REPUBLIQUE TUNISIENNE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE DE TUNIS EL MANAR

INSTITUT SUPERIEUR INFORMATIQUE ISI
المعهد العالي للإعلامية

INSTITUT SUPERIEUR D'INFORMATIQUE

RAPPORT DE STAGE DE FIN D'ETUDES

Présenté en vue de l'obtention de
La Licence Appliquée en Informatique
Parcours Systèmes Informatiques et logiciels

Développement d'une plate-forme multitouch

Par
Wathek Bellah LOUED

Entreprise d'accueil :
Gedeom Communication

Encadrant à l'entreprise : M. Jérémy DERAMCHI
Encadrant à l'ISI : Dr. Wassim Youssef

Année Universitaire 2010-2011

2011

Sommaire

Remerciements

Je tiens à remercier dans un premier temps, toute l'équipe pédagogique de l'Institut Supérieur d'Informatique et les intervenants professionnels responsables de la formation « Licence Appliquée en Système Informatique et Logiciels ».

Je remercie particulièrement mes maîtres de stage, **Dr. Wassim YOUSSEF** pour son encadrement pendant le stage, ses précieux conseils et son suivi continue de l'évolution du travail. Ainsi que **M. Jérémy DERAMCHI** président de **Gedeom Communication** qui m'a très bien accueilli pour effectuer mon stage en France et qui m'a beaucoup aidé à apprendre dans d'excellentes conditions afin de pouvoir accomplir ce stage d'une façon parfaite.

Je tiens à remercier tout particulièrement et à témoigner toute ma reconnaissance à toute ma famille, mon père **Habib**, ma mère **Zohra** et mon frère **Nader**, pour leur soutient pendant le stage.

Liste des figures

Présentation de l'entreprise et de ses secteurs d'activités

Gedeom Communication est une société créée en 2008 et opérant dans le domaine des nouvelles technologies. Cette société basée à Toulouse a pour objectif principal la recherche applicative dans le domaine de la technologie de pointe et les technologies se référant l'interaction homme-machine.

Gedeom Communication est composée de plusieurs services chacun travaillant sur un domaine spécifique. Parmi les spécialités de cette société :

- Développement de solutions Web.
- Développement d'applications spécifiques.
- Installation et configuration de solutions de sécurité réseau.
- Marketing et print.

La société aide ses clients à développer leurs activités et à étendre leurs champs d'action en proposant plusieurs services permettant l'optimisation de la stratégie de communication, qui se basent essentiellement sur les nouvelles technologies.

La parfaite maîtrise de ces différentes technologies et l'optimisation des méthodes de travail garantissent à ses clients des solutions sur mesure et parfaitement efficaces, dans le respect de leurs exigences.

Gedeom Communication est constituée essentiellement de jeunes enthousiastes qui cherchent à innover et à donner le meilleur d'eux même, en fournissant la meilleure qualité.

En choisissant les nouvelles technologies comme secteur d'activités, la société a su s'imposer en Europe et concurrencer les plus grandes entreprises de la région.

Introduction Générale

L'histoire de l'humanité avance à une grande vitesse. L'homme existe sur cette planète depuis environ un million d'années et il n'a jamais arrêté d'innover. L'invention de l'imprimerie, il y a environ 500 ans, a permis d'accélérer la diffusion des connaissances. L'apparition de la machine à vapeur, il y a 200 ans, a constitué une révolution industrielle. L'ordinateur n'a que 50 ans d'âge et il a évolué d'une façon extrêmement rapide. Actuellement les recherches vont dans le sens de permettre une meilleure interaction entre l'homme et la machine. Les cartes perforées, les boutons poussoirs, les voyons et les imprimantes font partie des premières interfaces de contrôle de la machine. L'invention de la souris, du clavier et des écrans ont créé une nouvelle forme d'interaction homme-machine.

Les interfaces homme-machine constituent l'intermédiaire entre l'homme et la machine. Ces interfaces permettent de traduire les demandes de l'homme à la machine. A fin de permettre une communication aisée, de nouvelles interfaces homme-machine sont inventées. Parmi ces nouvelles technologies, il y a les dispositifs monotouch et d'autres multitouch, permettant à l'utilisateur d'interagir d'une façon presque naturelle avec la machine.

Ce travail porte sur le développement d'une plate-forme multitouch qui se base sur des solutions Libres. Cette plate-forme est composée de deux grandes parties :

- Le matériel : Il ne s'agit pas de se baser sur un matériel existant, mais plutôt de fabriquer l'interface homme-machine.
- Le logiciel : Développement du logiciel permettant d'exploiter ce matériel. La plus grande partie de ce logiciel est la reconnaissance de la gesture (des mouvements spécifiques effectués par l'utilisateur sur la surface tactile).

La réalisation de ce travail s'est faite au sein de l'équipe **Gedeom Communication**. Il porte, d'une façon générale, sur la technologie multitouch et plus précisément sur les tables multitouch. L'objectif principal est de développer une table multitouch (matériel et logiciel), utilisée principalement pour les présentations et les démonstrations.

Ce rapport est organisé en quatre chapitres. Au cours du premier chapitre, nous présentons un aperçu historique des interfaces homme-machine (IHM), avec une brève description de chacune

des technologies IHM. Ensuite dans ce même chapitre nous présentons les différentes méthodes pour obtenir du multitouch et nous définissons la structure d'un environnement de bureau. A la fin nous expliquons rapidement l'importance de la gesture dans les interfaces homme-machine multitouch.

Dans le deuxième chapitre nous évoquons les spécifications de la table multitouch, ceci en énumérant les besoins fonctionnels, les besoins non fonctionnels ainsi que la spécification de la table multitouch.

Dans le troisième chapitre nous présentons d'une façon globale le système et nous décrivons les technologies utilisées, d'un point de vue Système d'exploitation et langage de programmation. Ensuite nous décrivons le protocole de communication de la table et finalement le « tracker » qui est un composant primordial au fonctionnement de la table multitouch, puisqu'il constitue l'élément qui envoie les messages multitouch à l'environnement de bureau.

Le quatrième chapitre décrit la conception et la réalisation de la pile logicielle de la table multitouch. Au sein de ce chapitre, nous décrivons la passerelle « qTuio » qui lie le « tracker » à tout l'environnement de bureau. Et nous expliquons d'une façon détaillée le module de la gesture, que ce soit pour la reconnaissance des gestures simples, que pour la reconnaissance des gestures complexes. Ensuite nous présentons l'environnement de bureau, ainsi que son architecture et son fonctionnement. A la fin nous évoquons les défis et les perspectives.

Chapitre 1

Étude de l'interface multitouch et de l'environnement de Bureau

Dans ce chapitre :

1- L'interface Homme-machine

2- La technologie Multitouch

3- Les différentes technologies du multitouch

- 3.1- Technologie à onde de surface
- 3.2- Technologie résistive analogique
- 3.3- Technologie capacitive
- 3.4- Technologie à infrarouge
- 3.5- Technologie Frustrated Total Internal Reflection (FTIR)

4- La table multitouch

5- Qu'est ce qu'un environnement de bureau ?

6- La gesture

1- L'Interface Homme-Machine (IHM)

L'interface Homme-Machine définit l'ensemble des éléments qui permettent à un utilisateur d'interagir avec une machine. Ces éléments constituent un domaine de recherche qui n'a cessé d'évoluer.

Dans les années 60 le clavier s'est imposé en compagnie de l'écran et cela a constitué un mode d'interaction plus convivial avec les machines. Les premières recherches dans ce domaine ont été menées par ***Xerox*** qui a aussi inventée la souris ainsi que la première interface graphique dans les années 70 en se basant sur les travaux de ***Douglas Engelbart***, qui a mis en place les principes de l'interface graphique moderne en 1963. Ces recherches ont été reprises par ***Steve Jobs*** en 1979, et ont permis de doter le premier Macintosh, lancé en 1984, d'une interface graphique et d'une souris.

Illustration 1: *Station Xerox Alto, 1ère interface graphique (1973).*[1]

Illustration 2: *Macintosh 128K.*[2]

L'IHM peut être divisée en trois grandes catégories :

- Les interfaces d'acquisition (clavier, souris, télécommande, etc).
- Les interface de restitution (écrans, hauts parleurs, etc).
- Les interfaces combinées (écrans tactiles, etc).

Les recherches les plus récentes dans le domaine de l'IHM vont dans le sens de l'interprétation des gestes naturels d'un être humain. Nous pouvons remarquer cela avec l'apparition des ***Wiimotes***[3] (Télécommande ***Wii***) qui a été inventée par ***Nintendo*** et qui fait en sorte que l'utilisateur puisse contrôler la console de jeux en bougeant le capteur sans fils.

Par la suite il y a eu l'apparition de la ***PlaystationEye***[4] qui a permis la reconnaissance des mouvements sans que l'utilisateur ne soit obligé de tenir un capteur (comme la ***Wiimote***), et ceci en analysant les images captées par la caméra.

L'évolution continue jusqu'à voir apparaître le ***Kinect***[5] de ***Microsoft*** qui est une caméra 3D qui permet non seulement de reconnaître les gestes mais aussi de connaître la profondeur des objets et ainsi une vue 3D de l'ensemble est obtenue.

D'autres recherches vont encore plus loin et permettent à l'utilisateur de contrôler une machine rien qu'avec la pensée. Le projet ***OpenViBE***[6] lancé en 2006 et rendu publique en 2009, exploite un casque électroencéphalographique pour enregistrer les signaux électriques renvoyés par le cerveau pour les analyser et interpréter le résultat.

Illustration 3: *Wiimote.*[7]

Illustration 4: *PSEye.*[8]

Illustration 5: *Kinect.*[5]

Par ailleurs, les tables multitouch font leur apparition. Elles permettent de manipuler un contenu informatique à l'aide d'un écran tactile, et même d'interagir avec l'utilisateur et les objets mis dessus.

Bien entendu il existe aussi des recherches dans la reconnaissance vocale qui constitue une autre façon d'interagir avec les appareils électroniques. Parmi les systèmes les plus performants, il

existe la recherche vocale créée par ***Google*** qui est présente dans les systèmes d'exploitation ***Android***.

Dans ce travail, nous nous intéressons aux tables multitouch. Nous détaillons dans ce qui suit les différentes technologies existantes ainsi que la gesture qui constitue une grande partie du domaine de la multitouch.

2- La technologie Multitouch

La technologie du multitouch existe depuis 1982. Elle est le fruit de recherches des laboratoires **Bell** et aussi de l'université de Torronto, mais elle n'a pas arrêté d'évoluer depuis. ***Microsoft*** a exploité cette technologie pour la table ***MS Surface***[9]**.**

La technologie multitouch vise à reconnaître, analyser et interpréter l'état de plusieurs points sur une surface. Cette technologie qui existe actuellement dans plusieurs appareils, essentiellement les smart phones, fait en sorte qu'il y ait une interaction naturelle entre l'utilisateur et la machine.

La surface tactile devient en quelque sorte un dispositif de pointage. Les trackpads des ordinateurs portables sont des surfaces tactiles qui, selon la technologie utilisée, peuvent être monotouch ou multitouch.

Généralement chaque doigt qui touche la surface tactile est considéré comme étant un point et possède certaines propriétés. Selon la taille du dispositif et selon le système développé on peut avoir plusieurs utilisateurs sur un même dispositif. Néanmoins ceci reste restreint pour l'instant à cause de certaines complexités techniques[1], qui touchent à la façon avec la quelle le logiciel peut interpréter ce que veux faire l'utilisateur et ceci peut entraîner une ambiguïté.

Il existe aussi des surfaces tactiles qui font office de dispositif de restitution, c'est à dire qu'elles jouent le rôle d'un écran (exemple : écrans tactiles, table multitouch, smart phones, tablettes, etc).

1 Voir Chapitre 4 : 4.1- Défis

3- Les différentes technologies du multitouch

Pour avoir de la multitouch il existe plusieurs technologies inventées. Ces technologies peuvent fonctionner d'une façon différente selon les facteurs environnementaux. Ci-dessous une description des approches principales.

3.1- Technologie à onde de surface

Cette technologie apparue en 2001, utilise des transducteurs piézo-électriques qui sont placés sur le bord horizontal et le bord vertical, et transforment un signal électrique en une onde ultra sonique qui se propage sur la surface. Par la suite cette onde est captée par les transducteurs, vertical et horizontal, pour pouvoir déterminer les coordonnées d'un point. [10]

L'inconvénient de cette technologie c'est qu'une simple rayure sur la surface entraîne des changements dans la propagation de l'onde et donc peut entraîner un dysfonctionnement du système.

3.2- Technologie résistive analogique

Cette technologie qui est la plus répandue, se base sur la mise en contact de deux couches par l'action du doigt ou de tout autre objet. La dalle de l'écran est recouverte d'une couche électriquement conductrice, puis on rajoute par-dessus une feuille de polyester souple, recouverte sur sa face interne d'une couche conductrice.

Cette feuille est isolée de la dalle de verre (et donc de la première couche conductrice) par un grand nombre de petits plots isolants et transparents. Lorsque l'on touche ce type d'écran, on enfonce légèrement la couche de polyester et celle-ci vient alors en contact avec la couche conductrice de la dalle de verre.

Plusieurs types de périphériques utilisent cette technologie entre autres ***l'iPhone***, certains smart phones (***HTC***, ***LG***, etc), des PDA de chez ***PALM***, etc.

L'inconvénient majeur c'est que la conductivité électrique s'use au cours du temps et c'est ce qui fait que la précision de la détection des coordonnées du point touché se réduit.

3.3- Technologie capacitive

Ce système est basé essentiellement sur la décharge des condensateurs. L'écran étant revêtu d'une couche métallique conductrice qui est soumise à une tension électrique permettant ainsi la création d'un champ électrique uniforme sur toute la surface. Lorsque l'utilisateur touche l'écran, il devient lui-même une partie du circuit électrique et c'est ce qui crée un déficit quantifiable qui permet la détermination du point de contact. Cette technologie est utilisée dans plusieurs appareils tel que l'***iPhone*** et l'***iPad*** et certains smart phones de chez ***Motorola***, ***HTC*** et ***LG***.

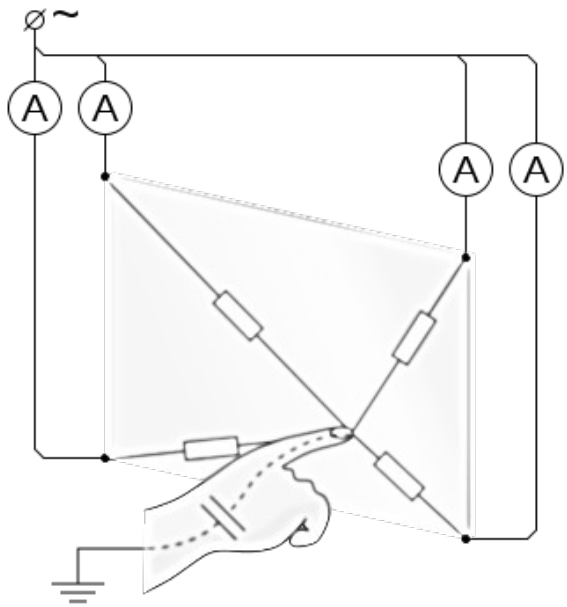

Illustration 6: *Écran tactile capacitif*[11]

3.4- Technologie à infrarouge

Cette technique se base sur la chaleur, en effet la température du corps qui touche l'écran permet, grâce a des capteurs thermiques, de reconnaître la position de contact. On reproche à cette méthode d'être lente et bien entendu il est nécessaire d'avoir les mains chaudes pour pouvoir interagir avec l'appareil. Il est aussi à noter que les stylets sont inefficaces avec cette technologie.

Les écrans tactiles à infrarouge sont les plus résistants et de ce fait ils sont souvent utilisés pour des applications militaires.

3.5- Technologie Frustrated Total Internal Reflection (FTIR)

Cette technologie repose sur un phénomène optique bien connu et exploité par exemple dans la fibre optique : la réflexion totale. Lorsque la lumière se propage d'un matériau à un autre sa direction est modifiée à chaque interface, car tous les matériaux n'ont pas le même indice de réfraction.[12]

Il est donc nécessaire d'avoir une source lumineuse et un dispositif qui va capter les ondes réfléchies et envoyer les données à une unité de calcul et d'analyse qui déterminera les coordonnées d'un point.

Cette technologie est très utilisée dans la construction des tables multitouch, notamment dans la table ***MS Surface*** de chez ***Microsoft***.

4- La table multitouch

Toutes les tables multitouch sont constituées de deux grandes parties. La partie affichage, qui est généralement un dispositif qui va montrer à l'utilisateur le résultat de ses opérations. Et la partie capture, qui va capturer ce qui existe sur la surface de la table.

La majorité des tables multitouch existantes utilisent la technologie FTIR (Frustrated Total Internal Reflection). Une table multitouch est constituée d'un ensemble d'éléments qui permettent une réflexion totale de la lumière. Pour cela il est nécessaire d'avoir une source lumineuse (LED infrarouge le plus souvent) qui est installée de telle sorte qu'il y ait une réflexion totale des ondes dans le milieu dans lequel l'onde se propage. Le milieu utilisé est généralement une plaque de plexiglas spéciale qui joue le rôle de guide d'onde.

En se basant sur les lois de ***Snell-Descartes***[2] qui décrivent le comportement de la lumière lors d'un passage d'un milieu à un autre on peut déterminer l'angle de diffusion de la lumière pour qu'il y ait une réflexion totale.

Lorsque le doigt de l'utilisateur se met en contact avec la surface, les ondes seront diffusées dans toutes les directions à l'intérieur de la plaque de plexiglas. Certaines de ces ondes verront leurs

2 Voir Annexe A

angles de réflexion changer et ainsi elles vont pouvoir quitter le milieu où elles se trouvent et se propager à l'intérieur de la table.

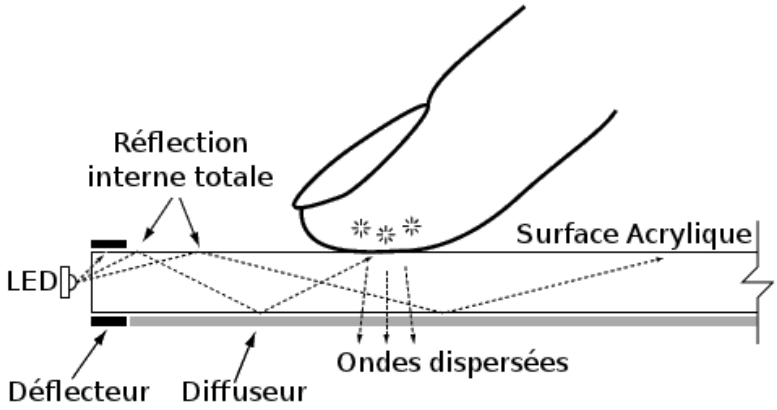

Illustration 7: *Diffusion/Déviation des ondes*[13]

La caméra infrarouge va capter ces ondes et va transmettre au « tracker » les informations à traiter. Le « tracker » a pour but d'analyser l'image renvoyée par la caméra et déterminer les coordonnées des points, et par la suite de diffuser le résultat.

Certaines tables utilisent d'autres technologies d'éclairage qui permettent aux caméras de capter les « Fiducials »[3] (qui sont des dessins facilement reconnaissables) et de lancer une action selon le « Fiducial » reconnu. Outre les « TouchPoints » et les « Fiducials », il existe les « BLOB » (Object Geometry) qui sont des formes géométriques auxquelles l'utilisateur attribut des identifiants pour une reconnaissance ultérieure.

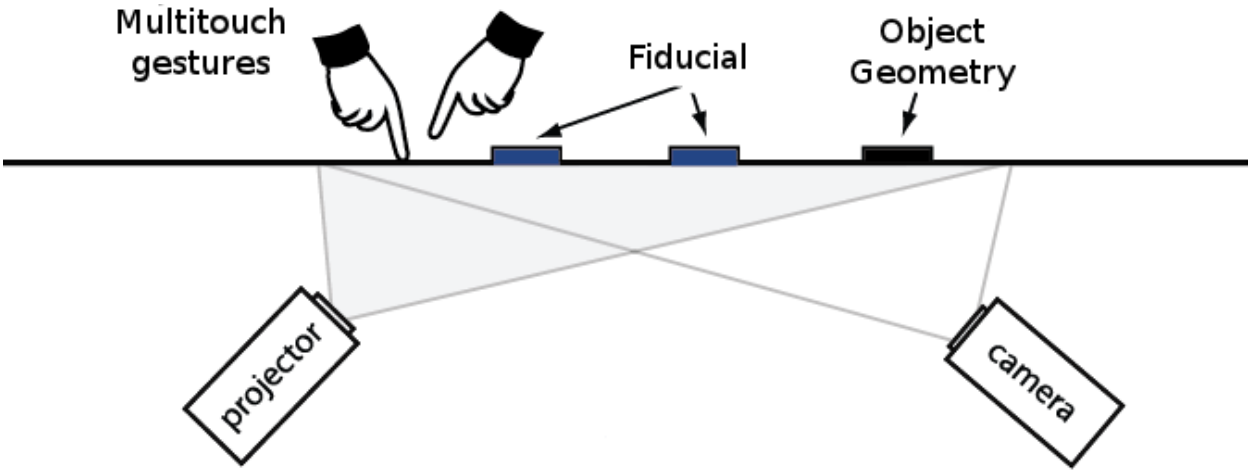

Illustration 8: *Description de la table multitouch d'un point de vue matériel*[14]

Il est aussi possible d'utiliser plusieurs dispositifs de capture (plusieurs caméras), surtout pour avoir une surface plus grande et permettre à plusieurs utilisateurs d'utiliser la table.

3 Voir Annexe B

La lumière utilisée pour éclairer la surface de projection, qui est une lumière blanche, contient tout le spectre lumineux, y compris les infrarouges. Il est donc nécessaire d'effectuer certains réglages afin d'avoir une image propre. Pour cela il existe deux solutions, la solution qui exploite les filtres et donc les infrarouges de la lumière blanche de l'éclairage du dispositif de projection seront inhibées et ainsi il ne peut plus y avoir des perturbations au niveau de la caméra. Ou bien l'utilisation de moyens logiciels et l'exploitation de certaines techniques de traitement d'images et de faire une opération de soustraction de l'image captée par la caméra à l'état repos (aucun objet sur la table) et les autres images captées par la caméra.

Il est possible d'utiliser les deux techniques pour avoir un meilleur rendu et une image de très bonne qualité, qu'on peut utiliser pour la détermination des coordonnées des points touchés.

5- Qu'est ce qu'un environnement de bureau ?

Un environnement de bureau (ou Desktop Environment) est un ensemble de programmes qui permettent à l'utilisateur, en proposant des outils graphiques, d'effectuer des tâches rapidement et facilement.

Après l'apparition du premier ***Macintosh*** avec une interface graphique, plusieurs autres systèmes d'exploitation ont suivi l'idée et ont fait évoluer les environnements de bureaux pour passer à des systèmes multi-fenêtres avec un design et une ergonomie bien étudiée.

Cette évolution n'a cessé de croître surtout avec l'invention des bureaux 3D, qui initialement était un projet communautaire et libre. Par la suite ***Microsoft*** a inventé son interface ***Aero***[15] qui l'a intégrée à ***Windows Vista*** et qui reprend un peu le même principe en proposant des effets 3D et des effets visuels.

Un Environnement de bureau est en réalité une partie de l'ensemble de couches d'une interface graphique. L'exemple du système de fenêtrage X (***X.org*** anciennement ***XFree86***) est un bon exemple pour comprendre le fonctionnement.

Tout d'abord on a besoin d'une couche qui sera l'intermédiaire entre le matériel et le reste (gestionnaire des fenêtres et environnement de bureau). Cette couche est le système de fenêtrage ;

elle permet à l'utilisateur d'interagir via les les périphériques de capture (clavier, souris, etc). Le système de fenêtrage fournit des primitives graphiques telles que le rendu de polices de caractères, le tracé de lignes, et dans le cas de X, cette couche permet même l'exploitation du réseau et ainsi elle devient une couche distante pour une autre machine. Par exemple si on a deux ordinateurs avec ***GNU/Linux*** dessus et que la machine A dispose de ***X.org*** (le système de fenêtrage) et que la machine B ne dispose pas de cette couche. La machine A peut se connecter sur la machine B et fournir, à cette dernière, cette couche et donc elle peut exécuter des applications graphiques en utilisant le ***X.org*** local. Cette technique s'appelle le X Forwarding.

Au dessus de la couche « Système de Fenêtrage », il existe la couche « Gestionnaire des Fenêtres » (voir Illustration 9). L'arrangement des fenêtres, c'est à dire leurs positions, leurs dimensions, leurs déplacements et tout ce qui a rapport avec le comportement des fenêtres, est géré par cette couche. Le gestionnaire des fenêtres offre plusieurs fonctionnalités telles que les bureaux virtuels, le fond d'écran, un gestionnaire de sessions, etc. Un exemple de système de fenêtrage : ***Windows Explorer***, ***Compiz Fusion***, ***Kwin***, ***Metacity***, etc.

La dernière couche avant l'utilisateur est la couche « Environnement Graphique ». C'est ce qui permet à l'utilisateur de personnaliser son espace de travail et d'interagir avec son ordinateur.

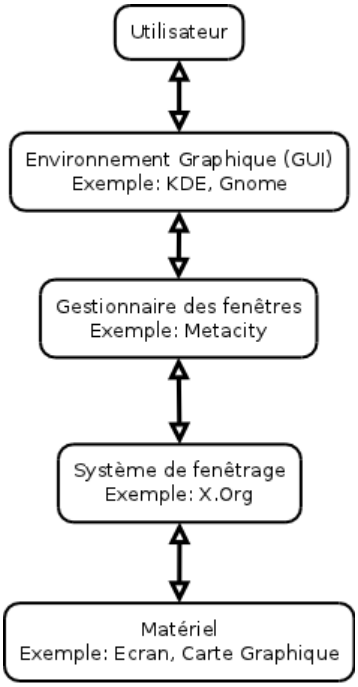

Illustration 9: *Couches d'une interface graphique*

6- La Gesture

La gesture est un langage de communication particulier qui se base sur les mouvements du corps à la place de la parole. D'après plusieurs études effectuées[16], la communication se fait à 80% par le corps, 10% par la parole et 10% par l'intonation. L'être humain fait passer l'information par son corps et par les mouvements de ses membres dés sa naissance (un bébé quand il n'a plus faim ou plus soif, et que sa maman lui donne à manger ou à boire, commence à bouger sa tête pour éviter ce que sa maman lui donne).

Ces gestures ont bien été étudiées par des psychologues et des scientifiques et c'est ce qui a constitué la base des recherches dans le domaine de la reconnaissance des gestures. Parmi les gestures fondamentales, il y a :

- *TapGesture* qui consiste à mettre un point de contact et de l'enlever sans bouger et ceci pendant une petite durée.
- *TapAndHoldGesture* qui ressemble au *TapGesture* sauf que là on maintient la position.
- *PanGesture* qui consiste à déplacer le point de contact ajouté (un mouvement de déplacement).
- *SwipeGesture* qui ressemble au *PanGesture* sauf qu'il tient compte de la vitesse et de l'accélération.
- *PinchGesture* utilisé essentiellement pour faire un agrandissement ou une rotation.

Illustration 10:
TapAndHoldGesture[17]

Illustration 11:
TapGesture[17]

Illustration 12:
PinchGesture[17]

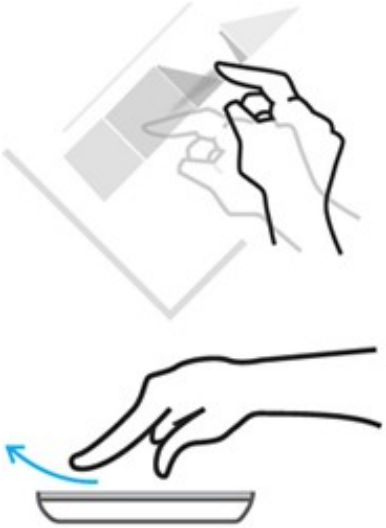

Illustration 13:
SwipeGesture[17]

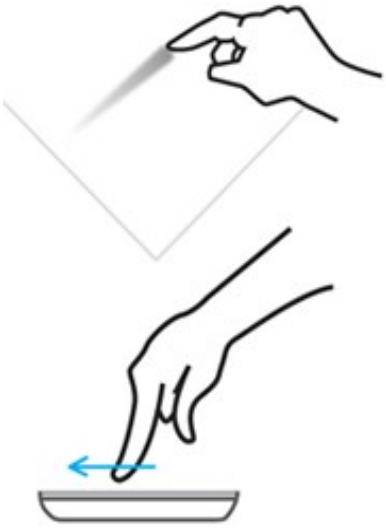

Illustration 14:
PanGesture[17]

Hormis le *PinchGesture* (qui nécessite deux points de contact), toutes les autres gestures se font avec seulement un seul point de contact. Il est aussi possible de rajouter d'autres gestures qui dérivent essentiellement des gestures de bases, par exemple un *TwoFingerTapGesture*.

D'autres gestures plus complexes ont été inventées aussi, par exemple la reconnaissance de formes géométriques, c'est à dire si l'utilisateur dessine un cercle, ou un triangle, le système peut reconnaître ces gestures et selon la programmation faite, un évènement sera exécuté.

Chapitre 2

Spécifications de la table multitouch

Dans ce chapitre :

1- Besoins fonctionnels

2- Besoins non fonctionnels

3- Spécifications de la table multitouch

Les objectifs du projet sont la construction d'une table multitouch et le développement de logiciels pour cette table, qui consiste en un environnement de bureau. Le logiciel devrait être évolutif et surtout modulaire, c'est à dire que les autres développeurs, peuvent concevoir des modules supplémentaires et les intégrer rapidement au système.

Pour cela il est nécessaire de créer une base (un ensemble de classes) que les développeurs peuvent utiliser pour réaliser leurs propres modules. Les modules sont en réalité des applications pour l'environnement de bureau.

Ce projet est divisé en deux grandes parties :

- Partie matérielle : nécessite une étude de l'existant, autrement dit des technologies existantes et aussi une étude mathématique et physique des phénomènes de la propagation des ondes.
- Partie logicielle : consiste en une étude des environnements existants, du protocole de communication de la table, du choix du langage de programmation et du framework.

La table construite se base sur la technologie FTIR (Frustrated Total Internal Reflection).

1- Besoins fonctionnels

La table multitouch doit offrir plusieurs fonctionnalités. Parmi les besoins fonctionnelles de cette interface :

- Reconnaissance des « TouchPoint », autrement dit les doigts des utilisateurs posés sur la surface de la table.
- Définition et reconnaissance des « BLOB » qui sont des taches aillant des formes géométriques spécifiques que l'utilisateur enregistre. Ainsi le système peut lancer une action lorsqu'il détecte la présence d'un des « BLOB » enregistrés.
- Reconnaissance des gestures simples (Exemple : *TapGesture*, *SwipeGesture*, …).
- Reconnaissance des gesture complexes, c'est-à-dire les formes géométriques tracées par l'utilisateur sur la surface.
- Programmation de nouvelles gestures complexes, en enregistrant les templates équivalents et les évènements qui seront lancés.
- Administration des applications installées.

- Démarrage des applications installées.
- Multi-utilisateurs.

2- Besoins non fonctionnels

Les besoins non fonctionnels incluent :

- La facilité d'interagir avec le système en présentant une interface conviviale et ergonomique.
- L'interface doit être réactive, autrement dit le temps de réponse devrait être optimisé.
- Le logiciel et le matériel doivent être compatible afin d'assurer un bon fonctionnement.
- Le système doit être évolutif, que ce soit d'un point de vue logiciel ou matériel.

3- Spécifications de la Table Multitouch :

Le diagramme ci-dessous décrit les fonctionnalités du logiciel de la table multitouch :

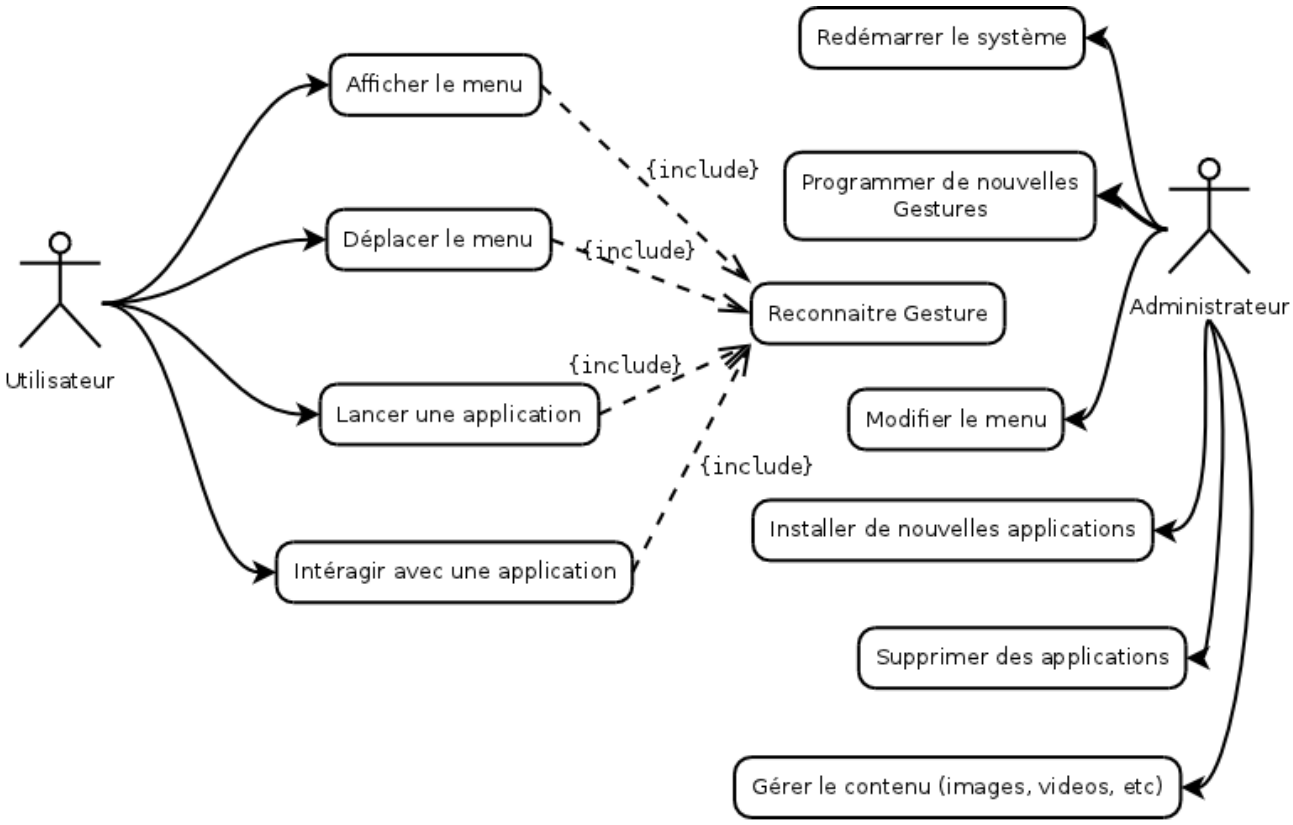

Illustration 15: *Diagramme de cas d'utilisation du logiciel de la table multitouch*

Ci-dessous le diagramme de séquence de la table multitouch :

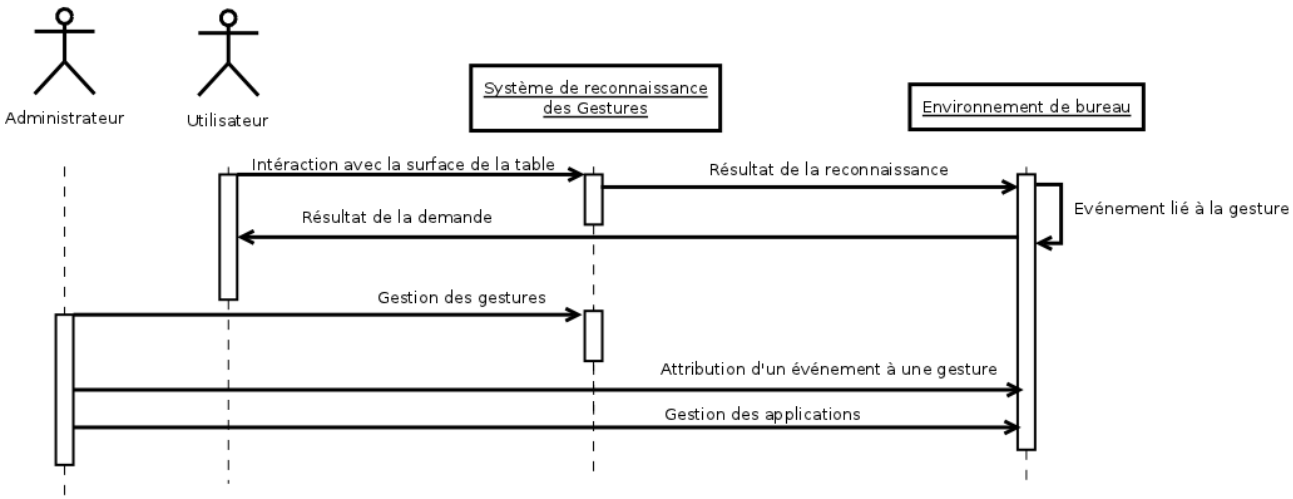

Illustration 16: *Diagramme de séquence de la table multitouch*

Chapitre 3

Étude préalable de la pile logicielle de la table multitouch

Dans ce chapitre :

1- Vue globale

2- Choix technologiques

- 2.1- Système d'exploitation
- 2.2- Langage de programmation
- 2.3- Frameworks

3- Protocole de communication TUIO

4- Le « Tracker »

Au dessus de la couche matérielle, la couche logicielle devrait être développée. Cette couche se compose d'un système d'exploitation, du protocole de communication et de l'environnement du bureau qui est l'interface visible à l'utilisateur. Pour cela plusieurs choix technologiques s'offrent.

1- Vue globale du système

Le système est composé de plusieurs couches qui communiquent entre elles. Ces couches permettent à l'utilisateur d'interagir d'une façon indirecte avec la couche la plus basse qui est la couche matérielle. L'illustration ci-dessous décrit cette architecture.

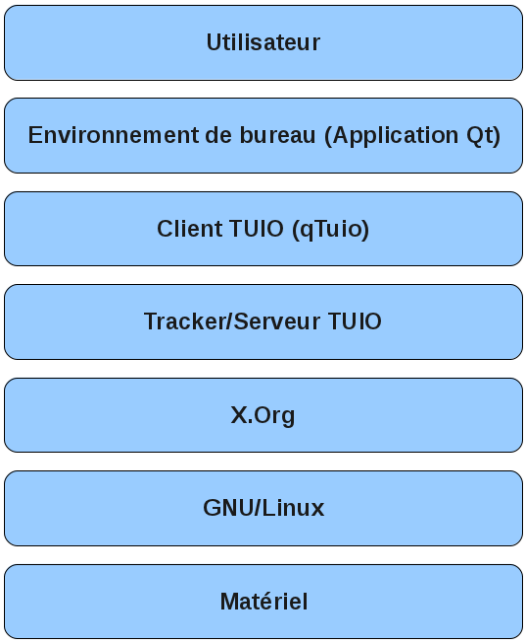

Illustration 17: *Architecture du système*

La partie comprise entre le matériel et l'utilisateur représente l'ensemble des logiciels utilisés et développés pour assurer le fonctionnement du matériel et garantir une interface à l'utilisateur.

2- Choix technologiques

2.1- Système d'exploitation

Le système de base choisi pour la création du logiciel pour la table multitouch est le système **GNU/Linux**. Ce choix se base sur le fait que **GNU/Linux** est libre, sous la licence **GPL** (GNU

Public License), ceci permet un accès plus facile au code et autorise toutes modifications et améliorations de l'existant.

Contrairement à d'autres systèmes fermés, **GNU/Linux** dispose d'une communauté très dynamique. Ce système d'exploitation est assez connu pour sa stabilité, sa fiabilité et sa robustesse. La table multitouch peut être amenée à fonctionner pendant de longues heures sans interruption, c'est pour cela qu'elle nécessite un tel système d'exploitation.

*Remarque : **GNU/Linux** n'est pas **Unix**. Le noyau **Linux** est de type **Unix** et donc inspiré de ce dernier dans son fonctionnement, mais **Unix** est tout un système d'exploitation et **Linux** est seulement un noyau. **GNU/Linux** est un système d'exploitation et **GNU** qui a été créé par **Richard M. Stallman**[18] est un acronyme récursif pour dire « GNU's Not Unix »*[19].

Le système d'exploitation **GNU/Linux** se compose essentiellement de deux grandes parties :

- Le noyaux du système d'exploitation qui est **Linux** (développé par **Linus Torvalds** en 1991). C'est la partie qui communique avec le matériel, et qui fournit les interfaces nécessaires aux applications pour accéder à la couche matérielle.
- Les outils qui entourent le noyau, c'est à dire **GNU**. Ces outils incluent la ligne de commande **Bash**, le compilateur **GCC**, le debugguer **GDB**, etc.

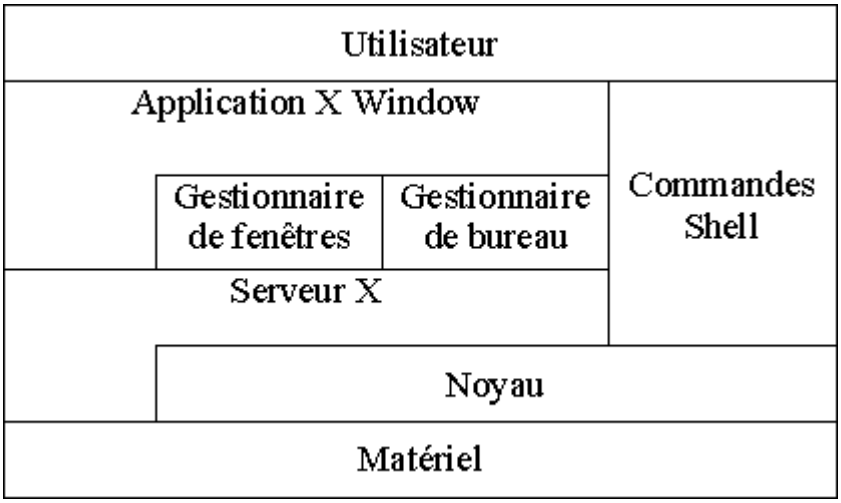

***Illustration 18:** Architecture du système GNU/Linux*[20]

2.2- Langage de programmation

Le langage de programmation choisi pour le développement de l'environnement de bureau est le langage **C++**. Ce langage est connu comme étant le meilleur outil de programmation système.

Comparé au langage de programmation **Java**, le **C++** a l'avantage de permettre aux développeurs de contrôler la mémoire avec les instructions *new* et *delete,* ainsi qu'un passage de paramètres par adresse ou par référence ce qui peut augmenter la rapidité de l'application. Sans oublier bien sur le multi héritage, les templates, et surtout le fait que le **C++** est plus rapide en exécution que le **Java**, vu qu'il n'a pas besoin d'une machine virtuelle pour s'exécuter. Il est toutefois nécessaire de recompiler le code source pour chaque plate forme, contrairement au **Java** qui n'exige qu'une seule compilation pour fonctionner sur n'importe quelle plate-forme.

Pour ce projet le langage de programmation **Python** aurait pu être utilisé, sauf que **Python** peut présenter certains problèmes de compatibilité avec l'ensemble choisi (indisponibilité d'un support à jour).

2.3- Frameworks

A coté du langage de programmation **C++**, le framework **Qt** a été utilisé. Ce framework est complètement orienté objet et offre plusieurs modules permettant un développement rapide et multi-plateformes.

Qt a une structure modulaire :

- **QtCore** : le noyau de **Qt**.
- **QtGui** : contenant toutes les classes pour créer une interface graphique, ainsi que les classes pour gérer le multitouch et la gesture.
- **QtXml** : permet la gestion des fichiers **XML**, en fournissant des classes pour le parsing et pour la création.
- **QtOpenGL** : pour interagir avec le moteur graphique **OpenGL**.
- **QtNetwork** : pour tout ce qui a rapport avec le réseau.

Il existe d'autres modules tels que **QtMultimedia**, **QtOpenVG**, **QtScript**, etc.

Le choix de **C++** et du **Qt** ne se limite pas seulement au fait qu'ils soient multi-plateformes et qu'ils présentent une panoplie d'outils et de classes facilitant le développement, mais aussi une possibilité dans le future d'intégrer d'autres frameworks tels que :

- **Ogre3D** qui est un moteur de jeu, et facilite le développement de jeux vidéo.
- **Bullet Physics** pour les interactions physiques, c'est à dire une possibilité de prendre en considération la vitesse, l'accélération, le choc entre deux ou plusieurs éléments visuels, etc.

3- Protocole de communication TUIO

TUIO (Tangible User Interface Objects) est le protocole de communication utilisé pour assurer la communication entre la table multitouch et l'environnement de bureau. Ce protocole est encodé suivant le format **OSC** (Open Sound Control) et permet une communication efficiente, en fournissant les données d'une façon claire et rapide. L'envoie des données se fait, généralement en **TUIO/UDP** sur le port 3333, mais il est possible de le faire en **TUIO/TCP** ou en **TUIO/FLC** (Flash Local Connection) dans le cas d'utilisation d'applications **Adobe Flash**.

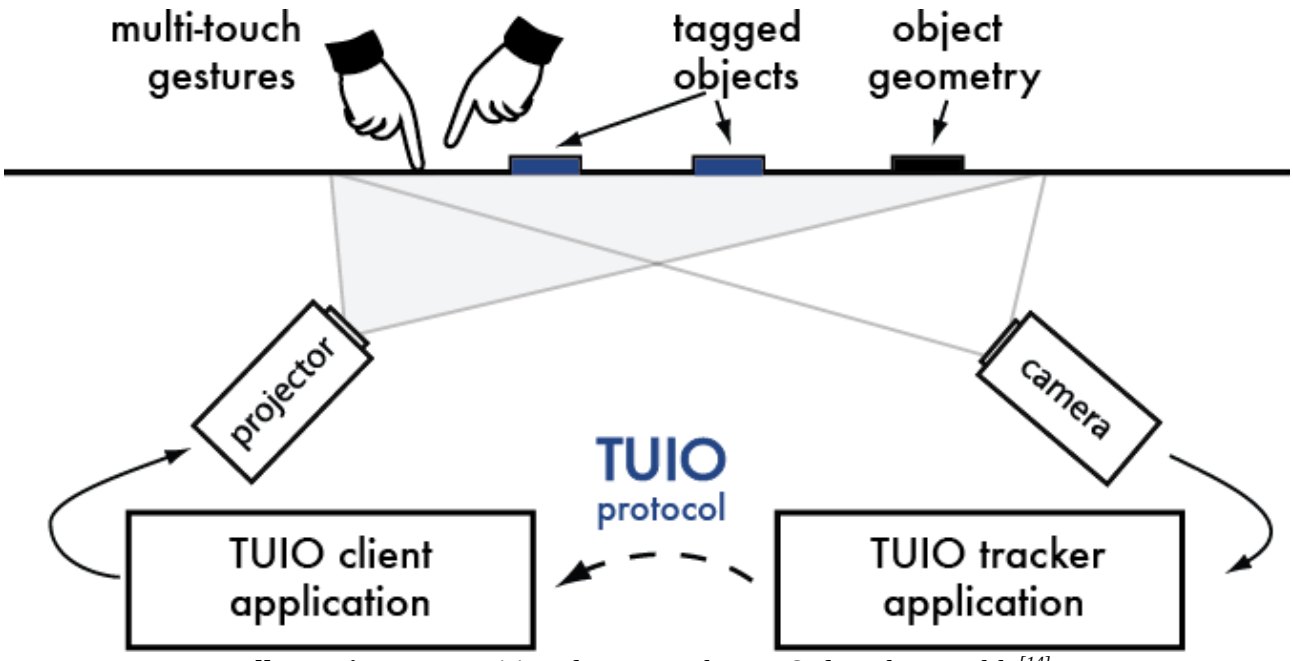

Illustration 19: *Position du protocole TUIO dans l'ensemble*[14]

L'implémentation du protocole se fait à travers deux classes principales de messages :

- SET : permet de communiquer les informations sur l'état de l'objet, c'est à dire sa position, son orientation, etc.
- ALIVE : indique l'ensemble des objets présents sur la surface. Chaque objet possède un identifiant unique.

D'autres types de messages existent aussi, tels que SOURCE pour identifier la source et son adresse, et FSEQ qui permet d'associer un identifiant pour un ensemble de messages SET et ALIVE.

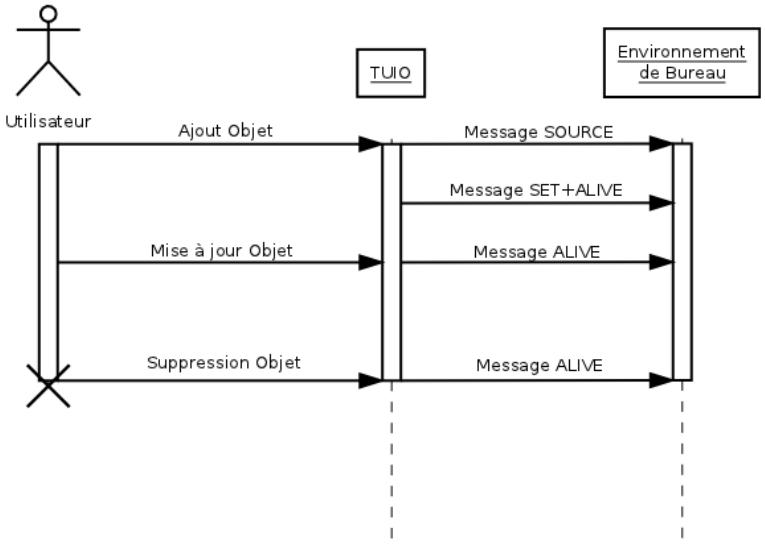

Illustration 20: *Diagramme de séquence des messages TUIO*

Les messages SET dépendent du milieu d'interaction. Sur une surface 2-dimensions, la forme d'un message SET :

- d'un « TouchPoint » :

```
/tuio/2Dcur set session_id position_x position_y vitesse_X vitesse_Y acceleration
```

- D'un objet :

```
/tuio/2Dobj set session_id class_id position_x position_y angle vitesse_X
vitesse_Y vitesse_rotation acceleration_mouvement acceleration_rotation
```

Dans l'espace (dans le cas de l'utilisation d'une caméra 3D telle que le Kinect), les messages SET changent un peu en rajoutant d'autres paramètres liés à la troisième dimension. Le message devient pour un « TouchPoint » :

```
/tuio/3Dcur set session_id position_x position_y position_z vitesse_x vitesse_y
vitesse_z accleration
```

Outre les « TouchPoint » et les objets qui sont en réalité des « Fiducials », il est possible de définir d'autres types d'objets appelés des « BLOB ».

Les « BLOB » sont des formes aperçues par la caméra et que l'utilisateur enregistre afin de pouvoir programmer des actions spécifiques. **TUIO** permet de communiquer des informations supplémentaires sur les « BLOB » par exemple l'angle de rotation, les dimensions, la vitesse de rotation, etc.

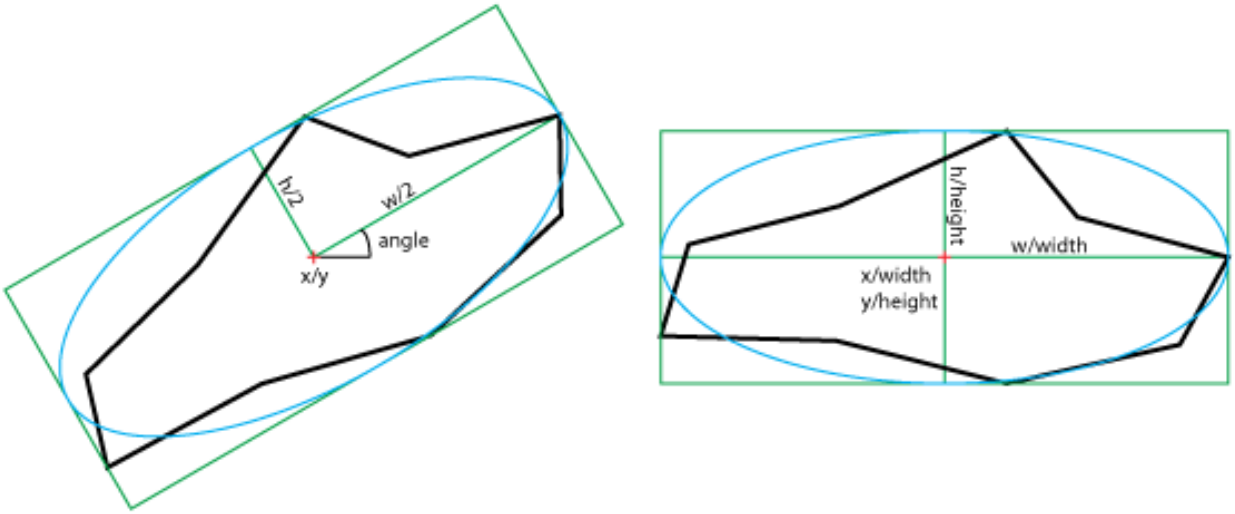

Illustration 21: *Les « BLOB »*[14]

Toutes ces informations sont envoyées à l'application qui va les traiter et prendre les bonnes décisions selon la programmation faite. L'implémentation de **TUIO** peut se faire à deux niveaux, soit une implémentation au niveau de **X.Org**, soit une implémentation indépendante.

L'implémentation bas niveau permet de faire en sorte que chaque « TouchPoint » soit interprété comme étant un pointeur a part, autrement dit une souris indépendante. Sous **GNU/Linux**, ceci est possible avec **MPX**[4] (Multi-Pointer X) qui est une modification de **X.Org**. Cette méthode a l'avantage de rendre multitouch n'importe quel environnement de bureau. Mais malheureusement elle présente énormément de problèmes de stabilité et surtout elle est limitée (impossible de reconnaître les objets, les « BLOB » et les « Fiducial »). L'apparition de la technologie **Optimus**[5] présente un obstacle d'intégration de **MPX** dans **X.Org**.

La deuxième méthode consiste à implémenter **TUIO** indépendamment du serveur graphique **X.Org**. Cette méthode est plus stable, plus efficace et permet un développement rapide en séparant le serveur **TUIO** de tout autre composant. Ceci permet de programmer plusieurs clients qui peuvent recevoir les données de ce serveur.

4 Voir Annexe C
5 Voir Annexe D

Nous avons choisi la deuxième méthode d'implémentation puisqu'elle répond plus à nos besoins et nous permet de mieux évoluer par la suite.

4- Le « Tracker »

Le « tracker » est une application qui va recevoir l'image captée par la caméra, la traiter et renvoyer des messages **TUIO**. Le « tracker » utilisé est **CCV** (Community Core Vision) qui est développé par **NUI Group**, un groupe de recherche dans le domaine de l'interaction homme-machine créé en 2006.

CCV est développé en **C++** et joue le rôle d'un analyseur d'images et d'un serveur **TUIO**. Il est possible avec **CCV** d'inter-connecter plusieurs caméras pour avoir un champ de vision plus large.

Parmi les fonctionnalités de **CCV** :

- Disponibilités de plusieurs filtres (soustraction du background, amplification, seuil, etc). Ceci permet de faire des réglages afin d'obtenir de meilleures performances.
- Possibilité de changer de caméra dans le cas où il y en plus qu'une.
- Calibrage selon les dimensions des tables.
- Réflection d'image.
- Diffusion des données **TUIO** en réseau.
- Utilisation du **GPU** (Graphics Processing Unit).
- Reconnaissance de certains « Fiducial ».
- Enregistrement d'objets sous la forme « BLOB ».

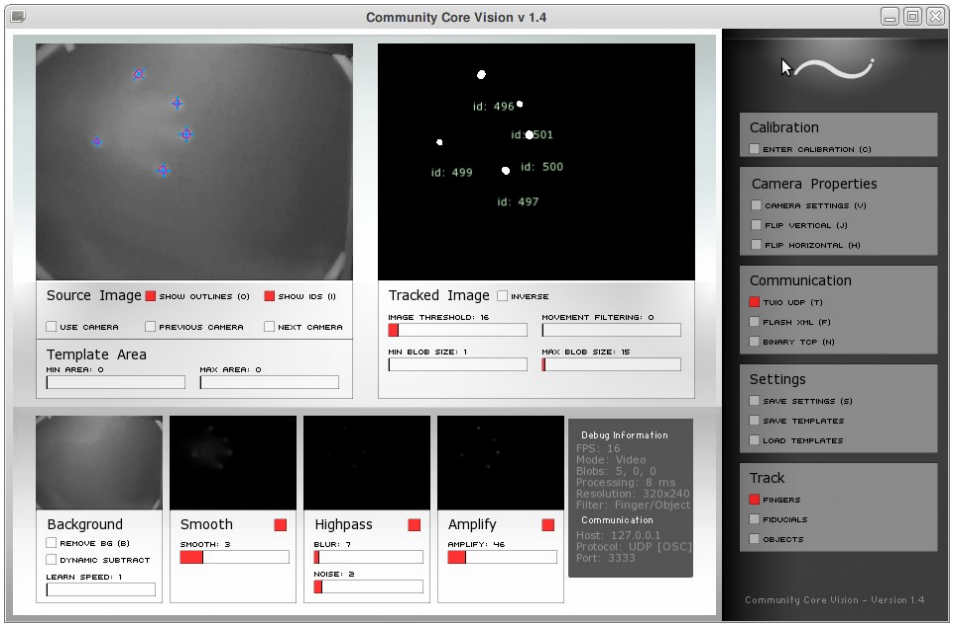

Illustration 22: *Interface de* ***CCV***

L'interface de **CCV** permet de donner un aperçu (dans le cadre supérieur gauche) de ce que la caméra capte et permet aussi de changer de caméra grâce aux boutons « previous camera » et « next camera ».

Le cadre d'à coté donne quelques informations sur les données qui seront envoyées par **TUIO** (session id de chaque objet ou « TouchPoint »). Il donne la possibilité aussi de faire quelques réglages, essentiellement sur la quantité de données minimale et maximale.

En bas il existe d'autres réglages qui permettent d'améliorer la qualité de capture et aussi d'éliminer les perturbations lumineuses (le cadre Background).

Il est aussi nécessaire d'effectuer un calibrage de **CCV** avant de l'utiliser. Ceci se fait en appuyant sur le bouton « Enter Calibration ». Pour effectuer le calibrage il faut spécifier des points de référence.

Illustration 23: *Calibrage* ***CCV***

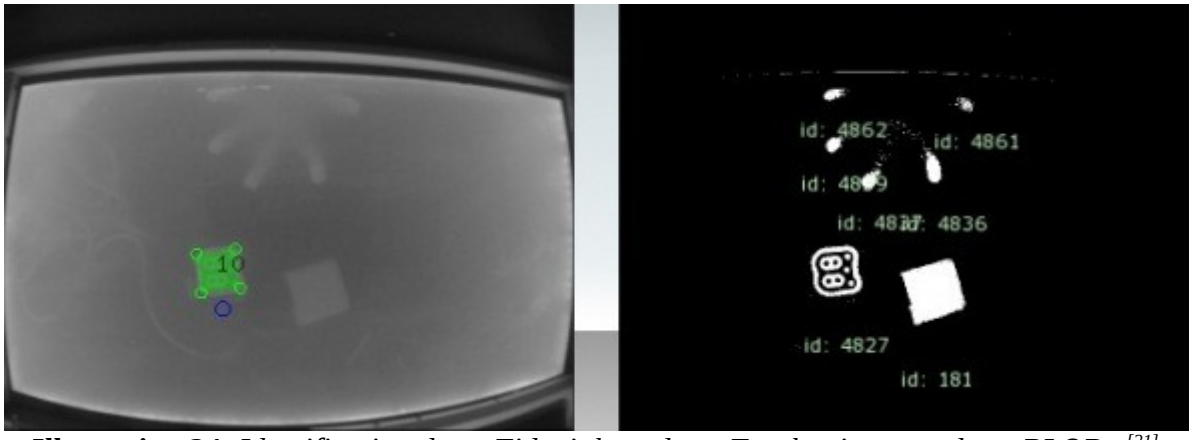

Illustration 24: *Identification des « Fiducials », des « Touchpoints » et des « BLOB »*[21]

Chapitre 4

Conception et réalisation de la pile logicielle de la table Multitouch

Dans ce chapitre :

Le développement de la plate-forme s'est fait en utilisant le framework **Qt**. Cette plate-forme est composée de deux parties. La première partie (Module Multitouch et Gestures) est d'une importance primordiale dans n'importe quelle technologie multitouch, elle traite les messages TUIO venant du « tracker » et aussi la reconnaissance des gestures.

La deuxième partie (Environnement de Bureau) représente l'interface visible à l'utilisateur. Cette interface est l'ensemble des logiciels que l'utilisateur peut lancer.

L'illustration ci-dessous montre les différentes parties du projet :

Illustration 25: *Les différentes parties du projet*

1- Module multitouch et gestures

Ce module peut être décomposé en deux grandes parties :

- La passerelle qTuio qui reçoit les messages TUIO et construit des événements multitouch.
- La reconnaissance des gestures qui analyse les événements générés par qTuio et si une gesture est reconnue, un événement spécifique est engendré pour être envoyé à l'environnement de bureau.

1.1- qTuio

Le support multitouch a été rajouté à **Qt** à la version 4.6 qui est sortie en fin 2009. Mais malheureusement **Qt** ne supporte pas le protocole **TUIO**, il ne peut communiquer qu'avec les écrans tactiles dotés d'un support natif dans le système d'exploitation. Pour cela il faut développer une passerelle qui reçoit les messages **TUIO** et qui les transforme en événements multitouch **Qt**.

qTuio a été initialement développé par un étudiant de l'Université Technique de Munich, dans le cadre de ses études pour le diplôme d'ingénieur. **qTuio** est en réalité une implémentation de **TUIO.** Il exploite trois bibliothèques importantes :

- **TUIO** : Bibliothèque de base permettant d'avoir des structures de données contenant toutes les informations sur les « Fiducials » et les « TouchPoints ».

- **OSC** : Pour les connexions réseau et l'échange des paquets UDP.
- **Qt** : Pour pouvoir créer les événements **QTouchEvent** et les envoyer à l'application **Qt**.

Malheureusement ce module avait quelques bugs dans le renvoie des états des « TouchPoints » et il ne pouvait pas envoyer les événements à tout les composants de **Qt**. Nous avons dû reprendre le code source, corriger les bugs, l'améliorer et bien sûr redistribuer les modifications que nous avons faites.

Parmi les problèmes rencontrés avec la toute première version de **qTuio,** nous évoquons :

- Un problème avec l'accès mémoire : tentative d'accéder à des objets qui n'existent pas et ceci provoque une *Erreur de Segmentation*.
- Un problème avec les états des « TouchPoints », ce qui fait que les **QTouchEvent** n'indiquent pas à l'application **Qt** si l'objet a subit une mise à jour de ses attributs ou bien il a été supprimé.
- Un problème avec l'envoie de la liste des « TouchPoints ».

Ces problèmes ont été surmontés et une nouvelle version a été publiée sur la plate-forme **github**[22]. Il est important aussi de savoir que **qTuio** fonctionne en tant qu'un « thread » au sein de l'application générale. Il récupère les messages **TUIO,** construit une liste avec les identifiants des points, leurs positions, leurs états, etc, et renvoie cette liste à l'application **Qt** sous forme de **QTouchEvent**. Et l'application devrait traiter ces événements et exécuter les opérations adéquates.

Outre les modifications effectuées, il existe plusieurs autres améliorations[6] qui peuvent être apportées au module **qTuio** pour le rendre encore plus performant.

Le diagramme de classe suivant décrit la structure de la passerelle qTuiod'une façon simplifiée (voir Annexe E pour le diagramme complet et plus détaillé) :

6 Voir Chapitre 4 : 4.2 perspectives

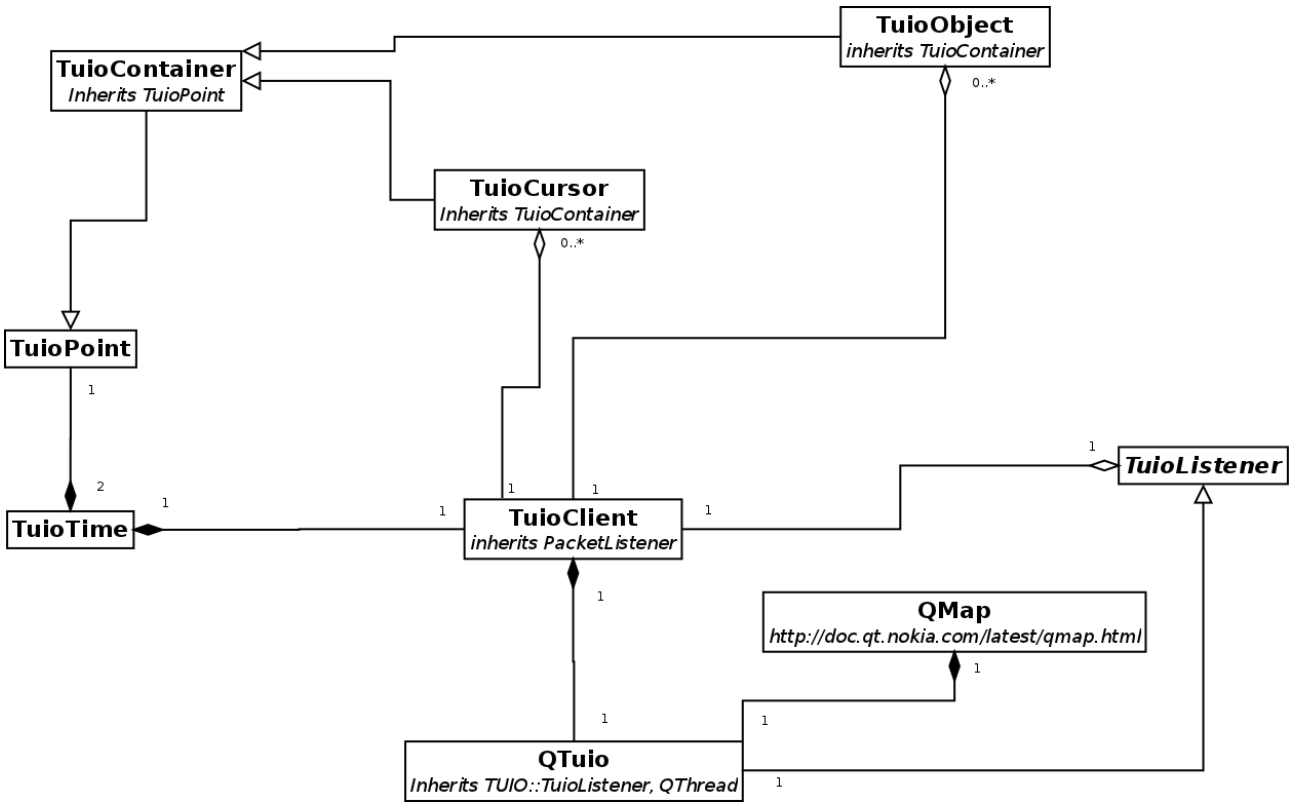

Illustration 26: *Diagramme de classe simplifié du module* ***qTuio***

qTuio est composé d'un ensemble de classes :

QTuio est la classe mère de la passerelle **qTuio**. Elle permet de préparer les événements du toucher (*QTouchEvent*) et de les envoyer à l'application à laquelle elle est liée. Cette classe hérite de deux classes :

- QThread, pour pouvoir bénéficier du threading et donc avoir une exécution parallèle.
- *TuioListener* qui est une classe abstraite décrivant les méthodes à reimplementer pour pouvoir détecter l'ajout, la mise à jour et la suppression des « TouchPoints » et des « BLOB ».

Cette classe (*QTuio*) est composée d'un *QMap* et d'un *TuioClient* :

- QMap est une structure de données contenant l'ensemble des « TouchPoints » avec leurs états.
- *TuioClient* est la classe qui reste en écoute du serveur TUIO.

TuioContainer est une classe représentant d'une façon générale les « TouchPoints » et les « BLOB ». Ainsi elle est considérée comme étant la classe mère de *TuioCursor* et de *TuioObject*. Cette classe hérite de la classe *TuioPoint* qui est une classe décrivant un point dans un repère 2-dimensions.

TuioTime est une classe qui représente le temps, ainsi il est possible de savoir quel « TouchPoint » est apparu en premier, etc.

1.2- Module gesture

La gesture est un module très important en multitouch. **Qt** propose deux classes permettant la programmation de la gesture :

- *QGestureRecognizer* : fournit une infrastructure pour la reconnaissance des gestures.
- *QGesture* : permet la représentation d'une gesture en décrivant toutes ses propriétés.

Pour programmer une nouvelle gesture il est nécessaire de créer deux classes qui héritent respectivement de *QGesture* et *QGestureRecognizer*. Il est nécessaire de reimplementer les méthodes suivantes pour pouvoir rajouter une gesture :

- *QGesture *QGestureRecognizer::create(QOBject *target)* : cette méthode est appelée pour vérifier l'existence d'une instance de la classe QGesture qui va contenir les propriétés de la reconnaissance de la gesture. Par exemple pour un *TapGesture*, la position du doigt est considérée comme étant une propriétés.
- *QGestureRecognizer::Result QGestureRecognizer::recognize(QGesture *state, QObject *watched, QEvent *event)* : Les étapes de la reconnaissance d'une gesture, c'est à dire le suivi de l'évolution des « TouchPoints » et l'analyse permettant de savoir s'il s'agit d'une gesture ou pas, se fait avec la reimplementation de cette méthode. Le résultat retourné est l'état de la gesture (FinishGesture si la reconnaissance est faite, CancelGesture si la façon avec laquelle les « TouchPoints » évoluent ne correspond pas à cette gesture, etc).
- *void QGestureRecognizer::reset(QGesture *state)* : cette méthode est utilisée pour réinitialiser les propriétés du QGesture. Ainsi aucune confusion ne peut avoir lieu si deux mêmes gestures se succèdent.

Le module de la gesture que nous avons développé permet la reconnaissance des gestures simples et des gestures complexes (les formes géométriques). Chaque *QGestureRecognizer* peut passer par plusieurs états selon le degré de reconnaissance d'une gesture :

- QGestureRecognizer::Ignore : Cet état indique qu'il faut ignorer cette gesture. Il est utilisé généralement comme état par défaut surtout quand aucun événement de TouchBegin, ni de TouchUpdate, ni de TouchEnd n'est provoqué.
- QGestureRecognizer::MaybeGesture : Cet état est utilisé lorsqu'il manque des données pour confirmer ou annuler une gesture.
- QGestureRecognizer::TriggerGesture : Cet état indique que la gesture a été déclenchée et peut poursuivre (pas encore terminée).
- QGestureRecognizer::FinishGesture : Ceci indique que la gesture a été reconnue et qu'elle s'est terminée.
- QGestureRecognizer::CancelGesture : Cet état indique qu'il ne s'agit pas de la bonne gesture.
- QGestureRecgonizer::ConsumeEventHint : Cet état permet de consommer l'événement sans rien faire. La différence avec QGestureRecognizer::Ignore est que cet état empêche la propagation de l'événement aux autres composants de l'application.

1.2.1- Gestures Simples

Les gestures simples sont les gestures qu'on trouve partout dans les périphériques multitouch, et qui sont le *TapGesture, TapAndHoldGesture, PanGesture, SwipeGesture et PinchGesture*.

TapGesture :

Le *TapGesture* (représenté dans l'illustration 10 au Chapitre 1) consiste à toucher la surface avec un seul doigt et de libérer ce « TouchPoint » sans dépasser la période fixée (en moyenne 400 millisecondes) et sans le déplacer de sa position d'origine plus loin que la distance fixée (environ 20 pixels).

Cette gesture peut être considérée comme étant un clique d'une souris. Elle est utilisée généralement pour appuyer sur un bouton ou pour lancer une action. Le diagramme de classes suivant décrit le *TapGesture* :

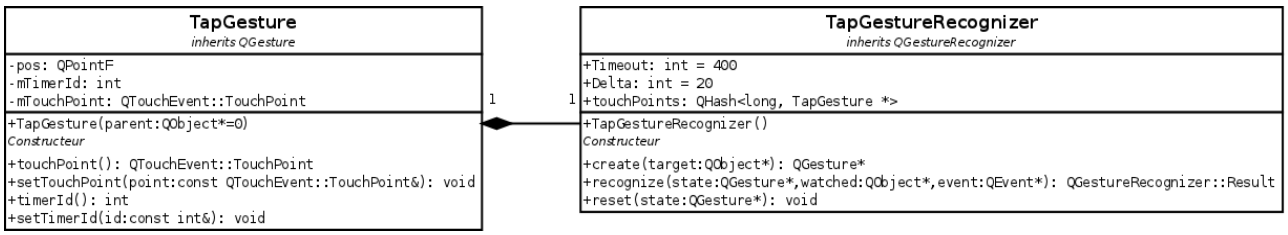

Illustration 27: *Diagramme de classe du TapGesture*

La reconnaissance du *TapGesture* passe par plusieurs étapes le diagramme de flux de données suivant explique la procédure :

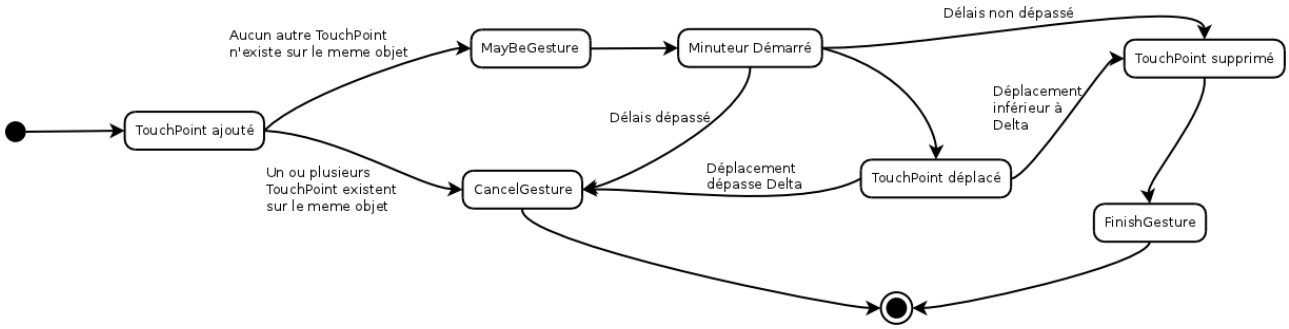

Illustration 28: *Diagramme de flux de données décrivant la reconnaissance du TapGesture*

Trois actions peuvent annuler le *TapGesture*, la toute première vérification étant de s'assurer qu'il n'y a qu'un seul « TouchPoint » sur l'objet (une image, un bouton, etc).

La deuxième action est de déplacer le « TouchPoint », le *TapGesture* exige un maintient de la position mais un facteur de tolérance est nécessaire, ce facteur est fixé à 20 pixels et se base sur des tests que nous avons faits en prenant plusieurs personnes et en calculant la moyenne.

La troisième action est de dépasser le « Timeout » fixé. Un clique d'une souris ne peut être considéré comme étant un clique que si la durée entre l'appuie sur le bouton et le relâchement est courte.

Si une de ces trois actions a lieu, l'état du *TapGesture* change en CancelGesture et la fonction *void TapGestureRecognizer::reset(QGesture *)* est appelée pour remettre à zéro les propriétés de la classe *TapGesture*.

TapAndHoldGesture :

Le *TapAndHoldGesture* (illustration 10, Chapitre 1) est une gesture qui ressemble au *TapGesture* dans son fonctionnement, la seule différence est le fait que l'utilisateur ne doit pas relâcher le « TouchPoint » ; en quelque sorte c'est un clique maintenu de la souris.

Les propriétés de *TapGesture* sont gardées. La modification intervient essentiellement le minuteur au lieu d'annuler la gesture si le « Timeout » est dépassé, le système signale plutôt que la gesture a été reconnue.

Le diagramme de classe suivant décrit la structure de cette gesture :

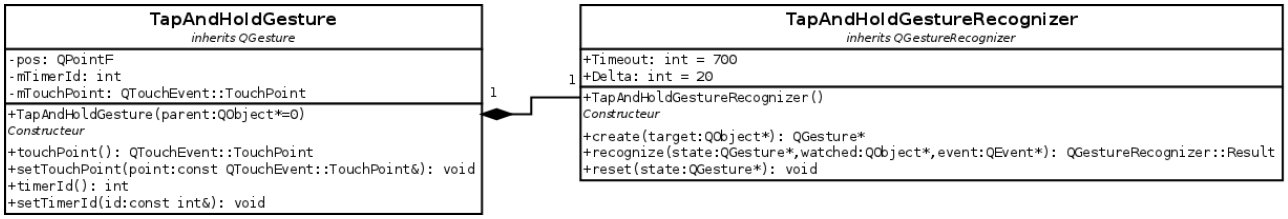

Illustration 29: *Diagramme de classe de la gesture TapAndHold*

La propriétés « Timeout » de la classe *TapAndHoldGestureRecognizer* indique le temps du maintient du « TouchPoint » à sa position pour que la gesture soit interprétée comme étant un *TapAndHoldGesture*.

Ci dessous le diagramme de flux de données :

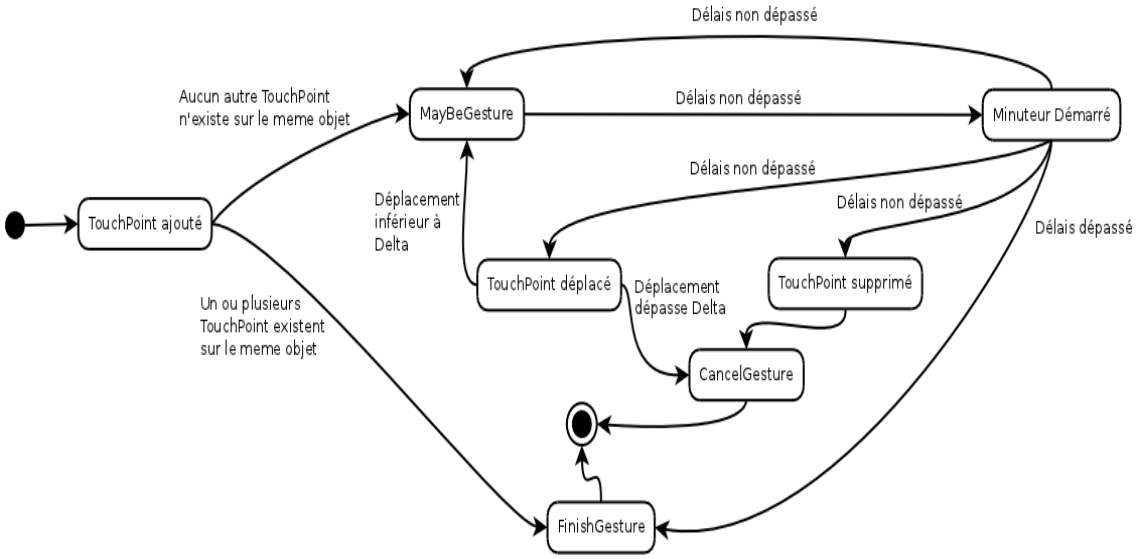

Illustration 30: *Diagramme de flux de données décrivant la reconnaissance du TapAndHoldGesture*

Les actions pouvant annuler un *TapAndHoldGesture* sont essentiellement la suppression du « TouchPoint » avant le « Timeout » ou bien le déplacement supérieur à 20 pixels de ce « TouchPoint ».

PanGesture :

Le *PanGesture* (illustration 14, Chapitre 1) est une gesture qui nécessite un seul « TouchPoint » et consiste à bouger le point pour produire un déplacement. Cette gesture est utilisée pour pouvoir déplacer des éléments qui s'affichent (exemple des images). Donc l'élément suivra la position du « TouchPoint » qui est dessus.

Le *PanGesture* est parmi les gestures les plus simple à programmer, vu qu'il n'y a pas de propriétés spécifiques à rajouter et il n'y pas un traitement spécial à faire. Lorsqu'un « TouchPoint » est sur un objet et que ce « TouchPoint » bouge, l'état du *PanGestureRecognizer* change en TriggerGesture jusqu'à ce que le « TouchPoint » est supprimé, alors l'état passera en FinishGesture.

Le diagramme de classe suivant décrit le *PanGesture* :

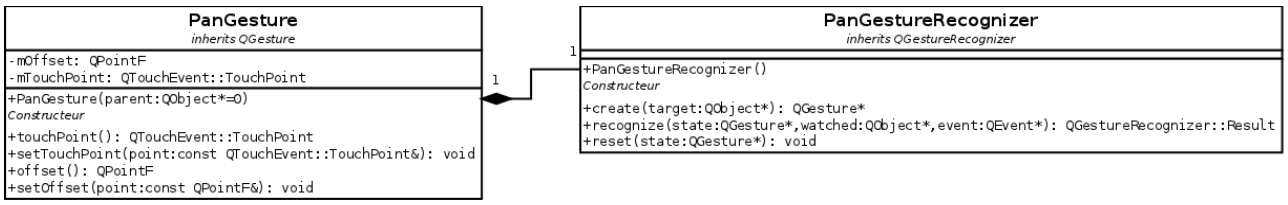

Illustration 31: *Diagramme de classe de la gesture PanGesture*

Ci dessous le diagramme de flux de données qui explique la procédure de reconnaissance de cette gesture :

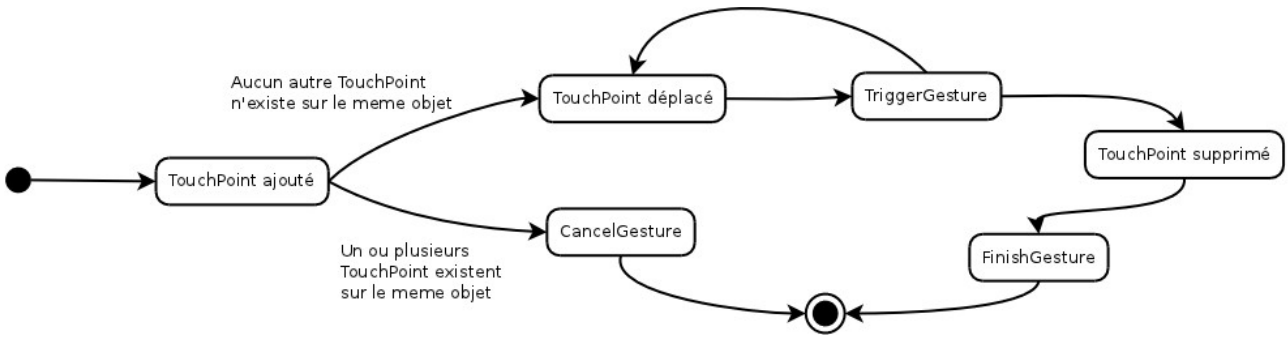

Illustration 32: *Diagramme de flux de données décrivant la reconnaissance du PanGesture*

SwipeGesture :

Le *SwipeGesture* (illustration 13, Chapitre 1) est une gesture semblable au *PanGesture*, sauf qu'elle tient compte de la vitesse et de l'accélération. Cette gesture est très utilisée pour naviguer dans un album photos ou pour passer d'une page à une autre ou même pour envoyer un élément d'un endroit sur la surface à un autre endroit. L'utilisateur avec son doigt dessine la trajectoire que doit emprunter l'élément et selon la vitesse, l'élément va continuer à se déplacer en décélérant jusqu'à s'arrêter.

La programmation de cette gesture doit prendre en considération le point de départ et le point d'arrivée et de ces deux paramètres le système peut reconnaître la direction et le sens du déplacement. Il faut aussi calculer le temps écoulé entre le TouchBegin et le TouchEnd, autrement dit le temps écoulé entre l'ajout du « TouchPoint » et la suppression du « TouchPoint », avec ce paramètre il sera possible de calculer la vitesse et donc de déterminer le déplacement que va faire l'élément.

Le schéma ci dessous montre la position finale d'un élément après le *SwipeGesture*. Contrairement au *PanGesture* l'élément continue à se déplacer dans la même direction en diminuant de vitesse.

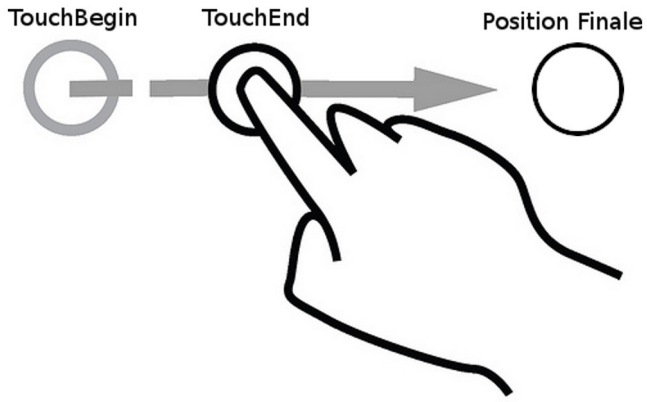

Illustration 33: *Exemple du SwipeGesture. La position finale de l'élément dépend de la vitesse du déplacement entre le TouchBegin et le TouchEnd[*

Il est impossible de différencier le *PanGesture* et le *SwipeGesture* vu que les deux gestures ont un fonctionnement identique. Pour cela il est préférable qu'un élément ne capte qu'une seule des deux gestures.

PinchGesture :

Le *PinchGesture* (illustration 12, Chapitre 1) est une gesture assez connue et très utilisée pour faire un agrandissement ou une réduction de la taille d'une image. Cette gesture a besoin de deux « TouchPoints » qui se déplacent sur une même direction mais dans deux sens opposés. Ainsi la distance entre les deux points sera utilisée pour calculer le facteur de mise à l'échelle.

Le facteur de mise à l'échelle est calculé à partir de la position de départ et la position d'arrivée selon la formule suivante :

facteur = distance (Point1, Point2) / distance (Point1_départ, Point2_départ)

Le diagramme de classe suivant explique la structure de cette gesture :

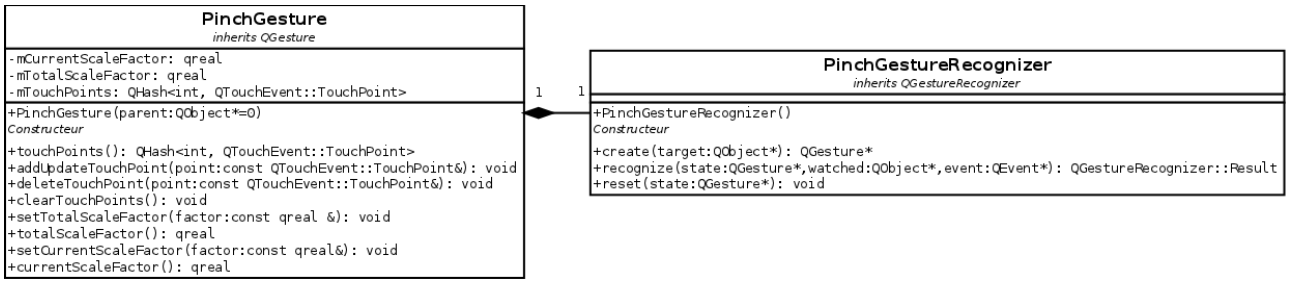

Illustration 34: *Diagramme de classe de la gesture PinchGesture*

Le diagramme de flux de données ci dessous décrit le fonctionnement de la reconnaissance de cette gesture :

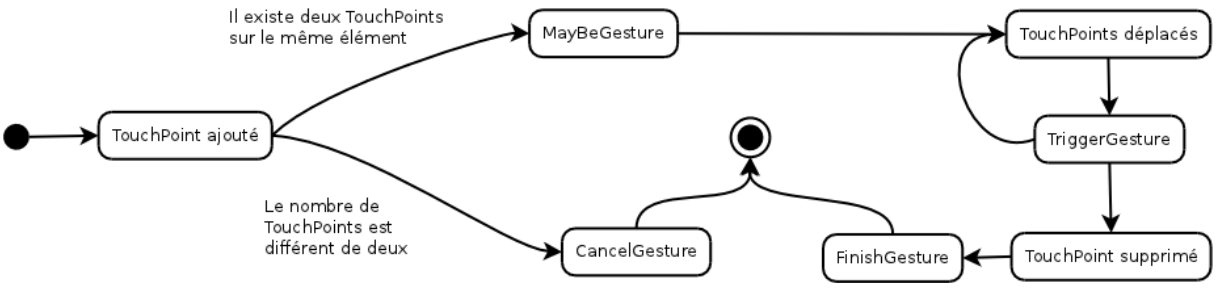

Illustration 35: *Diagramme de flux de données décrivant le fonctionnement du PinchGesture*

1.2.2- Les gestures complexes

Pour permettre une meilleure flexibilité du module gesture, nous avons préféré opter pour une solution évolutive. Pour cette partie nous nous sommes basés sur une recherche menée par :

- Dr. Jacob O. Wobbrock[7] de l'Université de Washington.
- Andrew D. Wilson[8] du département de recherches de Microsoft à Redmond.
- Dr. Yang Li[9] de l'Université de Washington.

Ces trois chercheurs ont travaillé sur l'interaction homme-machine et ont élaboré un algorithme en 2007 permettant de reconnaître les formes géométriques dessinées sur une surface multitouch, l'algorithme s'appelle « 1$ Gesture »[24].

Nous avons implémenté cet algorithme en l'adaptant à nos besoins et à notre plate-forme. Le fonctionnement se base sur une comparaison des formes, pour cela il faut définir des modèles. Le système va devoir comparer entre ce que l'utilisateur a dessiné et la base de données des modèles et essayer de reconnaître la forme.

L'existence de modèles permet une évolutivité souple, c'est-à-dire que l'administrateur du système peut définir ses propres formes géométriques et c'est ce qui permet de rendre le système plus intelligent, puis qu'il peut apprendre de l'utilisateur.

Nous avons relevé certains problèmes avec les gestures complexes. Si par exemple le système doit reconnaître un triangle, il ne peut pas savoir à l'avance le type du triangle (équilatéral, isocèle, rectangle ou quelconque), il ne peut pas non plus connaître les dimensions de ce triangle ni son orientation. C'est ce qui fait qu'il faut trouver une solution générique permettant la reconnaissance d'une forme géométrique quelque soit la façon avec laquelle l'utilisateur la dessine.

Un autre problème est le nombre de points enregistrés lorsque l'utilisateur dessine une forme géométrique. Ce nombre dépend de la vitesse avec laquelle l'utilisateur dessine. Plus il dessine vite, moins il y a de points enregistrés.

Les modèles (ou templates) sont des fichiers textes contenant les coordonnées de l'ensemble des points d'une forme géométrique. Ci-dessous un extrait du fichier template qui décrit un cercle :

7 Page personnelle du Dr. Wobbrock : http://faculty.washington.edu/wobbrock/
8 Page personnelle de Andrew D. Wilson : http://research.microsoft.com/en-us/um/people/awilson/
9 Site Web du Dr. Li : http://yangl.org/

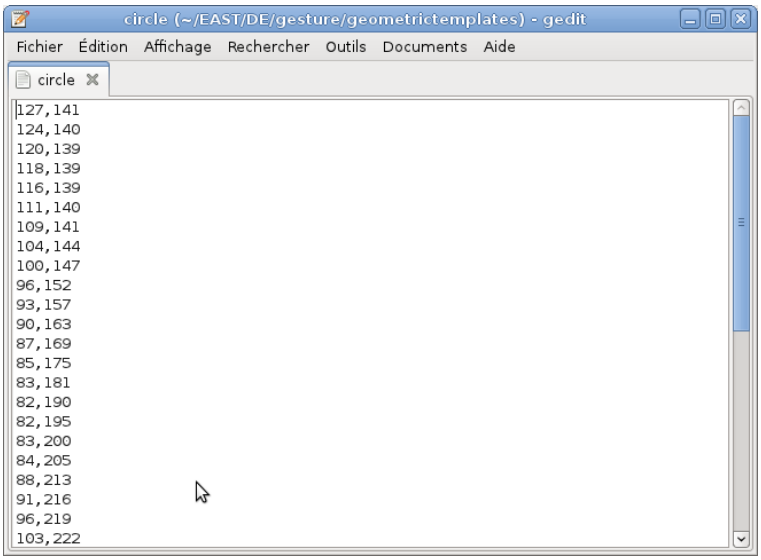

Illustration 36: *Coordonnées des points décrivant un cercle*

La reconnaissance d'une forme nécessite un passage par quatre étapes qui permettent de préparer l'ensemble des points enregistrés (ou la forme dessinée) afin de pouvoir les comparer avec les modèles dans la base de données et calculer le pourcentage d'exactitude et renvoyer l'identifiant ou le nom de la forme géométrique.

Première étape : Échantillonnage

Elle permet d'avoir un nombre fixe de points et ceci afin de pouvoir opérer une comparaison correcte avec les templates. D'après des tests effectués il faut un nombre de points compris entre 32 et 256. L'illustration ci dessous montre le nombre de points enregistrés selon la vitesse avec laquelle l'utilisateur dessine :

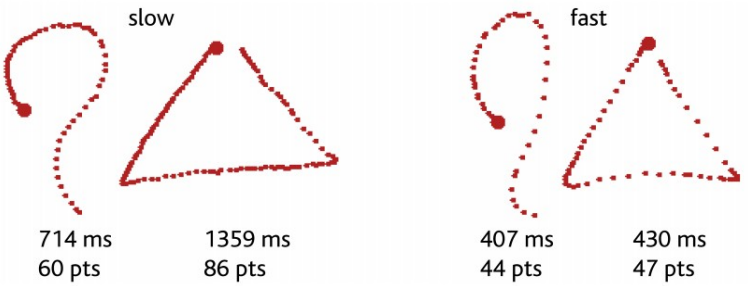

Illustration 37: *Nombre de points enregistrés selon la vitesse* [24]

Pour faire l'échantillonnage, il faut calculer en premier lieu la longueur du chemin et diviser cette longueur par N-1 (N étant le nombre de points qu'il faut obtenir à la fin de l'échantillonnage), ceci donnera la distance « d » qu'il faut avoir entre chaque deux points successifs.

Ensuite en partant du point de départ, il faut calculer la distance entre celui ci et le point suivant, si cette distance est supérieur à « d », un nouveau point sera rajouté à une distance égale à « d » du point de départ. L'itération suivante calculera la distance entre le point rajouté et le point qui se trouve tout juste après, et répétera la même opération. Si la distance est inférieur à « d » le point est supprimé.

Ainsi il est possible d'avoir un ensemble de points qu'on peut utiliser pour la comparaison avec les modèles. Ci-dessous un échantillonnage d'une étoile avec N= 32, N= 64 et N= 128. L'étoile rouge est le dessin fait par l'utilisateur :

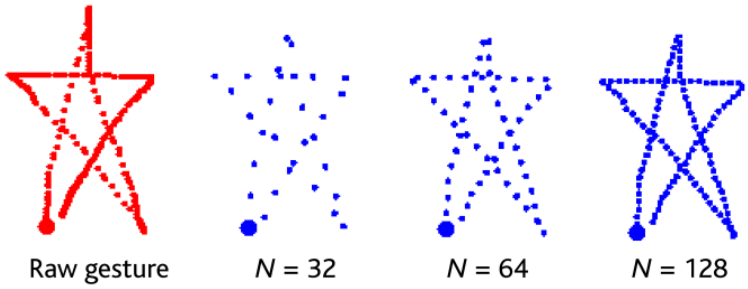

Illustration 38: *Échantillonnage*[24]

Deuxième étape : Rotation

La deuxième étape consiste à faire une rotation du dessin afin de pouvoir le comparer avec les modèles. Ainsi l'utilisateur peut dessiner n'importe quelle forme de n'importe quelle façon. Par exemple, pour un triangle isocèle, l'utilisateur peut le dessiner avec le sommet en haut ou en bas et donc il faut trouver un angle de rotation pour l'obtention d'un triangle unique.

Pour cela il existe plusieurs méthodes qui peuvent être gourmandes en ressources. Parmi ces méthode, il y a la possibilité de faire une rotation itérative de +1° jusqu'à 360° et de comparer avec les modèles. Cette méthode n'est pas efficace et est très lente.

L'idée est de calculer l'angle entre la droite horizontale qui passe par le centre de l'ensemble des points captés et la droite qui passe par le point de départ et ce centre. Ensuite il faut faire une rotation du dessin pour que cet angle soit égal à zéro.

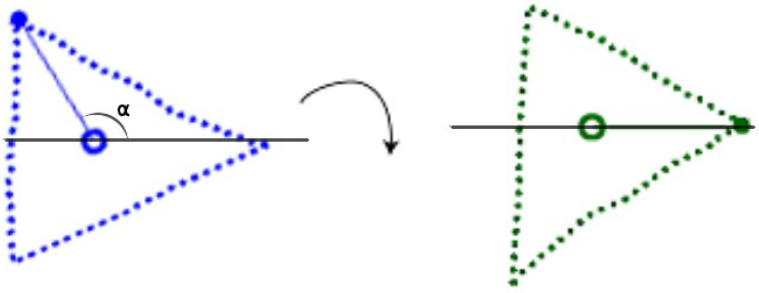

Illustration 39: *Rotation de l'ensemble des points*[24]

Troisième étape : Mise à l'échelle et translation

Cette étape permet de redimensionner le dessin à une taille fixée dans le code, ainsi la forme dessinée et le modèle auront les mêmes dimensions (le redimensionnement ne garde pas les proportions, il est plus rapide et plus facile d'utiliser une dimension carrée).

Une fois la mise à l'échelle effectuée, il faut faire une translation du centre de l'ensemble des points pour ramener ses coordonnées à l'origine du repère.

Quatrième étape : Comparaison avec les templates

Une fois le dessin de l'utilisateur est passé par les trois étapes précédentes le système va comparer l'ensemble des points avec l'ensemble des points des templates.

Avant de comparer avec un template il faut tout d'abord le faire passer par les trois étapes précédentes.

Le calcul du pourcentage d'exactitude se base sur une formule mathématique simple dérivée de la formule de calcul de la distance[10] entre deux points :

$$d_i = \frac{\sum_{k=1}^{N} \sqrt{(C[k]_x - T_i[k]_x)^2 + (C[k]_y - T_i[k]_y)^2}}{N}$$

- *k est l'indice du point de coordonnées (x, y).*
- *N est le nombre total des points.*
- *C est l'ensemble des points dessinés par l'utilisateur après le passage par les trois étapes précédentes.*
- *T_i est le template numéro i.*

10 $AB = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$

Il suffit de retenir le numéro du template avec la distance la plus petite. De cette façon le système reconnaîtra la forme dessinée.

Ainsi l'utilisateur peut enregistrer ses propres templates en les dessinant directement sur la surface multitouch et programmer certaines actions selon la forme géométrique.

Le diagramme de classe suivant décrit la structure des classes permettant la reconnaissance des gestures géométriques :

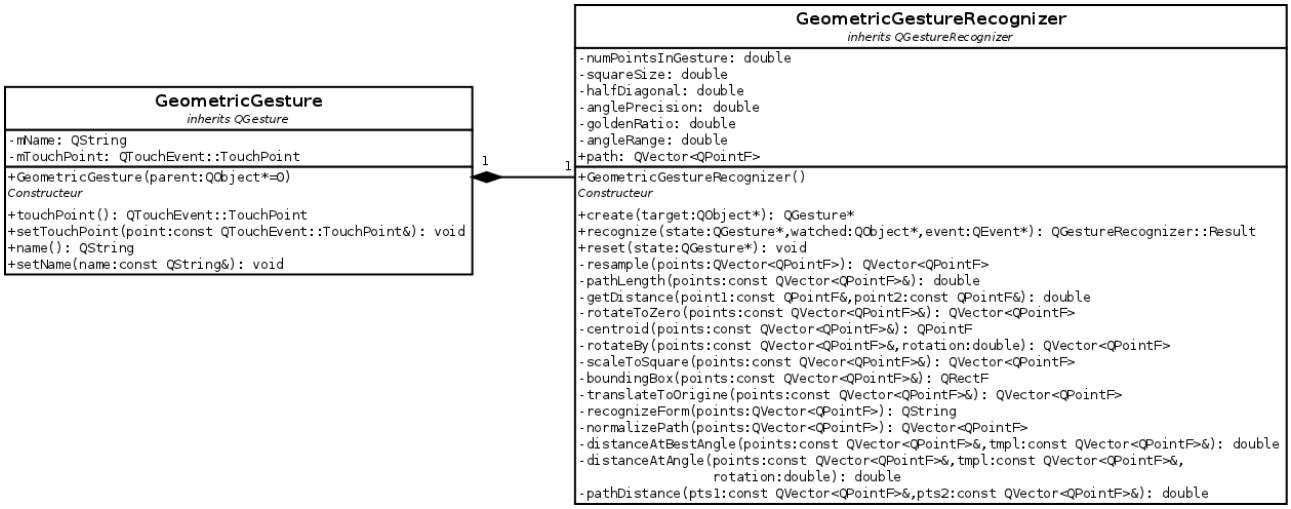

Illustration 40: *Diagramme de classe de la structure de la GeometricGesture*

Le diagramme de flux de données suivant explique le fonctionnement de la classe *GeometricGestureRecognizer* :

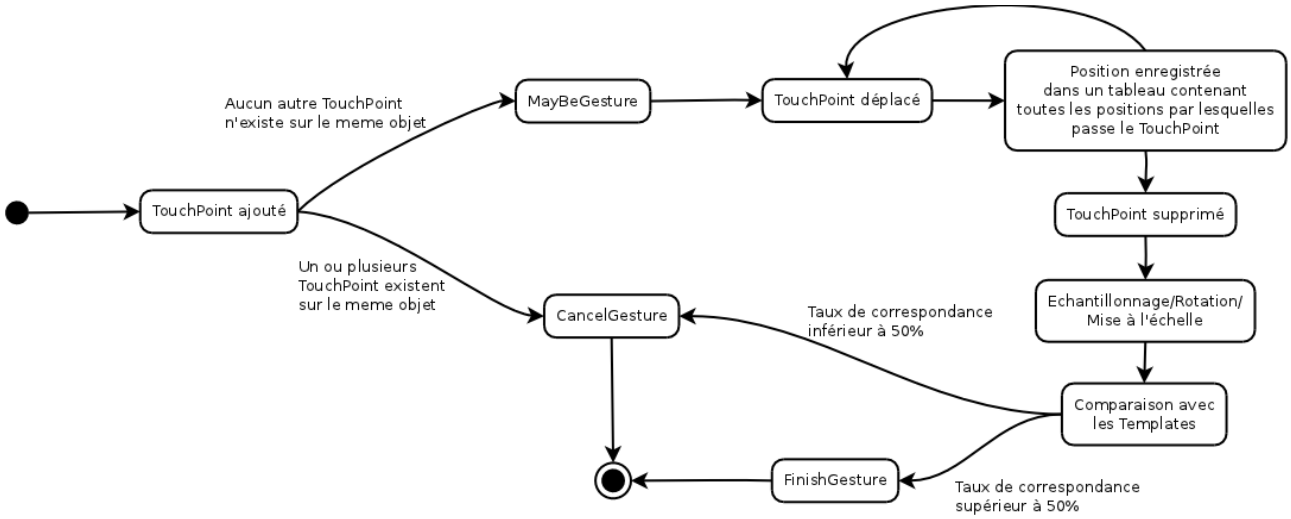

Illustration 41: *Diagramme de flux de données de la reconnaissance des formes géométriques*

2- L'environnement de bureau

L'environnement de bureau est le dernier composant de l'ensemble du système et c'est ce qui est visible à l'utilisateur. Cet environnement est constitué d'une interface permettant l'exécution d'applications par le toucher.

Chaque utilisateur, suite à une gesture spécifique, peut avoir le menu proposant les différentes applications. La gesture choisie permettant l'affichage du menu principal est le dessin d'un cercle. Un *TapGesture* sur le bouton permet d'afficher les sous éléments de ce bouton. Un *PanGesture* peut être utilisé sur le menu pour le déplacer.

Une fois que l'utilisateur ait choisi l'application à lancer, le menu disparaît et l'application apparaît. Si aucune activité n'est enregistrée sur le menu pendant une période, il commence à s'estomper jusqu'à disparaître complètement.

La structure du menu est décrite dans un fichier **XML**, ainsi il est possible d'avoir une structure hiérarchique, et aussi il est plus facile et plus rapide d'éditer son contenu.

L'environnement de bureau est constitué d'une scène sur laquelle seront affichés les éléments. La scène est une classe qui hérite d'une classe de **Qt** s'appelant *QGraphicsScene*. La scène est contenue dans un *QGraphicsView* qui sera affiché à l'écran. Tout les autres éléments qui sont sur la scène héritent de la classe *QGraphicsItem*.

Les événements du *QTouchEvent* qui sont reçus par l'application se propagent en allant du *QGraphicsView*, au *QGraphicsScene* ensuite au *QGraphicsItem* le plus bas sur la scène et commencent à remonter jusqu'à atteindre le *QGraphicsItem* le plus haut. Un *QTouchEvent* capté par un des éléments ne doit pas continuer à se propager et doit s'arrêter, on dit que l'événement a été consommé.

Ci dessous le diagramme de classes de base de l'environnement de bureau :

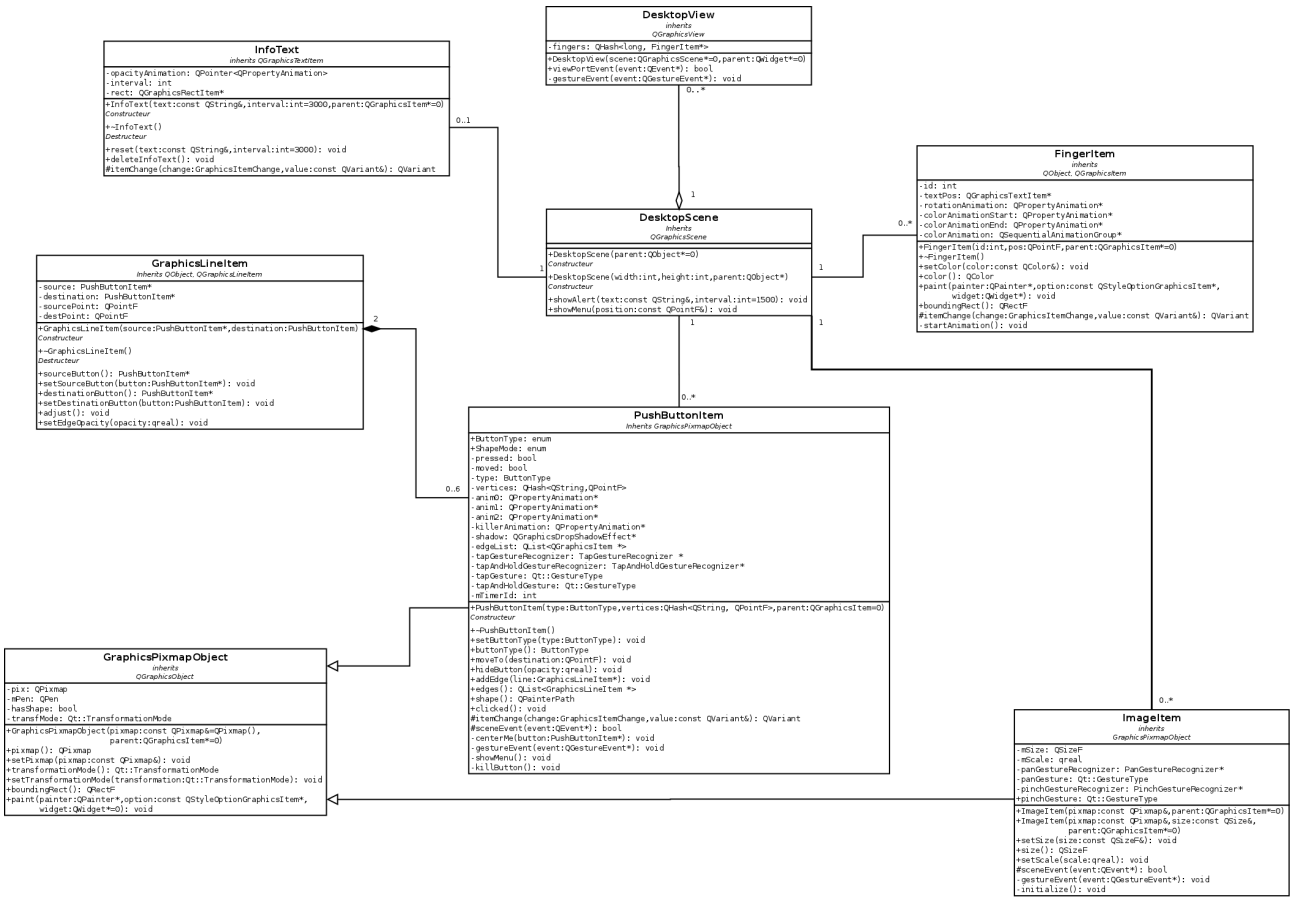

Illustration 42: *Diagramme de classe de l'environnement de bureau*

La classe *DesktopView* est la classe qui va afficher l'environnement de bureau. Elle contient la classe *DesktopScene* qui représente la scène sur laquelle les menus, les boutons, les images, et tout les autres éléments seront affichés. *DesktopScene* capte les *QTouchEvents* pour les analyser et les distribuer à ses fils (les éléments sur la scène).

La classe *PushButtonItem* est la classe permettant de créer le menu ainsi que les sous menus. Cette classe a besoin la classe *GraphicsLineItem* qui dessine le trait entre chaque bouton. Elle hérite de la classe *GraphicsPixmapObject* qui est une classe générique permettant de créer un élément pouvant afficher une image.

ImageItem est une classe représentant une image affichée sur la scène. Cette image peut être manipulée pour la déplacer, l'agrandir, la retourner, etc.

InfoText est une classe qui permet d'afficher une notification sur la scène pendant une durée de 3 secondes par défaut.

La classe *FingerItem* est une classe qui affiche la position d'un doigt qui est entrain de toucher la surface de la table. Elle dessine un anneau autour du doigt pour indiquer qu'il y a une interaction.

L'image ci-dessous illustre le bouton central qui affiche le menu, suite à une action de TapGesture :

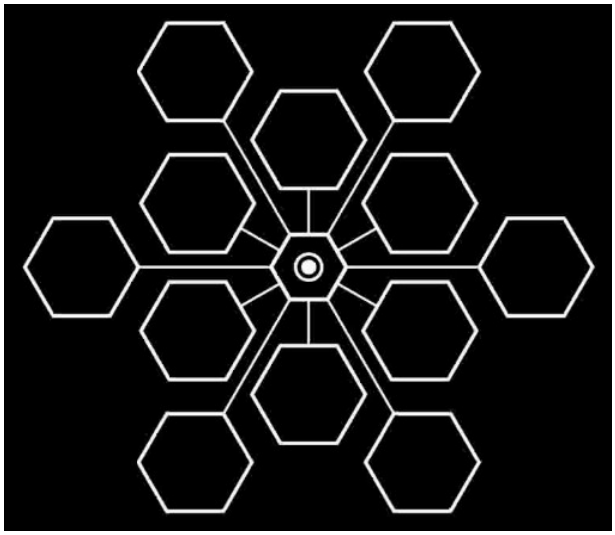

Illustration 43: *Structure du menu*

Chaque bouton fils peut être un parent et l'intérieur de l'hexagone peut, soit contenir du texte soit une image.

3- Synthèse globale

Durant cette phase de développement nous avons eu à concevoir d'une façon optimisée tout l'ensemble de la pile logicielle du système, en se basant sur le langage de programmation **C++** et le framework **Qt**. L'implémentation des classes est faite pour permettre une évolution par la suite et surtout pour permettre le développement d'autres applications pour l'environnement de bureau.

Le module de la gesture, quant à lui est indépendant de l'environnement de bureau. Ceci facilite la correction des bugs et aussi son amélioration. Actuellement ce module permet de reconnaître les gestures simples, en analysant l'évolution des « Touchpoints » dans le temps. Et permet aussi la reconnaissance des gestures complexes, qui sont des formes géométriques que l'utilisateur dessine sur la surface de la table multitouch avec son doigts.

qTuio est une passerelle nécessaire au fonctionnement de l'environnement de bureau. Cette passerelle, qui s'intègre à l'environnement de bureau en tant que « thread », permet la gestion des événements de la multitouch au niveau de **Qt** plus facilement.

Nous avons aussi eu à développer quelques applications que nous comptons intégrer dans notre système final. Parmi ces applications, une application qui simule le mouvements du sable, ci-dessous un « imprime écran » de cette application :

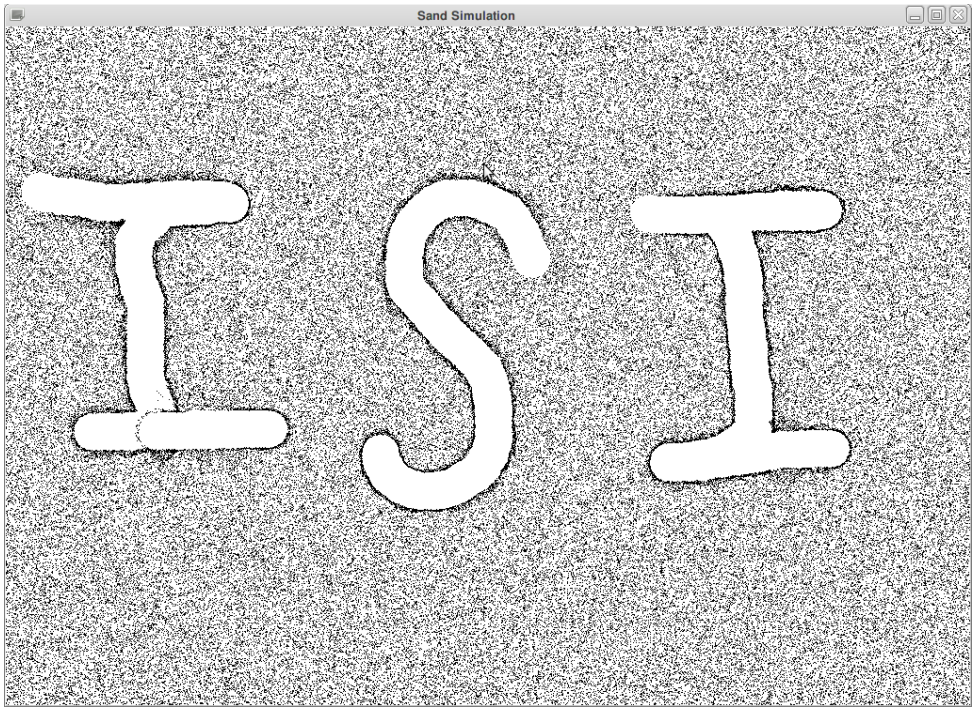

Illustration 44: *Simulateur du mouvement du sable multitouch*

4- Défis et perspectives

4.1- Défis

Défi 1 :

Le multitouch est un domaine encore en développement et plusieurs contraintes et problèmes techniques persistent jusqu'à aujourd'hui. Le plus grand problème c'est de deviner le contexte ; les dispositifs multitouch arrivent à savoir quand il y a un nouveau « TouchPoint » ou quand ce même « TouchPoint » se déplace ou disparaît, mais ils ne peuvent pas savoir dans quel contexte l'apparition ou la disparition du « TouchPoint » a eu lieu.

Un exemple simple avec une application de dessin : supposons qu'il y a une palette de couleurs et une surface de dessin, dans un premier temps, un « TouchPoint » apparaît sur la palette de couleur et la couleur rouge est sélectionnée ; tout de suite après le « TouchPoint » disparaît. Un autre « TouchPoint » apparaît et sélectionne la couleur verte et disparaît tout de suite après.

Maintenant les deux « TouchPoints » apparaissent sur la surface prévue pour le dessin ; il existe dans ce cas, plusieurs *scénarii*:

- L'utilisateur s'est trompé en sélectionnant la couleur rouge au début ; et donc il rectifie son choix en sélectionnant la couleur verte ; pour cela il dessine avec deux doigts ayant la même couleur (verte).
- L'utilisateur a sélectionné la couleur rouge avec un doigt et la couleur verte avec l'autre doigt et veut dessiner avec deux couleurs différentes.
- L'utilisateur s'est trompé en sélectionnant la couleur rouge ; et donc il rectifie en choisissant la couleur verte et veut dessiner avec deux doigts, mais le premier doigt garde sa couleur initiale. Il n'y a que la couleur du second doigt qui a changée.
- Il existe deux utilisateurs et chacun veut dessiner avec une couleur (identiques ou différentes).

Ceci constitue un aperçu des problèmes du multitouch. Dans certains cas il existe plusieurs *scénarii* possibles d'où l'ambiguïté.

Défi 2 :

Ce problème touche essentiellement à la gesture. Il est possible de reconnaître une gesture sur un objet, mais s'il y a deux gestures sur un même objet, la reconnaissance devient impossible.

Par exemple, il existe une image sur la scène et deux utilisateurs, le premier veut agrandir l'image et le deuxième veut la retourner et ceci en même temps. Donc il y a quatre « TouchPoints » en total. Le système ne peut pas distinguer les « TouchPoints » générant un agrandissement de ceux générant une rotation.

4.2- Perspectives

Ce travail a permis de concevoir les fondations pour une table multitouch. Il represente la base pour plusieurs types d'applications et d'utilisations.

Plusieurs parties de l'environnement de bureau restent encore en développement et plusieurs autres idées sont en attente d'être développées. Ce domaine n'a qu'une seule limite : l'imagination de l'être humain.

Parmi les évolutions possibles :

- Une amélioration de la reconnaissance de la gesture.
- Une amélioration de l'interface graphique.
- Développement d'applications supplémentaires.
- Possibilité d'administrer la table à distance.
- Inter-connectivité plusieurs tables distantes.
- Réécriture de **qTuio** en éliminant la bibliothèque **OSC** et en utilisant le module **QtNetwork** à sa place.

Il est même possible d'imaginer des panneaux publicitaires multitouch qui permettent à n'importe quel passant de savoir où il se trouve et même de récupérer l'itinéraire pour aller à un point donné.

La table multitouch peut aussi évoluer pour avoir une surface 3-dimensions et pouvoir détecter la distance entre la main de l'utilisateur et la table. De cette façon d'autres possibilités de développement s'ouvrent.

La caméra qui capte les rayons est une caméra infrarouge, c'est pour cela qu'elle ne peut avoir une image qu'au niveau du gris. L'ajout d'une autre caméra permettant d'enregistrer les images en couleur peut aussi ouvrir d'autres perspectives.

La table multitouch peut être un élément de la domotique. Elle peut être présente dans une maison pour faciliter les tâches journalières qui sont la consultation des mails, la lecture du journal ou bien la consultation des informations.

Conclusion et perspectives

Ce stage a été une expérience professionnelle très enrichissante que ce soit d'un point de vue approfondissement de mes connaissances en informatique, en électronique et en psychologie, que du point de vue relationnel et découverte d'un nouvel environnement de travail et une nouvelle mentalité.

J'ai eu le plaisir de contribuer au développement de plusieurs modules de la plate-forme multitouch, notamment le module de la gesture et l'architecture fondamentale de l'environnement de bureau. J'ai pu aussi participer à l'amélioration de certains projets communautaires qui touchent au même domaine tels que **Qt** et **qTuio**.

Le contact avec les autres développeurs m'a permis d'établir un échange de connaissances et de nouvelles idées, qui m'ont permises de progresser dans mon travail et de surmonter les problèmes rencontrés tout au long de la phase de développement.

Ce domaine d'interaction homme-machine est un domaine fascinant et met en œuvre plusieurs autres domaines (mathématique, physique, intelligence artificielle, psychologie, informatique, etc) qui le rendent encore plus excitant et riche en recherches.

De nouvelles perspectives s'ouvrent suite à cette expérience, qui visent essentiellement à innover et à créer de nouvelles solutions. L'intelligence artificielle couplée à l'interaction homme-machine me séduisent et me poussent à apprendre encore plus et surtout à essayer de contribuer à l'avancement technologique de la science.

Enfin, je tiens à exprimer ma satisfaction d'avoir pu travailler dans de bonnes conditions matérielles et dans un environnement agréable.

Annexes

Annexe A : Lois de Snell-Descartes

Annexe B : Fiducials

Annexe C : Multi-pointer X

Annexe D : Optimus

Annexe E : Diagramme de classe détaillé de qTuio

Annexe A : Lois de Snell-Descartes

Les précurseurs du domaine de l'optique étaient les deux persans **Abu Saâd Al Ala Ibn Sahl**[25] (un mathématicien qui a vécu entre 940 et 1000) et **Kamal AlDin AbulHassan Muhammad Al Farissi**[26] (un mathématicien et physicien qui a vécu entre 1267 et 1320).

Ibn Sahl avait découvert la loi de réfraction en 984 et a décrit ce phénomène dans un traité qu'il avait écrit et dans lequel il expose le procédé de la focalisation de la lumière en un point avec l'utilisation des miroirs courbes et des lentilles.

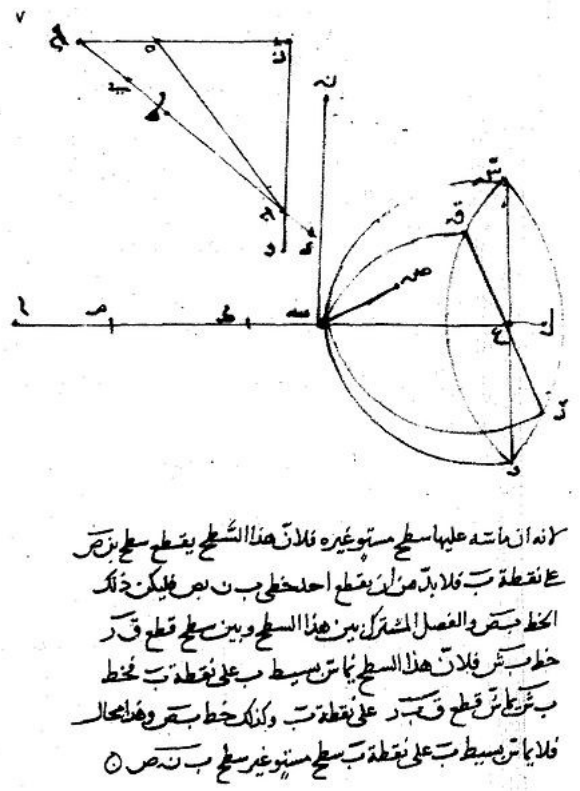

Illustration 45: *La loi de réfraction découverte par* ***Ibn Sahl***[27]

Environ 700 ans plus tard, **Snell** et **Descartes** redécouvrent les lois de l'optique. Les lois de Snell-Descartes sont au nombre de deux. La première décrit la réflexion et la deuxième la réfraction.

La première loi traite de la réflexion et s'énonce ainsi :

- Le rayon réfléchi est dans le plan d'incidence
- Les angles incidents et réfléchis sont égaux en valeurs absolus ; $\theta_1 = -\theta_2$

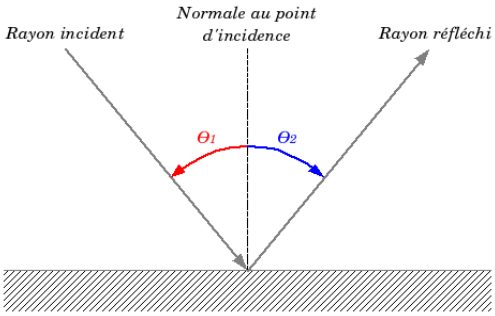

Illustration 46: *Loi de la réflexion*[28]

La deuxième loi traite de la réfraction qui décrit le changements de direction d'un faisceau lumineux lors d'un passage d'un milieu à un autre. Chaque milieu possède une caractéristique qui permet de définir la vitesse de propagation de la lumière. Cette caractéristique est appelée l'indice de réfraction n :

$$n = \frac{c}{v}$$

Avec v la vitesse de la lumière dans ce milieu et c la vitesse de la lumière dans le vide ($3*10^8$ m/s).

La loi de réfraction s'énonce ainsi :

- Le rayon réfracté est dans le plan d'incidence.
- La relation liant les indices de réfraction n_1 et n_2 de chacun des milieux et les angles incident θ_1 et réfracté θ_2 sont liés par la relation :

$$n_1 . \sin(\theta_1) = n_2 . \sin(\theta_2)$$

Quand $n_2 > n_1$ le rayon réfracté se rapproche de la normale :

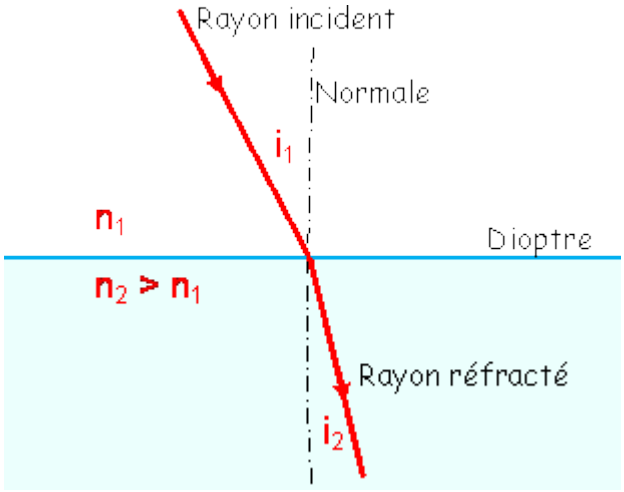

Illustration 47: *Réfraction avec* $n_2 > n_1$[28]

Quand $n_2 < n_1$ le rayon réfracté s'éloigne de la normale :

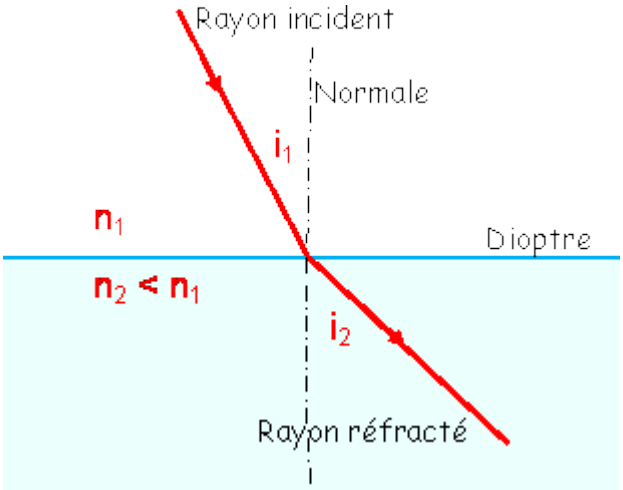

Illustration 48: *Réfraction avec $n_2 < n_1$*[28]

Si $n_2 = n_1$, cela veut dire que c'est le même milieu et donc il n'y a ni réflexion ni réfraction de la lumière.

Annexe B : Fiducials

Les « Fiducials » ont été inventés par les développeurs du framework reacTIVision. Ce sont des marqueurs facilement identifiables, autrement dit, des images uniques, créées dans le but de permettre à un système de les reconnaître et de lancer une action par la suite.

Ci dessous un exemple d'un marqueur :

***Illustration 49**:*
Fiducial[21]

Dans le domaine de la réalité augmentée, les « Fiducials » sont très utilisés. Il suffit d'une caméra ordinaire (même une webcam) et un système de reconnaissance de « Fiducial » pour pouvoir développer des applications touchant la réalité augmentée.

Les « Fiducials » sont très utilisés en physique (la **NASA** les utilise pour certaines de ses applications), dans les études géographiques, en imagerie médicale, dans la construction des circuits imprimés et bien d'autres domaines.

Annexe C : Multi-Pointer X

Multi-Pointer X (**MPX**) est une modification de **X.Org** permettant d'avoir plusieurs pointeurs de souris indépendants. Ceci permet d'effectuer plusieurs actions en même temps. Chaque pointeur peut effectuer des re-dimensionnements ou des déplacements des fenêtres sans gêner les autres pointeurs. De cette façon plusieurs utilisateurs peuvent utiliser un même ordinateur en même temps.

MPX a été développé en 2005-2008 par **Peter Hutterer** dans le cadre de son projet de doctorat à l'Université d'Australie-Méridionale. Il a été intégré à la version de développement de X.Org le 26 Mai 2008. Ensuite il a évolué et a été renommé en Xinput2 (XI2).

La dernière version de XI2 est incluse dans le paquet XServer 1.7 qui date du 2 Octobre 2009.

Annexe D : Optimus

La technologie **Optimus** est une invention du constructeur des cartes graphiques **NVidia**. Cette technologie est apparue fin 2010 et vise à contrôler l'activation de la carte graphique selon les exigences de l'utilisateur.

Optimus est actuellement utilisé dans les nouveaux ordinateurs portables et sera prochainement utilisé pour les ordinateurs de bureau. Le principe de cette technologie est simple : le microprocesseur d'une carte graphique puissante consomme énormément d'énergie, ceci diminue la durée de vie d'une batterie d'un ordinateur portable et diminue considérablement l'autonomie.

L'idée est de rajouter une autre petite carte graphique qui n'est pas puissante et qui consomme très peu d'énergie. Quand l'utilisateur a besoin d'une puissance graphique la carte graphique puissante s'active d'une façon transparente et effectue les taches demandées ensuite elle se désactive.

Optimus est la technologie qui succède le **Hybrid-SLI** qui a le même principe de fonctionnement sauf que la transition entre les deux cartes graphiques doit se faire manuellement.

Actuellement cette technologie n'est pas bien supportée sous GNU/Linux, ceci vient du fait que X.Org n'est pas développé pour permettre cette transition entre les cartes graphiques d'une façon transparente. Pour cela les développeurs de X.Org travaillent sur une nouvelle version (une réécriture complète) qui permettra d'exploiter cette technologie.

Annexe E : Diagramme de classe détaillé de qTuio

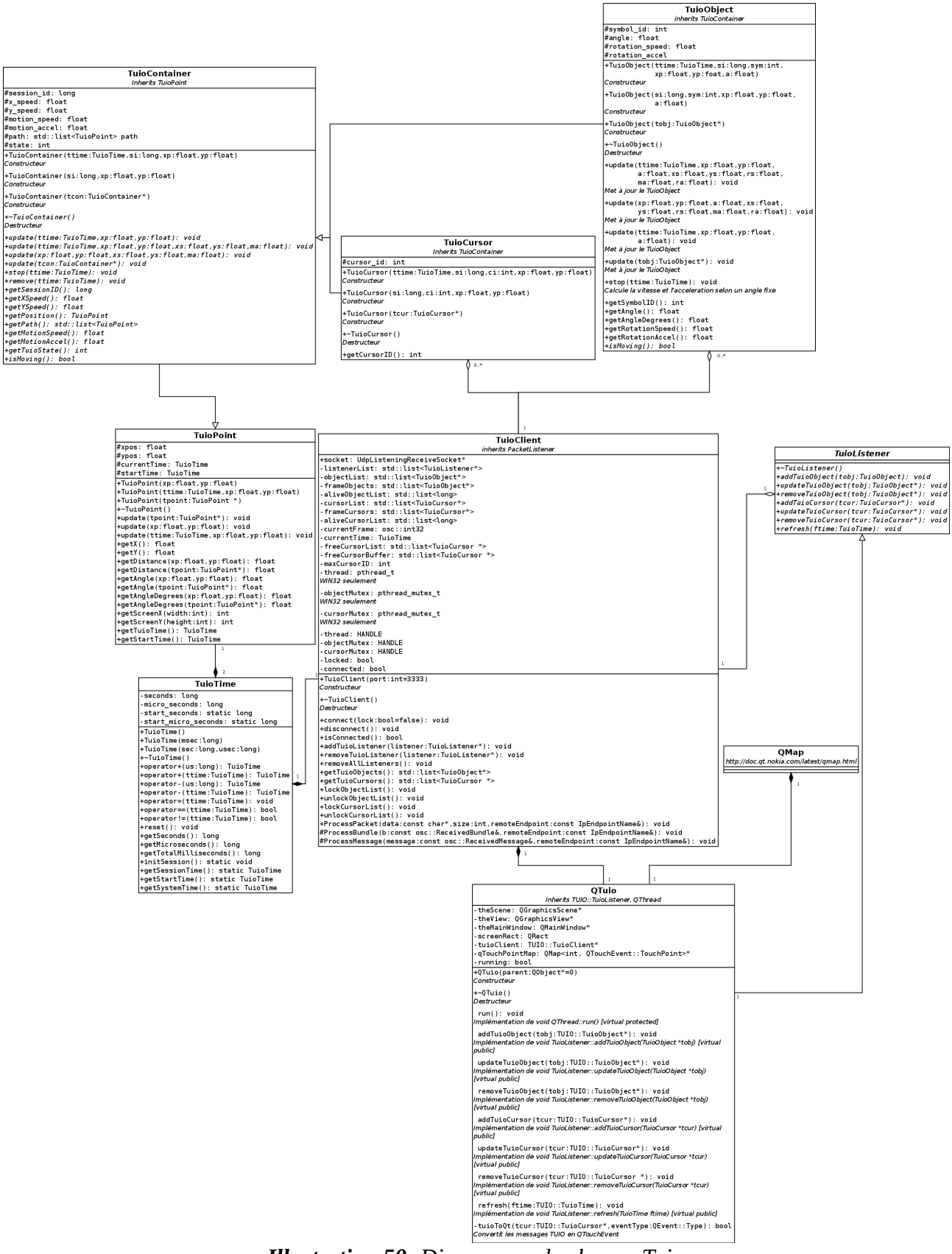

Illustration 50: *Diagramme de classe qTuio*

Références

Dernière date de visite : Mai 2011.

[1]: History of the Computer Part 2 - 1970-1980 (http://www.visionnet.nl/joomla/index.php/history-of/hardware/61-history-of-the-computer-part-2-1970-1980)

[2]: Macintosh 128K (http://fr.wikipedia.org/wiki/Macintosh_128K)

[3]: Présentation technique de la Wiimote (http://www.collegetabarly44.fr/news/spip.php?article306)

[4]: PlayStation Eye (http://en.wikipedia.org/wiki/PlayStation_Eye)

[5]: Présentation du Kinect pour XBox 360 (http://www.xbox.com/fr-FR/kinect)

[6]: OpenViBE Software for Brain Computer Interfaces and Real Time Neurosciences (http://openvibe.inria.fr/)

[7]: Télécommande Wii (http://fr.wikipedia.org/wiki/Wiimote)

[8]: Déscription de la Playstation Eye (http://fr.playstation.com/ps3/peripherals/detail/item78901/PlayStation%C2%AEEye/)

[9]: Microsoft surface description (http://www.microsoft.com/surface/)

[10]: Patschon, Mark (1988-03-15). Acoustic touch technology adds a new input dimension. Computer Design. pp. 89–93 (http://rwservices.no-ip.info:81/pens/biblio88.html#Platshon88)

[11]: Ecran tactile capacitif, fonctionnement de la technologie capacitive (http://interfacetactile.com/ecran-tactile-capacitif)

[12]: André Moussa & Paul Ponsonnet, Cours de Physique, Optique (Desvignes, Lyon, 1977).

[13]: Multi-Touch Sensing through Frustrated Total Internal Reflection (http://www.cs.nyu.edu/~jhan/ftirsense/)

[14]: TUIO 1.1 Protocol Specification (http://www.tuio.org/?specification)

[15]: What is Windows Aero (http://windows.microsoft.com/en-US/windows-vista/What-is-Windows-Aero)

[16]: Allan Pease - Body Language, How to read others' thoughts by their gestures.

[17]: Mouvements : glissement, panoramique et écartement (http://www.microsoft.com/windowsphone/fr-ca/howto/wp7/start/gestures-flick-pan-and-stretch.aspx)

[18]: Richard Stallman's Personal Home Page (http://stallman.org/)

[19]: GNU Operating System: Linux et le Projet GNU par Richard Stallman (http://www.gnu.org/gnu/linux-and-gnu.fr.html)

[20]: Guide d'installation et de configuration de Linux écrit par Christiant Casteyde. Chapitre 4: Présentation générale du système (http://linux.developpez.com/guide/c2462.html)

[21]: reacTIVision a toolkit for tangible multi-touch surfaces (http://reactivision.sourceforge.net/)

[22]: qTuio source code (https://github.com/x29a/qTUIO)

[23]: Gesture: One Finger Flick (http://gestureworks.com/support/supported-gestures/one-finger-flick/)

Bibliographie/Webographie

Dernière date de visite : Mai 2011.

1. Allan Pease - Body Language, How to read others' thoughts by their gestures.

2. Nuigroup - Multitouch Technologies.

3. La FTIR de Jeff Han (http://www.presence-pc.com/tests/ecran-tactile-22812/6/).

4. Magazine Mesure 740, décembre 2001 (http://www.mesures.com/archives/054_057_SOL.pdf).

5. Les différentes technologies d'écrans tactiles (http://fr.wikipedia.org/wiki/%C3%89cran_tactile).

6. NUI Group Community Wiki (http://wiki.nuigroup.com/Main_Page).

7. Lois de Snell-Descartes (http://fr.wikipedia.org/wiki/Lois_de_Snell-Descartes).

8. Lois de Snell-Descartes (http://www.techno-science.net/?onglet=glossaire&definition=6759).

9. Propagation de la lumière (http://villemin.gerard.free.fr/aScience/Physique/OPTIQUE/LDSnell.htm).

10. Histoire de la science (http://fr.wikipedia.org/wiki/Kam%C4%81l_al-D%C4%ABn_al-F%C4%81ris%C4%AB).

11. Histoire de la science (http://fr.wikipedia.org/wiki/Ibn_Sahl).

12. Multi-Touch sensing through frustrated total internal reflection (http://www.cs.nyu.edu/~jhan/ftirsense/).

13. Protocol de communicaition TUIO (http://www.tuio.org/).

14. Définition d'un gestionnaire de fenêtres (http://fr.wikipedia.org/wiki/Gestionnaire_de_fen%C3%AAtres).

15. Définition des système de fenêtrage (http://fr.wikipedia.org/wiki/Syst%C3%A8me_de_fen%C3%AAtrage).

16. Introduction to Window Manager (http://en.wikipedia.org/wiki/Window_manager).

17. Definition of Window Manager (http://dictionary.reference.com/browse/window%20manager).

18. Gesture Definition (http://en.wikipedia.org/wiki/Gesture).

19. Gestuelle des doigts et des mains (http://fr.wikipedia.org/wiki/Gestuelle_des_doigts_et_des_mains).

20. Gesture works multitouch made easy (http://gestureworks.com/).

21. Présentation du framework Qt (http://fr.wikipedia.org/wiki/Qt).

22. What's MultiPointer X (http://en.wikipedia.org/wiki/Multi-Pointer_X).

24. CCV presentation (http://amitsarangi.wordpress.com/).

25. Qt homepage (http://qt.nokia.com/).

26. C++ reference (http://www.cplusplus.com/).

27. Description of Fiduciary marker (http://en.wikipedia.org/wiki/Fiduciary_marker).

28. Optimus technology (http://www.nvidia.fr/object/optimus_technology_fr.html).

29. Nouveau driver for Optimus support (http://nouveau.freedesktop.org/wiki/Optimus).

30. Optimus Technology description (http://fr.wikipedia.org/wiki/Optimus_(NVIDIA)).