Université de Caen
UFR des Sciences
Département Informatique
Master 1 2013/2014

LeapShortcut

M. Youssef Chahir

Rapport de projet

Louis Aubert

Rendu le 09 Mai 2014

Table des matières

Remerciements **2**

Introduction **3**

1 Leap Motion **4**
1.1 Présentation 4
1.2 Détails matériel 4
1.3 Détails logiciel 5
1.3.1 Les Mains 5
1.3.2 Les Doigts 5
1.3.3 Les outils 5
1.3.4 Les gestes 6
1.4 Utilité 7

2 LeapShortcut **9**
2.1 Présentation 9
2.1.1 Solutions actuelles 9
2.1.2 Choix du langage 9
2.1.3 Diagramme prévisionnel 10
2.2 Conception 11
2.2.1 Caractéristiques d'un raccourci 11
2.2.2 Stockage des raccourcis 11
2.2.3 Interface graphique utilisateur 12
2.2.4 Débogage 12
2.2.5 Diagramme de classe 13
2.3 Développement 14
2.3.1 La classe LeapListener 14
2.3.2 Enregistrement d'un raccourci en XML 15
2.3.3 Analyse d'un mouvement et exécution du raccourci associé 15

Conclusion **17**

Références **18**

Annexes **18**
Annexe 1 : LeapListener 18
Annexe 2 : Enregistrer un raccourci en XML 22
Annexe 3 : Analyse d'un mouvement 24
Annexe 4 : Exécution d'un fichier script 25
Annexe 5 : Manuel Utilisateur 26

Remerciements

Je tiens sincèrement à remercier Youssef Chahir, mon tuteur de projet, pour avoir accepter spontanément de m'encadrer lors de ma demande d'effectuer un projet en rapport avec le Leap Motion. Il a été une aide précieuse et nos moments de réflexion m'ont permis d'évoluer de façon positive dans ce projet.

Introduction

Le projet qui m'a été proposé et que je vous présente dans ce rapport porte sur le Leap Motion, un périphérique de reconnaissance gestuelle. Dans un premier temps, j'ai dû effectuer des recherches sur la technologie utilisée et effectuer des tests sur ses performances. Après une bonne connaissance de l'outil, j'ai conçu et développé une application permettant d'associer un groupe de gestes à une liste de fichiers à exécuter.

Le Leap Motion permet de reconnaître nativement, et avec une rapidité quasi instantanée, des gestes simples tel que des mouvements rapides verticaux et horizontaux. Le périphérique reconnaît quand la main de l'utilisateur effectue un cercle. Pour finir, il reconnaît la simulation de touche du clavier et la simulation de touche d'un écran tactile. Avec ses 4 types de gestes simples et naturellement reconnus par le périphérique, on peut créer des centaines de combinaisons et y assigner des actions spécifiques.

L'application développée, nommée Leap Shortcut, a pour but de faciliter la vie de l'utilisateur. Une première partie permet, grâce à une interface graphique, de créer des raccourcis en assignant des groupes de gestes simples appelés mouvements à une liste de fichiers qui peuvent être des fichiers exécutables, des pages web, des scripts. La seconde partie de l'application est un service, activé en arrière plan, totalement invisible, qui va écouter en permanence les gestes effectuées sur le Leap Motion. Si un mouvement exécuté par l'utilisateur correspond à un raccourci enregistré, la liste des fichiers assignés à ce raccourci sera exécutée.

Ce rapport présente dans un premier temps le Leap Motion, ses caractéristiques matérielles, logicielles et ses utilités. Dans un second temps, le rapport présente l'analyse conceptuelle du Leap Shortcut et son développement.

1 Leap Motion

1.1 Présentation

Le Leap Motion est un nouveau périphérique de reconnaissance gestuelle apparu en 2012 après une campagne de collecte de fonds de plus de 40 millions de dollars. Ce périphérique permet de reconnaître les doigts avec une précision en dessous du millimètre. Grâce à la combinaison de ses deux capteurs optiques et de ses trois capteurs infrarouges, le Leap Motion est en mesure de capter entre 30 et 300 images par seconde en fonction de la puissance de l'ordinateur utilisé. Le Leap Motion se démarque des autres périphériques de reconnaissance gestuelle par le fait qu'il soit "transparent", c'est à dire qu'il n'y ait aucun support à fixer sur l'utilisateur comme c'est le cas pour les diverses technologies concurrentes comme les exosquelettes, les gants de données à fibres optiques, les capteurs à accéléromètres ou encore les trackers magnétiques. Ce sont des technologies de pointe qui, combinées ensemble, peuvent produire des résultats extrêmement performants mais à des prix inaccessibles aux consommateurs. C'est avec l'avènement du Kinect de Microsoft que les capteurs dit "low-cost" sont apparus en masse sur le marché. Le Kinect, contrairement au Leap Motion, permet de reconnaître la gestuel du corps entier grâce à l'utilisation d'une caméra 3D par stéréoscopie et triangulation active. Mais son point faible est son faible taux de rafraîchissement qui est bloqué à 30 images par seconde.

Le Leap Motion se veut donc transparent pour l'utilisateur et plus précis que ses concurrents directs.

1.2 Détails matériel

D'après les ingénieurs du Leap Motion, aucune technologie avant-gardiste ou révolutionnaire n'est utilisée dans le périphérique. C'est la combinaison de deux caméras monochromatiques et de trois LEDs infrarouges qui permettent au Leap Motion de capter les mouvements dans une zone quasi hémisphérique de 1 mètre de diamètre. Les LEDs génèrent des modèles 3D de points alors que les deux caméras génèrent au maximum 300 images par seconde de données réfléchies. Ces données sont ensuite transmises par USB à l'ordinateur qui va les analyser grâce à des algorithmes mathématiques très puissants qui n'ont pas été dévoilés par les ingénieurs. En raison des restrictions des brevets et des secrets de recherche, on ne connaît pas vraiment la nature précise des algorithmes utilisés. Mais étant donné que le Leap Motion combine deux caméras monochromatiques et trois LEDs infrarouges, le contrôleur peut être classé comme un système de suivi optique basé sur la vision stéréoscopique.

FIGURE 1 – Deux caméras monochromatiques et 3 LEDs infrarouges

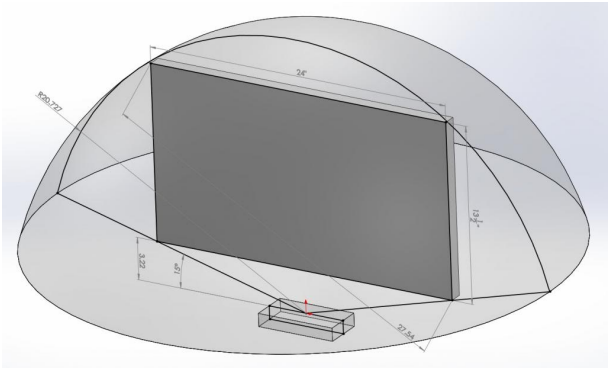

FIGURE 2 – Zone de détection du Leap Motion

1.3 Détails logiciel

Un SDK (Software Development Kit) est fourni avec le Leap Motion permettant aux développeurs de récupérer la position d'une main, d'un doigt ou d'un objet mais aussi certains gestes simples nativement reconnus par le contrôleur. L'un des atouts du SDK est qu'il est disponible dans un nombre important de langages. La librairie est développée initialement en C++. Mais grâce à SWIG, un outils open-source, la librairie est aussi disponible en java, C#, javascript, python, Objective-C...

Le Leap Motion "écoute" ce qui se passe dans son champ de vision, il reconnaît les mains, les doigts, les outils et fournit une trame contenant les informations qu'il met à jour à chaque changement d'états d'un de ces objets. L'API fournit certaines mesures de grandeurs physiques comme :

— La distance en millimètres
— La durée en microsecondes
— La vitesse en millimètres/seconde
— L'angle en radians

1.3.1 Les Mains

L'objet Hands fournit des informations sur la position, les caractéristiques et le mouvement de la main détectée. Une main reconnue dans le champ de vision se voit assigner un identifiant qui permet d'utiliser l'objet au fil du temps. Une liste de doigts et d'outils sont aussi assignés à l'objet.

1.3.2 Les Doigts

Le contrôleur détecte et suit chaque doigt (objet Finger) qu'il reconnaît dans son champ de vision. Il est capable de différencier chaque doigt en calculant leurs positions dans l'espace et leurs tailles par rapport aux autres. Tout comme une main, un doigt se voit assigné un identifiant unique.

Les doigts ont différentes caractéristiques physiques :

— La direction dans laquelle le doigt pointe
— La taille du doigt en millimètres

1.3.3 Les outils

Un outils (objet Tool) est en fait une dérivée d'un doigt mais c'est un objet droit, plus mince et plus long que l'auriculaire. Un outils a les mêmes caractéristiques qu'un doigt.

1.3.4 Les gestes

Le contrôleur détecte certains schémas de mouvements comme des gestes (objet Gesture) effectués par l'utilisateur.

FIGURE 3 – Swipe - Un long mouvement linéaire d'un doigt

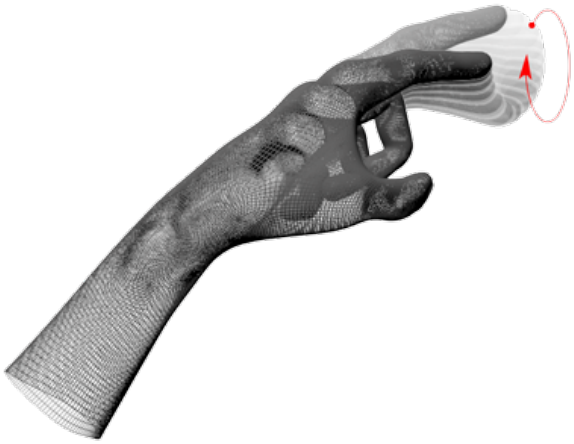

FIGURE 4 – Circle - Le doigt traçant un cercle

FIGURE 5 – Key Tap - Un mouvement du doigt tapant une touche du clavier

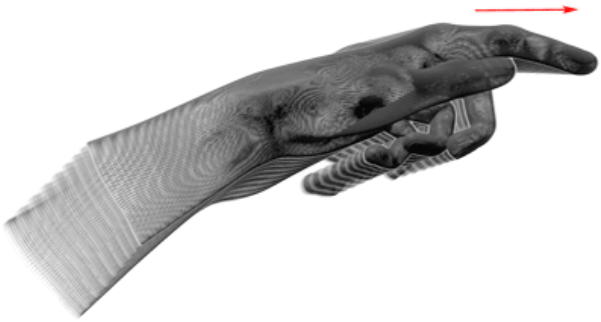

FIGURE 6 – Screen Tap - Un mouvement du doigt tapant un écran tactile

Le développeur peut configurer le contrôleur pour savoir si un schéma de mouvements doit être reconnu comme étant un geste en modifiant les caractéristiques physiques des différents gestes.

Un geste Circle peut être configurer en modifiant le rayon minimum que le mouvement doit avoir pour être détecté en tant que geste Circle.

Un geste Swipe peut être configurer en modifiant la vitesse, la vélocité et la longueur minimum d'un mouvement pour être reconnu en tant que geste Swipe.

Un geste Key Tape ou Screen Tap peut être configuré en modifiant la vélocité, respectivement la vélocité vers le bas, et la vélocité vers l'avant.

Tous ces gestes ont quatre états différents :

— START : le geste est détecté pour la première fois, un identifiant lui est assigné.
— UPDATE : le geste est en cours de réalisation.
— STOP : le geste est terminé.
— INVALID : le geste n'a pas rempli toutes les conditions pour être un geste valide.

On peut ainsi suivre au fil du temps, à travers les trames transmises à l'ordinateur, les gestes avec leurs identifiants et leurs états à un moment précis.

1.4 Utilité

Le Leap Motion a diverses utilisations aujourd'hui. Il est principalement utilisé dans les jeux, dans l'éducation et à des fins utilitaires.

Dans les jeux, il remplace les contrôleurs habituels et permet une nouvelle immersion dans les jeux en utilisant nos mouvements naturels.

FIGURE 7 – Immersion dans le jeu

Dans l'éducation, il permet au jeune d'apprendre d'une façon plus ludique, en utilisant le corps, il permet de développer un sens de l'habileté et du réflexe.

FIGURE 8 – Jeu d'habileté et de réflexe

Mais le Leap Motion se veut dans un premier temps, un outil utilitaire, permettant d'interragir avec l'ordinateur de façon naturelle, simple et précise. Le Leap Motion a été présenté comme un périphérique futuriste à l'image du film Minority Report, où Tom Cruise contrôle entièrement son ordinateur grâce à ses mains. Aucune application n'a réussi à approcher ce concept mais certaines applications permettent de remplacer des touches du clavier par un geste effectué sur le Leap Motion.

FIGURE 9 – Extrait du film Minority Report (2002)

Le Leap Motion peut avoir une certaine utilité dans le secteur de la médecine. Il peut être un outil très intéressant pour les personnes à mobilité réduite, un individu qui ne peut utiliser ses mains et plus particulièrement ses doigts avec précision pourrait trouver dans l'utilisation du Leap Motion une aide précieuse pour utiliser son ordinateur. Il pourrait aussi servir à la rééducation, en proposant des applications basées sur des exercices avec l'objectif de faire retrouver au patient une dextérité gestuelle perdue. Certains chirurgiens utilisent déjà le Leap Motion pour les assister dans le bloc opératoire. Il permet de contrôler les monitors sans toucher un clavier ou une souris, et permet ainsi de réduire les risques de contamination.

FIGURE 10 – Utilisation du Leap Motion par un chirurgien dentiste

Le Leap Motion a donc de multiple utilités, il peut être utilisé pour l'éducation, le loisir, la santé, la robotique, l'art, la musique mais aussi à des fins utilitaires. C'est cette dernière utilisation qui nous intéresse pour le projet présenté dans ce rapport, le LeapShortcut.

2 LeapShortcut

2.1 Présentation

Le projet LeapShortcut est un programme utilitaire visant à aider l'utilisateur en proposant des raccourcis pour l'exécution de fichiers et programmes grâce au Leap Motion. Lors de certaines tâches, l'utilisateur a besoin de démarrer plusieurs programmes : Avec le LeapShortcut, il peut aisément configurer un raccourci et y assigner une liste de programmes à lancer. Un raccourci est un ensemble de gestes simples, nativement reconnus par le Leap Motion, appelé mouvement. LeapShortcut est un service, il peut être exécuté de façon silencieuse et transparente sur l'ordinateur. Quand l'utilisateur exécute un mouvement dans le champ de vision du Leap Motion, le service va capter ce mouvement et chercher dans la base de données si le mouvement existe. Si c'est le cas, il va lancer les programmes et fichiers associés. Une interface utilisateur permet de créer et de gérer les raccourcis.

2.1.1 Solutions actuelles

Des logiciels comme HandWave ou BetterTouchTool ont été développés pour proposer d'assigner des raccourcis clavier avec des gestes simples du Leap Motion mais aucun de ces logiciels ne permet d'assigner une combinaison de gestes, ce qui limite très rapidement le nombre de raccourcis possibles.

2.1.2 Choix du langage

Lors de la phase de conception, il avait été annoncé que le programme serait écrit en C++, étant donné que c'est le langage qui est utilisé pour la librairie. Mais après une prise en mains plus profonde du périphérique et du SDK, le choix s'est naturellement tourné vers Java pour plusieurs raisons :

— Meilleures connaissances et expériences dans ce langage.
— Java est multi plate-formes, le logiciel tourne donc sur toutes les plates-formes disponibles.
— Aucune perte de performances par rapport à la librairie C++.

Le dernier point est important. La librairie est écrit en C++ mais disponible dans plusieurs langages grâce à SWIG un parseur open-source qui permet de transformer des librairies C++ en librairies appartenant à des langages différents. Dans un premier temps, on pourrait penser que ces librairies générées grâce à SWIG souffriraient d'un manque d'optimalité et donc connaîtraient une baisse de performances, ce qui serait rebutant étant donné que la force du Leap Motion est sa performance. Mais après une série de tests avec les scripts de base fournis pour les développeurs avec le SDK, aucune perte de performances notable n'a été rencontrée. Selon les développeurs, C++ est recommandé pour développer un jeu vidéo, mais pour développer un logiciel utilitaire, aucune préférence n'est émise. On peut donc utiliser Java sans risquer de compromettre les performances de détection du Leap Motion.

2.1.3 Diagramme prévisionnel

Lors de la phase de conception, un diagramme de Gantt a été généré pour planifier l'organisation du projet et avoir une visibilité sur l'avancement de celui-ci.

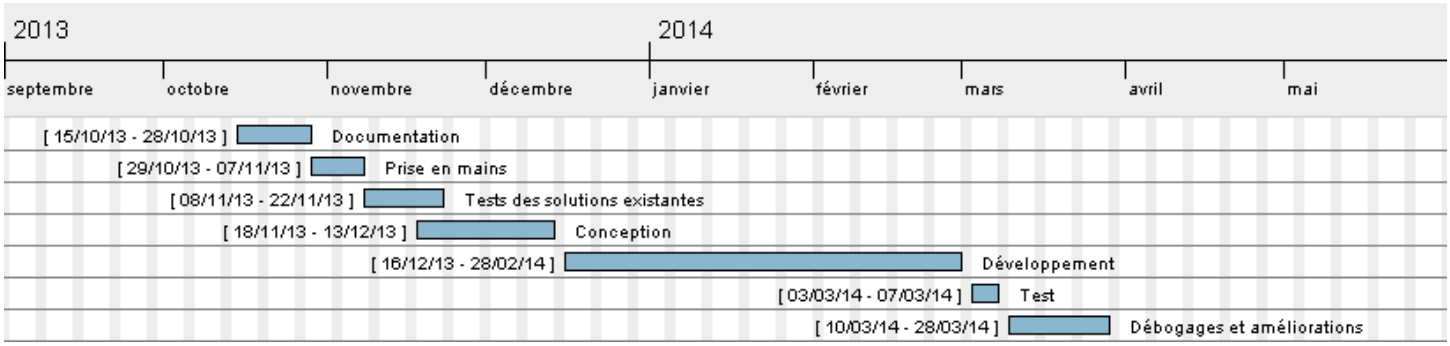

FIGURE 11 – Diagramme de Gantt initial

Les prévisions ont été optimistes et certaines tâches ont pris plus de temps que prévu. Voici le diagramme de Gantt final qui reflète l'organisation réelle du projet LeapShortcut.

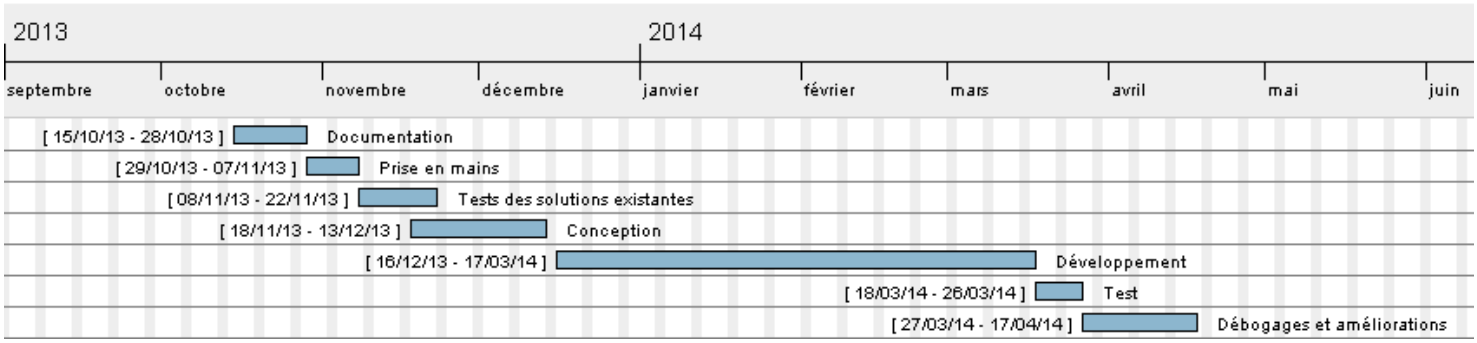

FIGURE 12 – Diagramme de Gantt final

2.2 Conception

Le LeapShortcut est une application qui doit avoir deux comportements différents, le premier comportement est la configuration d'un raccourci, l'application est munie d'une interface graphique et les actions sont déclenchées par l'utilisateur. Le deuxième comportement est un service qui s'exécute de façon transparente et qui permet par le biais du Leap Motion de reconnaître des mouvements et d'exécuter les raccourcis. Ces deux comportements ne sont pas totalement isolés l'un de l'autre, le même code est utilisé mais il existe deux contrôleurs différents. Les contrôleurs servent à traiter les données retournées par le Leap Motion suivant la nature de l'action définie dans le contrôleur. Ainsi, le contrôleur chargé de la configuration, appelé LeapShortcut, a des méthodes spécifiques à la configuration de raccourcis comme une méthode permettant d'enregistrer un raccourci dans la base de données. Alors que le contrôleur chargé de la détection silencieuse, appelé LeapShortcutService, contient des méthodes spécifiques qui vont par exemple démarrer le service ou encore analyser un schéma de gestes détectés par le Leap Motion, correspond à un mouvement enregistré dans la base de données.

2.2.1 Caractéristiques d'un raccourci

Un raccourci a comme caractéristiques :

- Nom du raccourci : le nom est unique.
- Description du raccourci : une courte description résumant l'utilité du raccourci.
- Mouvement : le schéma de gestes effectués par l'utilisateur pour activer le raccourci.
- Liste des fichiers : la liste des fichiers à exécuter, ça peut être des fichiers exécutables, des pages web, des scripts...
- Exécution séquentielle : si cette option est cochée, alors l'ensemble des fichiers ne sera pas exécuté dès la détection du mouvement mais de façon séquentielle. A chaque fois que l'utilisateur exécutera un et seulement un Tap Gesture, le fichier suivant de la liste sera exécuté.

2.2.2 Stockage des raccourcis

Pour stocker les raccourcis et les sauvegarder, on utilise un fichier XML. Ce type de fichier est la meilleure solution car il a de nombreux avantages pour des petites données :

- Facile à utiliser en Java.
- Structuré et donc facilement lisible.
- Rapide à exploiter quand les données ne sont pas excessives.

On stockera rarement plus de 50 raccourcis, le fichier sera donc petit et la machine virtuelle de Java y accédera très rapidement.

Structure du fichier XML

```
1  <shortcutDB>
2      <shortcut>
3          <name>Nom du raccourci</name>
4          <description>
5              Description du raccourci
6          </description>
7          <fileList sequentialOpening="false">
8              <file type="URI">fichier1</file>
9              <file type="URL">fichier2</file>
10         </fileList>
11         <gestureList>
12             <gesture>Geste1</gesture>
13             <gesture>Geste2</gesture>
14             <gesture>Geste3</gesture>
15         </gestureList>
16     </shortcut>
17 </shortcutDB>
```

2.2.3 Interface graphique utilisateur

L'interface graphique doit être munie de fonctionnalités permettant à l'utilisateur de configurer ses raccourcis. Un panneau principal permet de lister les raccourcis présents sur la machine et de les gérer, c'est à dire de modifier ou supprimer un raccourci de la liste. Quand l'utilisateur sélectionne un raccourci dans la liste, un résumé de celui-ci est affiché. Grâce à des boutons, l'utilisateur peut démarrer ou stopper le service. Un dernier bouton sert à créer un nouveau raccourci, un nouveau panneau apparaît permettant à l'utilisateur de renseigner les informations sur le raccourci.

2.2.4 Débogage

Une option de débogage, appelé "Debug" permet, lors de la détection, d'avoir tout les renseignements sur ce que détecte le Leap Motion quand l'utilisateur exécute un mouvement. On peut ainsi analyser les erreurs et corriger notre mouvement pour que le résultat de la détection soit correct par rapport au mouvement que l'on veut produire.

2.2.5 Diagramme de classe

Après avoir lister toutes les fonctionnalités de l'application et la structure de données, on peut passer à la phase technique : la conception de l'architecture logicielle.

Voici le diagramme de classe qui ressort de la phase de conception.

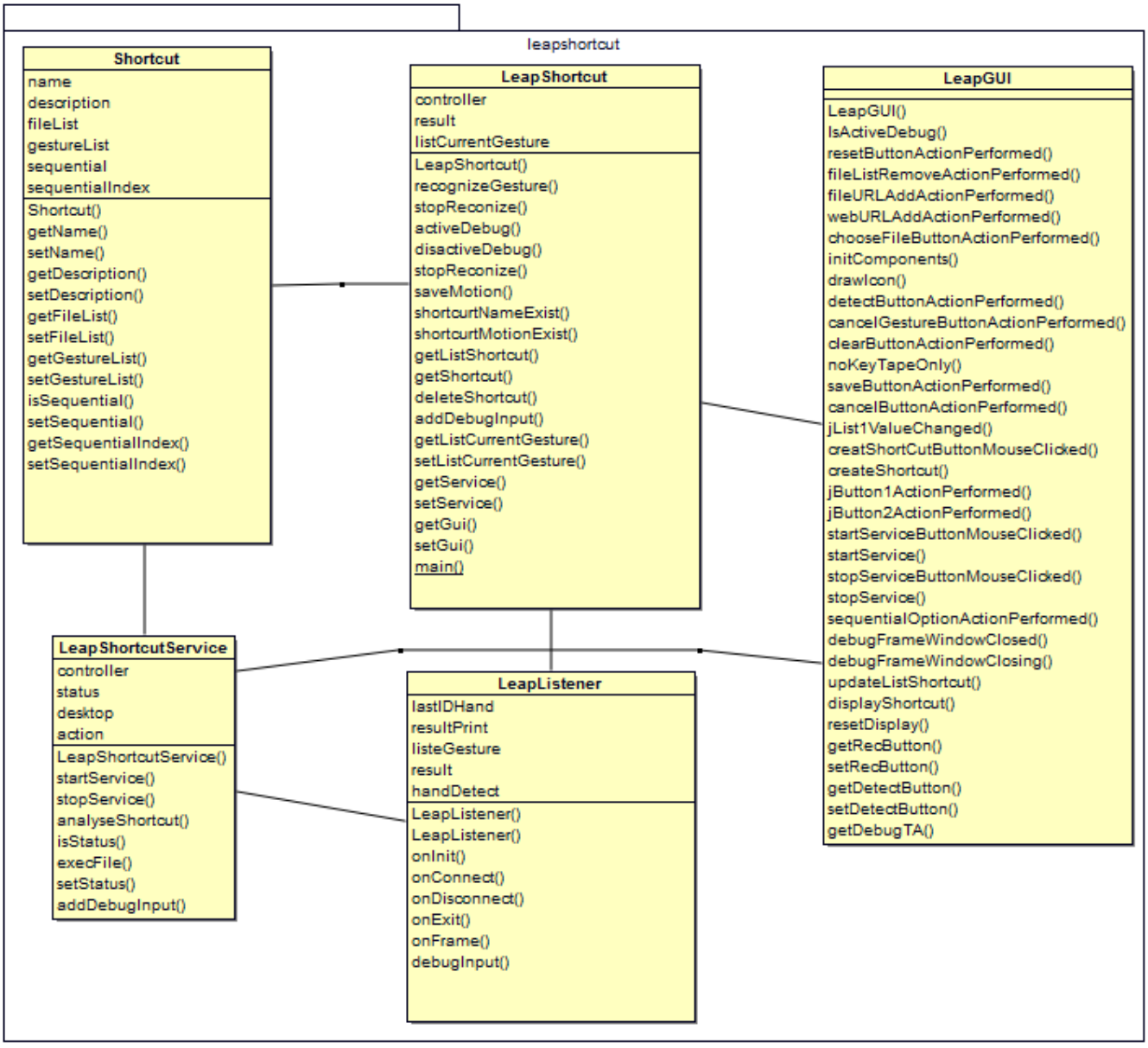

FIGURE 13 – Diagramme de classe

2.3 Développement

Dans cette partie, on expliquera quelques points essentiels du développement du logiciel Leap-Shortcut :

— Détection de mouvements avec la classe LeapListenner.
— Enregistrement d'un raccourci en XML.
— Analyse d'un mouvement et exécution du raccourci associé.

2.3.1 La classe LeapListener

La classe LeapListener fait le lien avec la librairie du Leap Motion. Elle hérite de la classe Listener et permet de connaître les différents états du périphérique :

— onInit() : Appelé une seule fois lors de l'initialisation de l'application.
— onConnect() : Appelé quand le périphérique est branché à l'ordinateur.
— onDisconnect() : Appelé quand le périphérique est débranché.
— onFrame() : Appelé à chaque fois qu'une donnée est captée par le périphérique.

Seul l'état onFrame nous intéresse ici, il permet de récupérer les données analysées par le Leap Motion en temps réel. C'est grâce à cette méthode qu'on récupère un mouvement pour ensuite l'analyser. Les gestes font parti d'une même liste tant que la main n'est pas retirée du champs de vision du Leap Motion. Quand la main n'est plus détectée, alors on analyse la liste des gestes pour constituer le mouvement. Étant donné que la méthode est appelée à chaque mise à jour des données captée par le périphérique, aucune boucle n'est nécessaire pour récupérer la liste des gestes, on déclenche seulement l'action d'analyse quand la main n'est plus dans le champ de vision. Un geste Swipe est un long mouvement linéaire du doigt, que le geste soit un mouvement vers la droite ou vers le haut, le Leap Motion reconnaîtra le même type de geste, un Swipe_Gesture. Mais grâce aux données du geste, plus particulièrement le paramètre "direction", on peut facilement déduire dans quelle direction le geste est effectué. Ansi, on a quatre mouvements pour un Swipe : haut, bas, droite et gauche.

Voici le déroulement de l'algorithme permettant de récupérer un mouvement :

```
1  Si il y a une main dans le champ de vision du Leap Motion
2  Alors
3      On récupère l'identifiant de la main
4
5      Si un geste est reconnu par le Leap Motion
6      Alors
7          on l'ajoute à la liste des gestes en cours.
8      Fin si
9
10 Sinon
11     Si l'identifiant de la main est supérieur à -1 //une main a été repérée
       auparavant
12     Alors
13         Pour chaque geste de la liste des gestes en cours
14         Faire
15             récupérer le type de geste et l'ajouter à une liste de chaînes de caractè
       res
16         Fin pour
17
18         Si le type de reconnaissance est "createShortcut"
19         Alors
20             envoyer le résultat à LeapShortcut
21         Sinon
22             envoyer le résultat à LeapShortcutService
23         Fin si
24     Fin si
25 Fin si
```

La classe LeapListener et l'algorithme sont consultable en annexe 1.

2.3.2 Enregistrement d'un raccourci en XML

Pour enregistrer un raccourci dans la base de données, qui est un fichier XML, on utilise la librairie JDOM. L'enregistrement se fait une fois le formulaire de création de raccourcis rempli. Il suffit donc de créer un élément XML pour chaque information renseignée dans le formulaire et de l'ajouter à la base de donnée XML. L'élément principal est appelé "shortcut". On ajoute à cet élément :

- un élément "name"
- un élément "description"
- un élément "fileList" : la liste des fichiers à exécuter
- un élément "gestureList" : la liste des gestes qui composent le mouvement

Pour chaque élément "file" de "fileList", on doit préciser la nature du fichier. Si c'est une page web, on a une URL web appelée URI, mais si c'est un fichier présent sur l'ordinateur de l'utilisateur, on a une URL basique. On ajoute donc à chaque élément "file" un attribut "type" avec la valeur "url" pour un fichier ou "uri" pour une page web. L'élément "fileList" a aussi un attribut appelé sequentialOpening avec la valeur "true" ou "false" et qui permet de savoir si il faut, soit ouvrir tous les fichiers lors de l'exécution du raccourci, soit les ouvrir de façon séquentielle.

Une fois tous les éléments créés ajoutés à l'élément principal "shortcut", on peut ajouter le raccourci dans le fichier database.xml. Si il n'existe pas, il est créé.

Le code de cette méthode permettant d'enregistrer un raccourci est consultable en annexe 2.

2.3.3 Analyse d'un mouvement et exécution du raccourci associé

Si le LeapShortcut est en mode service, il attend que l'utilisateur exécute un schéma de gestes dans son champ de vision. Une fois le schéma de gestes effectué, le service va l'analyser pour savoir si il correspond à un mouvement connu et ainsi exécuter les fichiers associés. Cette procédure est faite par la méthode "analyseShortcut" de la classe LeapShortcutService.

Si un raccourci séquentiel est en cours, il est sauvegardé en tant que variable de classe, dans la variable sequentialShortcut.

Voici le déroulement de l'algorithme :

```
1  Si le mouvement comporte seulement un geste Tap Gesture
2  Alors
3      f = le prochain fichier à exécuter
4      Si f est une url web
5      Alors
6          demander à l'objet java Desktop  d ouvrir  dans le navigateur l'url de f
7      Sinon
8          appeler la méthode executFile(f)
9      Fin si
10 Sinon
11     Si le mouvement est connu dans la base de données
12     Alors
13         Pour chaque fichier du raccourci
14         Faire
15             f = le prochain fichier à exécuter
16             Si f est une url web
17             Alors
18                 demander à l'objet Desktop d'ouvrir dans le navigateur l'url de f
19             Sinon
20                 appeler la méthode executFile(f)
21             Fin si
22         Fin pour
23     Fin Si
24 Fin Si
```

Le code de cette méthode est consultable en annexe 3

L'objet Desktop ne peut ouvrir les fichiers script, il faut impérativement un terminal. On utilise alors ProcessBuilder pour exécuter un terminal dans un nouveau processus.

la méthode executFile(f) :

```
1  Si l'extension de f est ".bat" ou ".cmd"
2  Alors
3      on demande à l'objet java ProcessBuilder d'ouvrir un terminal et d'exécuter le
       script f
4  Sinon
5      on demande à l'objet Desktop d'ouvrir le fichier avec son application par défaut
6  Fin pour
```

le code de cette méthode est consultable en annexe 4.

Conclusion

Le Leap Motion est un périphérique de reconnaissance gestuelle précis et performant permettant de capter les mouvements des mains et de leurs doigts avec une précision inférieure au millimètre. Grâce à une librairie facile à prendre en mains, les développeurs ont la possibilité d'implémenter des applications dans des domaines variés, jeux vidéos, musique, éducation, santé, utilitaire.

Le LeapShortcut est une application visant à aider l'utilisateur en lui permettant d'enregistrer des raccourcis en associant un schéma de gestes reconnus par le Leap Motion, appelé mouvement, avec une liste de fichiers. Quand l'utilisateur effectuera le mouvement, les fichiers associés seront lancés automatiquement. La principale puissance du LeapShortcut est qu'il s'exécute comme un service, de manière transparente pour l'utilisateur. En permettant, d'un simple mouvement, d'ouvrir une liste de fichiers, le logiciel va permettre de gagner un temps précieux dans le travail de l'utilisateur.

Le Leap Motion a un avenir prometteur, précis, facile d'accès et peu cher (70$), certains grands constructeurs comme HP ou Asus ont déjà misé sur lui en achetant les droits d'exploitation. Un ordinateur portable HP, le envy 17, en est équipé, facilitant certaines tâches comme la gestion multimédia. Combiné à d'autres technologies comme l'Oculus Rift, un casque de réalité virtuelle extrêmement performant, les possibilités seraient immenses : visites touristiques en immersion, aide aux personnes handicapés à améliorer leur quotidien, opérations chirurgicales à distance... L'évolution de l'interaction homme/machine est en marche, il ne reste plus qu'à faire un geste.

Annexes

Annexe 1 : LeapListener

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6
7  package leapshortcut;
8  /******************************************************************************\
9  * Copyright (C) 2012-2013 Leap Motion, Inc. All rights reserved.               *
10 * Leap Motion proprietary and confidential. Not for distribution.              *
11 * Use subject to the terms of the Leap Motion SDK Agreement available at       *
12 * https://developer.leapmotion.com/sdk_agreement, or another agreement         *
13 * between Leap Motion and you, your company or other organization.             *
14 \******************************************************************************/
15
16 import java.io.IOException;
17 import java.lang.Math;
18 import java.util.Stack;
19 import com.leapmotion.leap.*;
20 import com.leapmotion.leap.Gesture.State;
21 import java.net.URISyntaxException;
22 import java.util.ArrayList;
23 import java.util.logging.Level;
24 import java.util.logging.Logger;
25
26 class LeapListener extends Listener {
27
28     int lastIDHand = -1;
29     boolean resultPrint = false;
30     Stack<Gesture> listeGesture = new Stack<Gesture>();
31     LeapShortcut ls;
32     LeapShortcutService lss;
33     ArrayList<String> result = new ArrayList<String>();
34     boolean handDetect = false;
35
36     public LeapListener(LeapShortcut ls){
37         this.ls = ls;
38     }
39
40     public LeapListener(LeapShortcutService lss){
41         this.lss = lss;
42     }
43
44
45     public void onInit(Controller controller) {
46
47     }
48
49     public void onConnect(Controller controller) {
50
51         controller.enableGesture(Gesture.Type.TYPE_SWIPE);
52         controller.enableGesture(Gesture.Type.TYPE_CIRCLE);
53         controller.enableGesture(Gesture.Type.TYPE_SCREEN_TAP);
54         controller.enableGesture(Gesture.Type.TYPE_KEY_TAP);
55     }
56
57     public void onDisconnect(Controller controller) {
58         //Note: not dispatched when running in a debugger.
59
60     }
61
62     public void onExit(Controller controller) {
63
64     }
65
66     @Override
67     public void onFrame(Controller controller) {
68         // Get the most recent frame and report some basic information
69
70
71
72         Frame frame = controller.frame();
73         GestureList gestures = frame.gestures();
74         HandList hands = frame.hands();
```

```
75
76          Hand hand = hands.get(0);
77      Gesture gestureS = gestures.get(0);
78
79          if(!hands.isEmpty())
80      {
81              if(controller.config().getString("debugMode").equals("true") && !
        handDetect)
82                  debugInput(controller, "#####Hand on Frame#####");
83
84              handDetect = true;
85              lastIDHand = hand.id();
86              if(!gestures.isEmpty())
87              {
88
89                  switch(gestureS.type())
90                  {
91                      case TYPE_SWIPE:
92                      {
93                          SwipeGesture swipeS = new SwipeGesture(gestureS);
94
95                          if(controller.config().getString("debugMode").equals("true"))
96                              debugInput(controller, "Swipe id: " + swipeS.id()
97                                      + ", " + swipeS.state()
98                                      + ", position: " + swipeS.position()
99                                      + ", direction: " + swipeS.direction()
100                                     + ", speed: " + swipeS.speed());
101
102
103                         if (swipeS.state() == State.STATE_STOP)
104                         {
105                             listeGesture.push(gestureS);
106                         }
107                         break;
108                     }
109                     case TYPE_CIRCLE:
110                     {
111                         CircleGesture circle = new CircleGesture(gestureS);
112                         String clockwiseness;
113                         if (circle.pointable().direction().angleTo(circle.normal())
        <= Math.PI/4) {
114                             // Clockwise if angle is less than 90 degrees
115                             clockwiseness = "clockwise";
116                         } else {
117                             clockwiseness = "counterclockwise";
118                         }
119
120                         double sweptAngle = 0;
121                         if (circle.state() != State.STATE_START) {
122                             CircleGesture previousUpdate = new CircleGesture(
        controller.frame(1).gesture(circle.id()));
123                             sweptAngle = (circle.progress() - previousUpdate.progress
        ()) * 2 * Math.PI;
124                         }
125
126                         if(controller.config().getString("debugMode").equals("true"))
127                             debugInput(controller, "Circle id: " + circle.id()
128                                     + ", " + circle.state()
129                                     + ", progress: " + circle.progress()
130                                     + ", radius: " + circle.radius()
131                                     + ", angle: " + Math.toDegrees(sweptAngle)
132                                     + ", " + clockwiseness);
133
134
135                         if (circle.state() == State.STATE_START) {
136                                 listeGesture.push(gestureS);
137                         }
138                         break;
139                     }
140                     case TYPE_SCREEN_TAP:
141                     {
142                         ScreenTapGesture screenTap = new ScreenTapGesture(gestureS);
143
144                         if(controller.config().getString("debugMode").equals("true"))
145                             debugInput(controller, "Screen Tap id: " + screenTap.id()
146                                + ", " + screenTap.state()
147                                + ", position: " + screenTap.position()
148                                + ", direction: " + screenTap.direction());
149
150                         if (screenTap.state() == State.STATE_STOP)
```

```
151                             {
152                                 listeGesture.push(gestureS);
153                             }
154                             break;
155 
156                         }
157                         case TYPE_KEY_TAP:
158                         {
159                             KeyTapGesture keyTap = new KeyTapGesture(gestureS);
160 
161                             if(controller.config().getString("debugMode").equals("true"))
162                                 debugInput(controller , "Key Tap id: " + keyTap.id()
163                                    + ", " + keyTap.state()
164                                    + ", position: " + keyTap.position()
165                                    + ", direction: " + keyTap.direction());
166 
167                             if (keyTap.state() == State.STATE_STOP)
168                             {
169                                 listeGesture.push(gestureS);
170                             }
171                             break;
172 
173                         }
174                     }
175                 }
176                 resultPrint = false;
177             }
178             else if(resultPrint == false && lastIDHand > -1 )
179         {
180                 result = new ArrayList<>();
181 
182                 if(controller.config().getString("debugMode").equals("true"))
183                             debugInput(controller , "#####Hand out of Frame#####");
184 
185                 //si le nombre de geste depasse 12, on ne prend que les 12 premiers.
186                 int sizeResult = listeGesture.size() > 12? 12 : listeGesture.size();
187 
188                 for (int i = 0 ; i < sizeResult; ++i)
189                 {
190                     Gesture gesture = listeGesture.get(i);
191 
192                     switch( gesture.type())
193                     {
194                         case TYPE_SWIPE:
195                         {
196 
197                             SwipeGesture swipeS2 = new SwipeGesture(gesture);
198                             if(swipeS2.direction().getX() > 0.8)
199                             {
200                                   result.add("Swipe_Right");
201                             }
202                             else if(swipeS2.direction().getX() < -0.8)
203                             {
204                                    result.add("Swipe_Left");
205                             }
206                             if(swipeS2.direction().getY() > 0.8)
207                             {
208                                     result.add("Swipe_Up");
209                             }
210                             else if(swipeS2.direction().getY() < -0.8)
211                             {
212                                    result.add("Swipe_Down");
213                             }
214                             break;
215                         }
216                         case TYPE_CIRCLE:
217                         {
218                             CircleGesture circle = new CircleGesture(gesture);
219 
220                             if (circle.pointable().direction().angleTo(circle.normal())
        <= Math.PI/4) {
221                                result.add("Rotate");
222                             } else {
223                               result.add("Counter_Rotate");
224                             }
225                             break;
226                         }
227                         case TYPE_SCREEN_TAP:
228                         {
229                             result.add("Screen_Tap");
```

```
230                         break;
231                     }
232                     case TYPE_KEY_TAP:
233                     {
234                         result.add("Key_Tap");
235                         break;
236                     }
237 
238                 }
239             }
240             resultPrint = true;
241             listeGesture.clear();
242             handDetect = false;
243             switch (controller.config().getString("typeRecognize")) {
244                 case "createShortcut":
245                     ls.stopReconize(result);
246                     break;
247                 case "serviceShortcut":
248                     try {
249                         lss.analyseShortcut(result);
250                     } catch (URISyntaxException ex) {
251                         Logger.getLogger(LeapListener.class.getName()).log(Level.
    SEVERE, null, ex);
252                     }
253 
254                     break;
255             }
256              result.clear();
257     }
258        // System.out.println("okkkk");
259          //ls.stopReconize(this);
260     }
261 
262     public void debugInput(Controller controller, String input){
263         switch (controller.config().getString("typeRecognize")) {
264             case "createShortcut":
265                 ls.addDebugInput(input);
266                 break;
267             case "serviceShortcut":
268                 lss.addDebugInput(input);
269 
270                 break;
271         }
272     }
273 }
```

Annexe 2 : LeapShortcut - Enregistrer un raccourci en XML

```
1       public void saveMotion(String name, DefaultListModel fileList, String description
        , boolean sequentialOption){
2           Document document = null;
3           Element racine;
4
5           SAXBuilder sxb = new SAXBuilder();
6           try
7           {
8              document = sxb.build(new File("database.xml"));
9           }
10          catch(Exception e){}
11
12          if(document == null){
13              racine = new Element("shortcutDB");
14              document = new Document(racine);
15
16              try
17              {
18                 XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
19                 sortie.output(document, new FileOutputStream("database.xml"));
20              }
21              catch (java.io.IOException e){}
22          }
23          racine = document.getRootElement();
24
25          Element shortcutElement = new Element("shortcut");
26          Element shortcutName = new Element("name");
27          Element shortcutDescription = new Element("description");
28          Element shortcutFileList = new Element("fileList");
29          Element shortcutGestureList = new Element("gestureList");
30
31          shortcutName.setText(name);
32          shortcutDescription.setText(description);
33          Attribute seqOpt;
34
35          if(sequentialOption)
36              seqOpt = new Attribute("sequentialOpening","true");
37          else
38              seqOpt = new Attribute("sequentialOpening","false");
39
40          shortcutFileList.setAttribute(seqOpt);
41
42
43
44          for(int x = 0 ; x < fileList.size() ; x++){
45
46              Element file = new Element("file");
47
48              if(fileList.get(x).getClass() == java.net.URI.class){
49
50                  URI uri = (URI)fileList.get(x);
51
52                  Attribute classe = new Attribute("type","URI");
53                  file.setAttribute(classe);
54                  file.setText(uri.getPath());
55
56              }
57              else if(fileList.get(x).getClass() == java.io.File.class){
58
59                  File f = (File)fileList.get(x);
60                  Attribute classe = new Attribute("type","URL");
61                  file.setAttribute(classe);
62                  file.setText(f.getAbsolutePath());
63
64              }
65
66              shortcutFileList.addContent(file);
67          }
68
69          for(String s : listCurrentGesture){
70              Element gesture = new Element("gesture");
71              gesture.setText(s);
72              shortcutGestureList.addContent(gesture);
73          }
74
75          shortcutElement.addContent(shortcutName);
76          shortcutElement.addContent(shortcutDescription);
```

```
77            shortcutElement.addContent(shortcutFileList);
78            shortcutElement.addContent(shortcutGestureList);
79
80            racine.addContent(shortcutElement);
81
82            try
83            {
84               XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
85               sortie.output(document, new FileOutputStream("database.xml"));
86            }
87            catch (java.io.IOException e){}
88
89          //racine = document.getRootElement();
90       }
```

Annexe 3 : LeapShortcutService - Analyse d'un mouvement

```
1       //On analyse le mouvement reconnu par le LeapMotion
2       protected void analyseShortcut(ArrayList<String> result) throws
        URISyntaxException {
3
4
5
6           //ls.shortcurtMotionExist(result);
7
8           if(result.size() == 1 && result.get(0).equals("Key_Tap") == true &&
        sequentialShortcut != null){
9
10              int index = sequentialShortcut.getSequentialIndex();
11
12              if(index < sequentialShortcut.getFileList().size()){
13
14                  DefaultListModel dlm = sequentialShortcut.getFileList();
15
16                  if(dlm.get(index).getClass() == java.net.URI.class){
17                      URI uri = null;
18                      try {
19                          desktop.browse((URI)dlm.get(index));
20                      } catch(IOException ioe) {
21                          System.out.println("The system cannot find the " + uri + "
        file specified");
22                          //ioe.printStackTrace();
23                      }
24                  }
25                  else if(dlm.get(index).getClass() == java.io.File.class)
26                     execFile(dlm , index);
27
28                  index++;
29
30                  if(index < sequentialShortcut.getFileList().size()){
31                     sequentialShortcut.setSequentialIndex(index);
32                  }
33                  else{
34                      sequentialShortcut = null;
35                  }
36              }
37          }
38          else{
39              String name = ls.shortcurtMotionExist(result);
40
41              if(name != null){
42
43                  Shortcut shortcut = ls.getShortcut(name);
44                  DefaultListModel dlm = shortcut.getFileList();
45                  System.out.println("1");
46
47                  if(shortcut.isSequential()){
48                      sequentialShortcut = shortcut;
49                      int index = sequentialShortcut.getSequentialIndex();
50
51                      if(dlm.get(index).getClass() == java.net.URI.class){
52                              URI uri = null;
53                              try {
54                                  desktop.browse((URI)dlm.get(index));
55                              } catch(IOException ioe) {
56                                  System.out.println("The system cannot find the " +
        uri + " file specified");
57                                  //ioe.printStackTrace();
58                              }
59                      }
60                      else if(dlm.get(index).getClass() == java.io.File.class){
61                           execFile(dlm , index);
62                      }
63                      sequentialShortcut.setSequentialIndex(index+1);
64                  }
65                  else{
66
67                      //sequentialShortcut = null; //Permet d'effacer le shortcut
        sequentiel si un nouveau shortcut capté
68
69                      for(int x = 0 ; x < dlm.size() ; x++){
70                          if(dlm.get(x).getClass() == java.net.URI.class){
71                              URI uri = null;
72                              try {
```

```
73                                  desktop.browse((URI)dlm.get(x));
74                              } catch(IOException ioe) {
75                                  System.out.println("The system cannot find the " +
        uri + " file specified");
76                                  //ioe.printStackTrace();
77                              }
78                          }
79                          else if(dlm.get(x).getClass() == java.io.File.class)
80                              execFile(dlm , x);
```

Annexe 4 : LeapShortcutService - Exécution d'un fichier script

```
1           try {
2
3               File f = (File)dlm.get(x);
4               int i = f.getPath().lastIndexOf('.');
5               String extension = "";
6
7               if (i > 0) {
8                   extension = f.getPath().substring(i+1);
9               }
10
11              if(extension.equals("cmd") || extension.equals("bat")){
12
13                  ProcessBuilder pb = new ProcessBuilder("cmd.exe", "/c", "start", f.
        getPath());
14                  pb.directory(new File(f.getParent()).getAbsoluteFile());
15                  Process p = pb.start();
16              }
17              else{
18                  desktop.open((File)dlm.get(x));
19              }
```

Annexe 5 : Manuel Utilisateur

Ce manuel décrit les manipulations à faire pour enregistrer un raccourci.

Étape 1 - Lancer le LeapShortcut, l'interface apparaît.

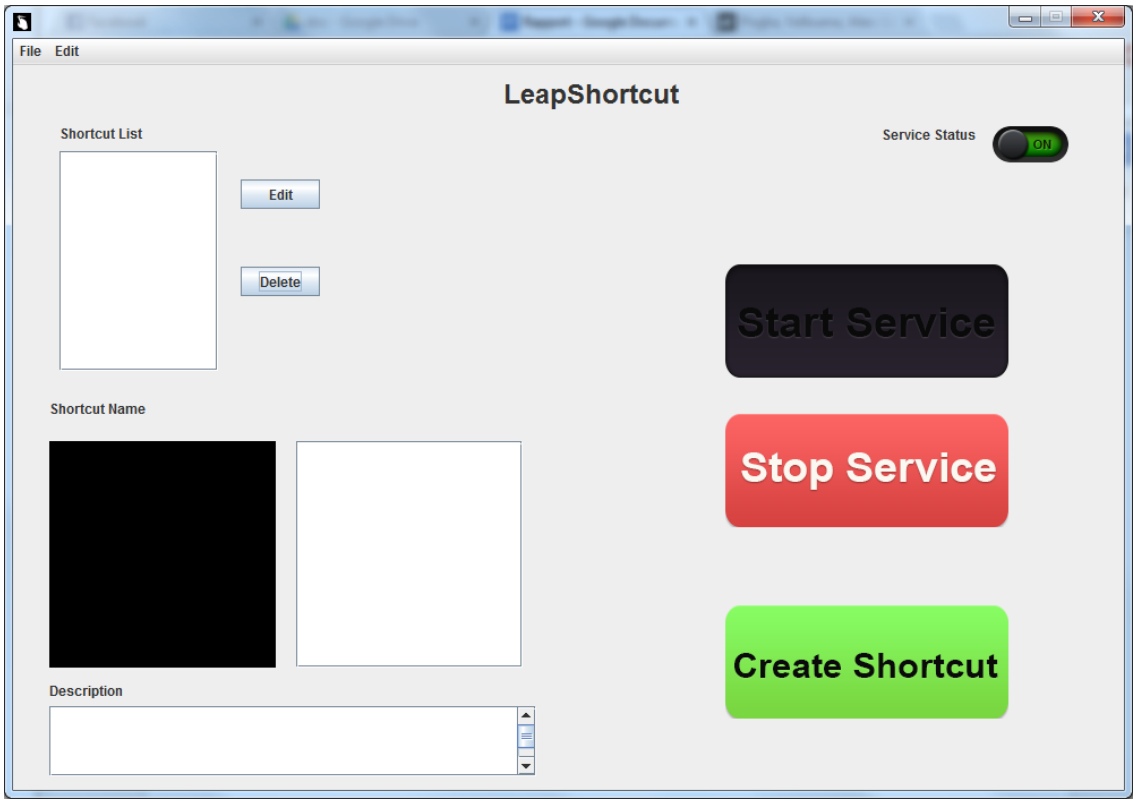

FIGURE 14 – Étape 1

Étape 2 - Cliquer sur le bouton “Create Shortcut” pour afficher le formulaire de création de raccourci.

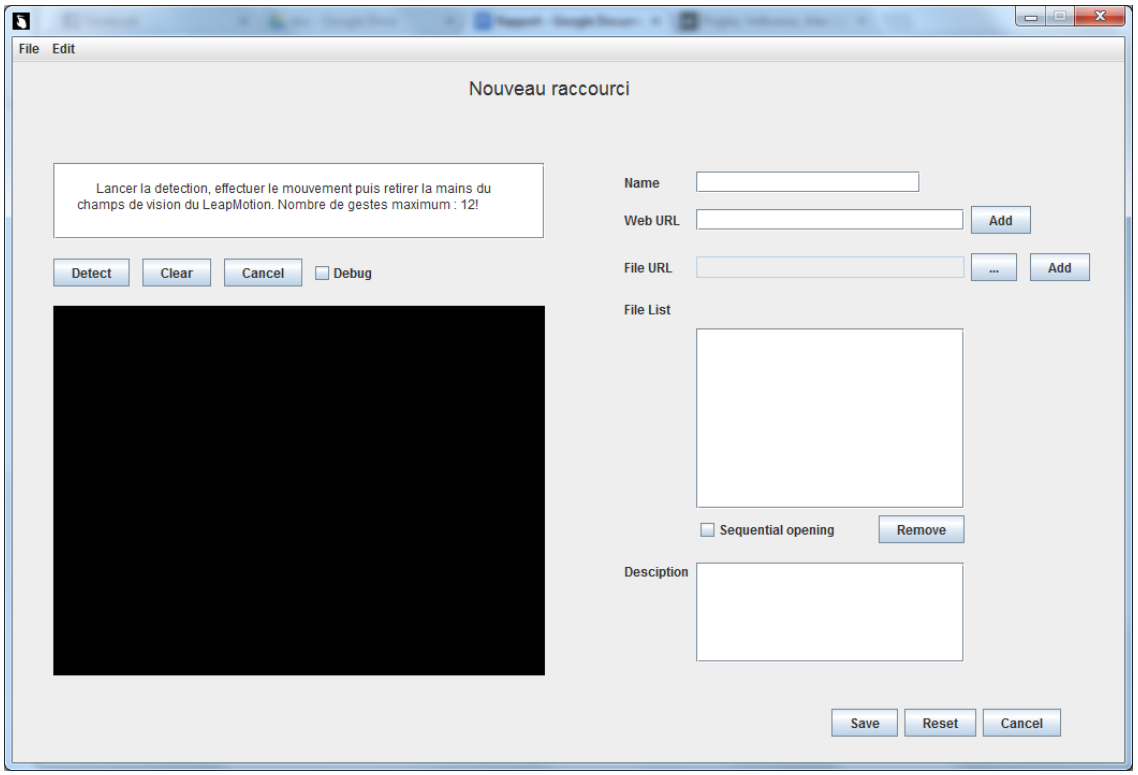

FIGURE 15 – Étape 2

Étape 3 - Remplir les champs “Nom” et “Description”. Pour remplir la liste des fichiers à exécuter, deux choix possibles :

— Si c’est une URL web, remplir le champs web URL puis cliquer sur le bouton “Add”
— Si c’est un fichier, cliquer sur le bouton “...” puis choisir le fichier désiré et enfin, cliquer sur le bouton “Add”

On peut répéter cette étape plusieures fois, pour ajouter plusieurs fichiers.

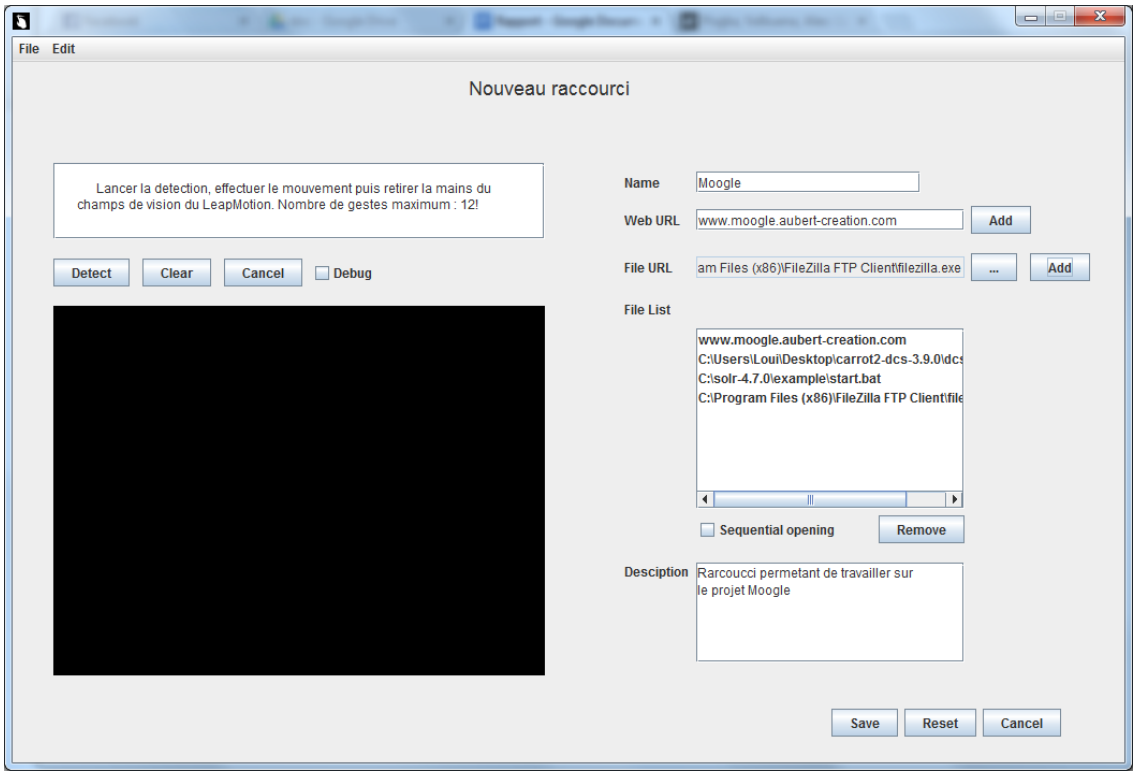

FIGURE 16 – Étape 3

Dans l’exemple, on crée un raccourci “Moogle” avec une URL, deux scripts “dcs.cmd” et start.bat” et pour finir le client FTP FileZilla.

Étape 4 - Cliquer sur Detect, mettre la main dans le champ de vision du Leap Motion, Effectuer le mouvement, puis retirer sa main du champ de vision, le mouvement est affiché. création de raccourci.

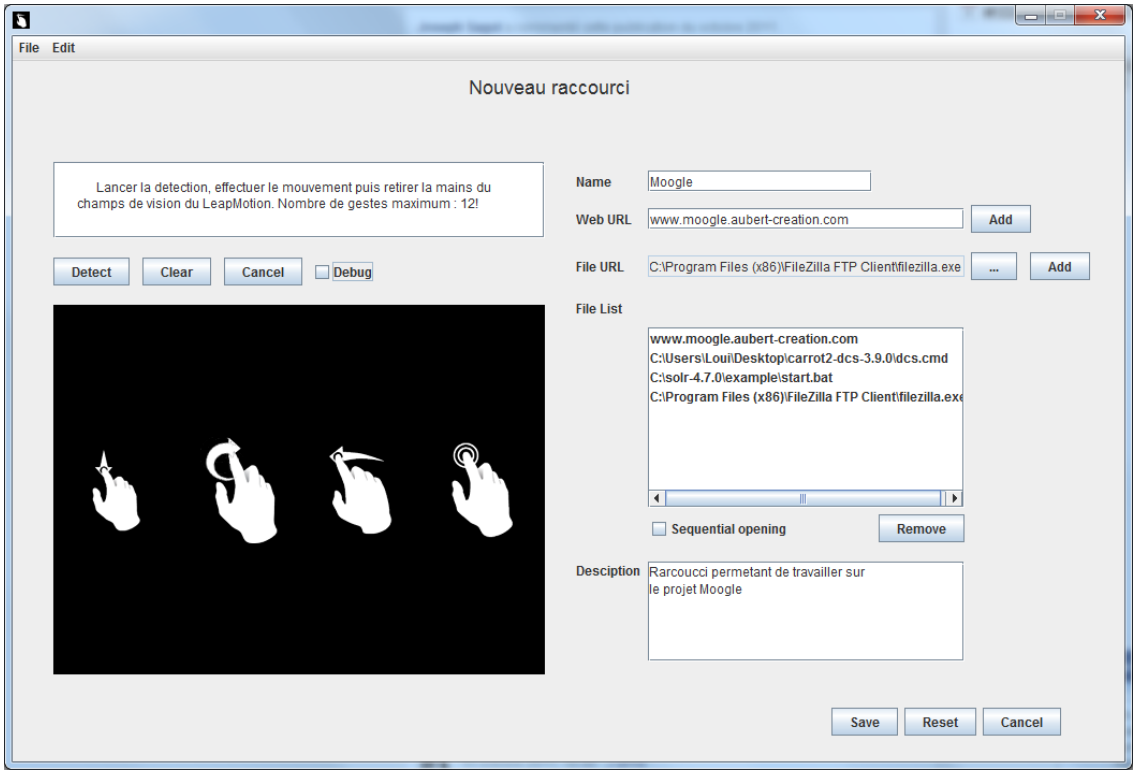

FIGURE 17 – Étape 4

Le mouvement est composé d'un geste vers le bas, d'un cercle, d'un geste vers la gauche et d'une tape de touche du clavier.

Étape 5 - Appuyer sur “Save” pour sauvegarder et retourner au menu principal. Le raccourci est affiché dans la liste et on peut cliquer dessus pour avoir sa description.

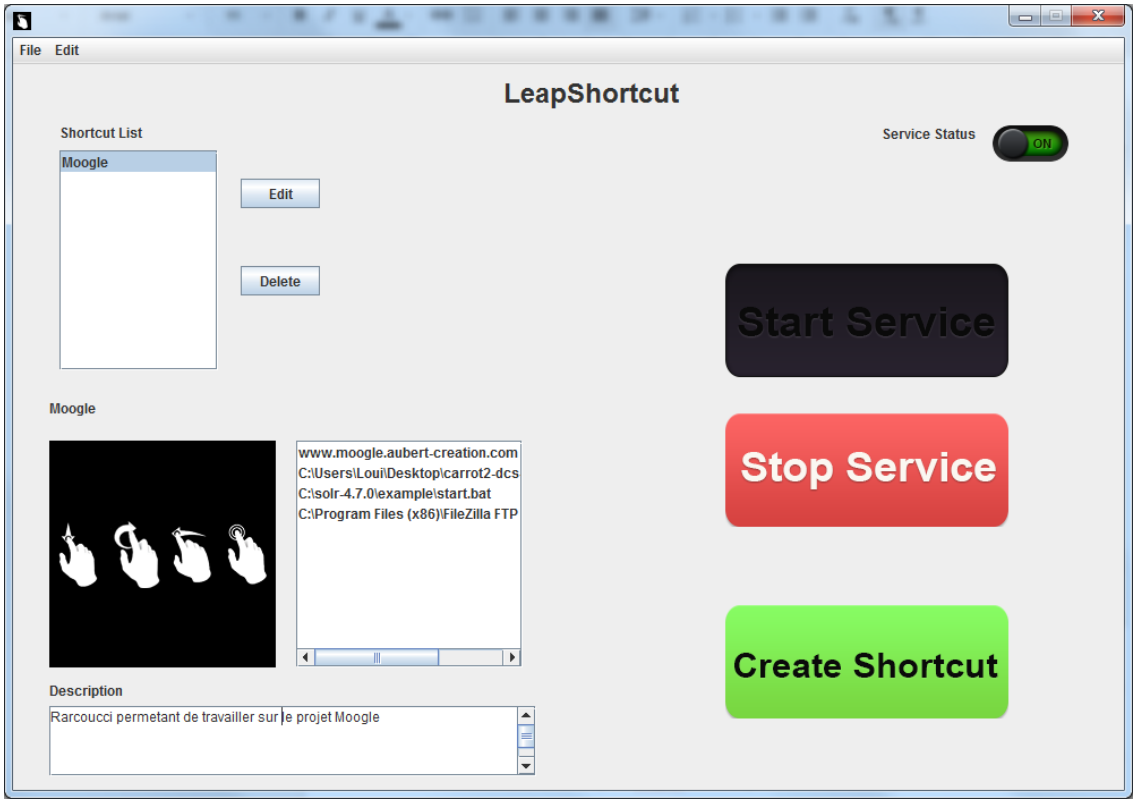

FIGURE 18 – Étape 5

On peut maintenant effectuer ce mouvement pour lancer les fichiers associés.

En fermant l’interface graphique, l’application se retrouve dans la barre d’icônes, permettant au service de fonctionner de manière transparente. On peut effectuer un clique droit sur l’icône afin d’ouvrir un petit menu ouvrant le programme ou effectuant les actions principales de l’application.

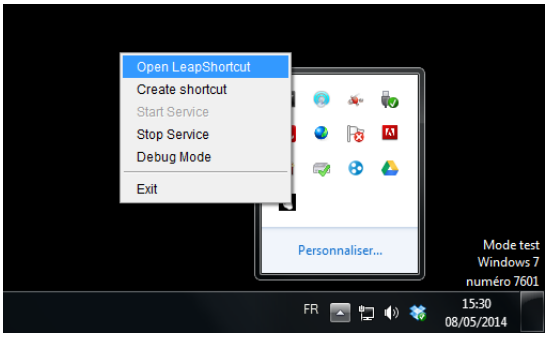

FIGURE 19 – Menu de l’application dans la barre d’icônes