonction place les deux ensembles pour former un doigt. Les trois mêmes fonctions existent en version colorée. Ainsi, rien ne change dans leur fonctionnement mais elles reçoivent une couleur en paramètre.

Une dernière fonction transforme les données de positions des doigts, originellement en normalisé, en position dite « réelle », c’est-à-dire la position exacte des doigts sur le panel dépendamment de l’emplacement de la paume.

La fenêtre principale reçoit la main du Leap Motion et celle enregistrée. Elle envoie les données à la classe Paint.cs qui permet de dessiner la main sur le panel souhaité. Le même principe se passe avec le modèle. Le panel de l’utilisateur est rafraîchit 10 fois par seconde pour assurer un maximum de fluidité sans trop surcharger l’ordinateur.

La fenêtre de création de nouveau modèle créé une instance de la classe MyHand.cs à laquelle il envoie la position des doigts, de la paume et le nom, la description et l’image.

# Réalisation

## Dossier de réalisation

## Description des tests

## Erreurs restantes

# Conclusion

# Annexes

## Sources

## Procédure d’installation

## Présentation de la fin du projet