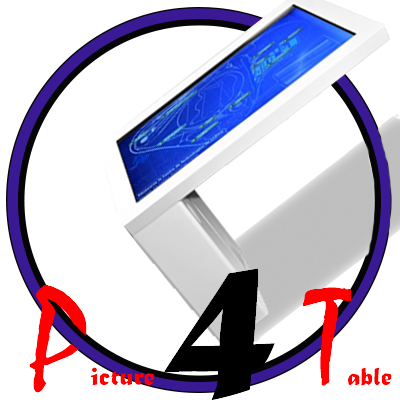
RAPPORT

DE

PROJET TUTEURE

Interface de demain, les grandes surfaces tactiles



Picture 4 Table

(Mars 2014)

**Réalisé par : BENKIRANE Mohamed Ali**

**DA SILVA CAMPOS Anis**

**DIALLO Amadou**

**TEBOUL Linda**

**Tuteurs : M. FENET Serge**

**M. PERRET Yannick**

SOMMAIRE

1.    Introduction 3

2.    Organisation du projet 4

2.1.  Elaboration du cahier des charges 4

2.2.  Analyse et documentation 4

2.3. Choix des outils 4

2.3.1. Choix de l'IDE 5

2.3.2. Gestion de version: 5

2.3.3. Librairies et framework: 6

2.4. Répartition des tâches 6

2.4.1. Diagramme de GANTT prévisionnel : 7

2.4.2. Diagramme de GANTT réel : 7

3.    Développement 7

3.1. Mise en place des librairies 7

3.2. Du mono-thread au multithread 9

3.2.1. Le mono 9

3.2.2. Le multi thread pour les Image 9

3.2.3. Les thread système 10

3.3. Le menu 10

3.4. Le conteneur et les fiducials 11

3.4.1. Le conteneur : 11

3.4.2. Les fiducials (ou Objets) 11

3.5. Problèmes persistants et taches non accomplies 11

4. Conclusion 13

4.1. Bilan Technique 13

4.2. Avenir du projet 13

4.3. Bilan personnel 14

4.3.1. DA SILVA CAMPOS Anis 14

4.3.2. BENKIRANE Mohamed Ali 14

4.3.3. DIALLO Amadou 15

4.3.4. TEBOUL Linda 15

5. Sources 16

# 1.    Introduction

En tant qu’étudiants en seconde année de DUT Informatique, nous avons appris au cours de notre cursus scolaire de nombreux langages de développements informatiques ainsi que des techniques de programmation.

C’est dans ce contexte là que nous avons été amenés à réaliser ce projet de fin d’études, autrement dit projet tuteuré, afin de les mettre en application.

Ce projet constitue une majeure partie de notre deuxième année puisqu’il occupera notre quotidien depuis le mois d’octobre et ce jusqu’à notre stage en entreprise, c’est donc un projet assez conséquent dans notre formation au sein de l’IUT.

Après la présentation des différents sujets lors de la réunion projets tuteurés par chacun des professeurs, c’est le sujet portant sur les grandes surfaces tactiles, tuteuré par Mr. FENET et Mr. PERRET, qui nous a attiré le plus.

Grand sujet d’actualité, en pleine expansion et qui a un impact sur beaucoup de personnes. En effet, du Smartphone aux tables tactiles, en passant par les tablettes, les surfaces tactiles font désormais partie intégrante de notre vie quotidienne. On constate près d’un milliard de Smartphones vendus dans le monde en 2013 et 60 millions de tablettes tactiles. Que ce soit pour consulter ses mails ou partager ses photos en ligne, chacun trouve son plaisir et son but à utiliser une surface tactile.

Malgré sa complexité, le sujet possède une richesse de problématiques qui laisse libre court à notre imagination. C'est dans cette optique que nous avons choisi de créer une application qui s'occupe de la gestion d'images des utilisateurs.

Par le biais d'une table tactile fabriquée par nos tuteurs, nous étions supposés la tester ainsi que l'optimiser pour permettre une utilisation multi-utilisateur.

# 2.    Organisation du projet

## 2.1.  Elaboration du cahier des charges

Nos tuteurs, nous ayant laissé libre cours à notre imagination concernant le choix du produit final à présenter, nous avons dû nous documenter et nous mettre d'accord sur un sujet. Plusieurs idées nous ont traversées l'esprit: développement d'une application qui détecte la position des doigts, développement d'un jeu tel que le casse briques, ou encore une application qui permet de visionner des photos à plusieurs autour de la table.

Nous avons finalement opté pour ce dernier choix qui nous intéressait le plus.

Nous avons donc réalisé un cahier des charges (ci-joint en annexe) que nous avons remis à nos tuteurs pour valider le sujet.

## 2.2.  Analyse et documentation

Avant de commencer la programmation, il fallait bien sûr au préalable faire une analyse de notre application, pour compléter les informations contenues dans le cahier des charges. C'est-à-dire définir son but final, ses fonctionnalités mais aussi sa modélisation UML pour comprendre comment l'application allait marcher.

## 2.3. Choix des outils



Durant la phase de documentation nous avions prévu de coder l'application en C++, c'est que par la suite que nous avons finalement décidé de programmer en java pour une meilleur portabilité de l'application, ainsi qu'une bonne compatibilité, pour ne pas dire total, car en effet, il y'a eu quelques petites contraintes que nous aborderons plus tard. De plus, étant donné que nous codions sur Windows et Mac et que l'OS prévu pour la table était LINUX, ce n'était qu'un argument en plus en faveur de JAVA.



### 2.3.1. Choix de l'IDE



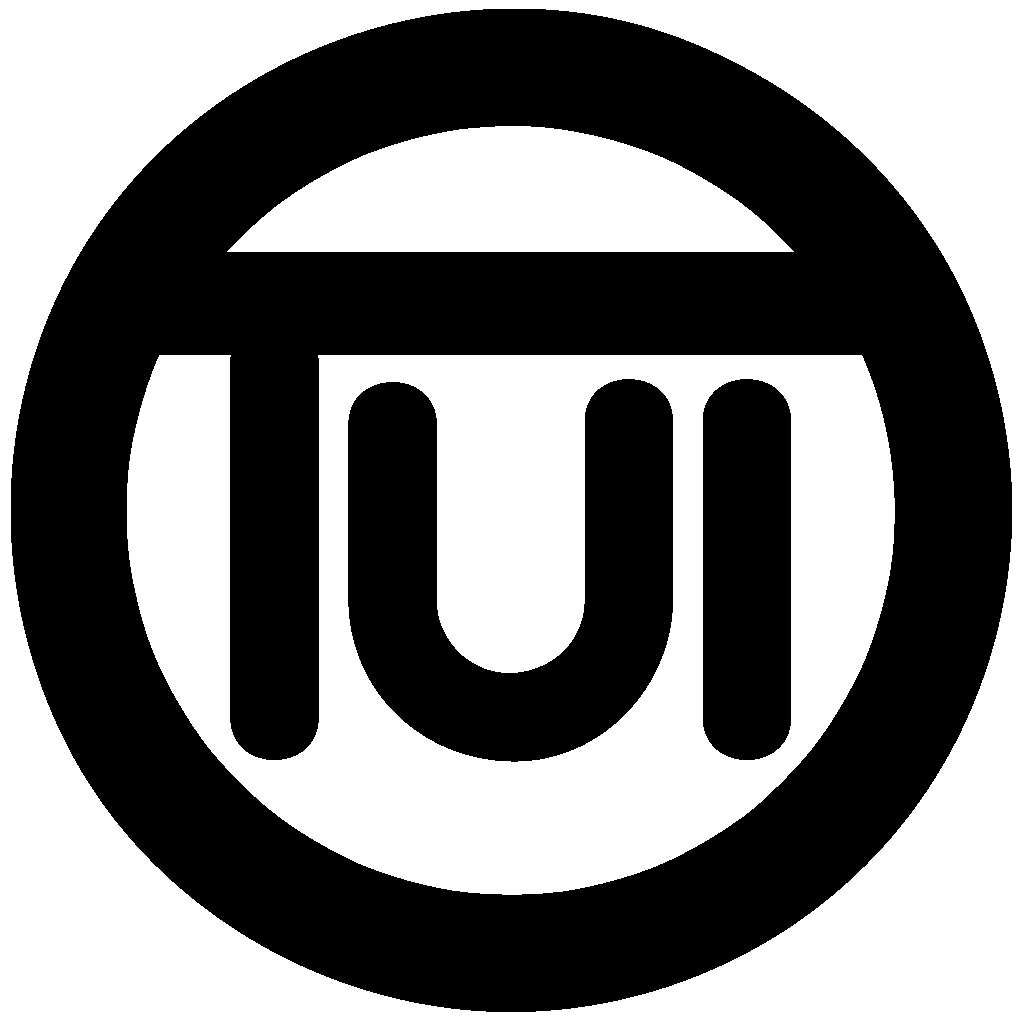
Le choix d'un IDE était essentiel pour le codage de l'application. En effet, un IDE facilite le développement et augmente la productivité. C'est ainsi que nous nous sommes décidé d'utiliser Eclipse car nous étions habitué à son environnement puisque depuis le début de notre cursus nous avons appris à l'utiliser via de nombreux TPs et programmes que nous avons dû réaliser. Ça a été donc un choix subjectif mais tout de même justifié par l'habitude de son utilisation. De plus, grâce à son assistance optimale (ajout systématique des accolades, parenthèses fermantes, indentations, recherche d'utilisation d'une variable ou méthode...), il nous semblait être l'outil le plus performant et rapide pour coder sans problème en JAVA.

### 2.3.2. Gestion de version:

Un projet informatique n'en est pas un sans un gestionnaire de version. En effet, le travail en équipe ne pouvait se faire sans la mise en place d'un répertoire auquel tous les membres du groupe pouvaient avoir accès. D'un autre côté, la sauvegarde de l'avancée de notre code était un aspect primordial pour le bien-être de notre application. C'est pour cela que nous avons choisi Git Hub. Un CVS que nous avons appris à utiliser pendant le cours d'AGL. De plus, c'est un service web d'hébergement et de gestion de développement efficace, sécurisé mais aussi gratuit, on ne pouvait trouver mieux. GitHub propose divers avantages tels qu'une interface console ou encore la gestion de branches pour faciliter le travail d'équipe. Simple et fonctionnel, il permet la modification d'une même ligne de code par deux personnes en même temps et gère ce genre de conflits.

### 2.3.3. Librairies et Framework:

#### 2.3.3.1. TUIO :

Du coté des librairies et Framework, Tuio est celui qui est le plus utilisé dans le domaine des tables tactiles. Son fonctionnement est simple, une application "tracker" analyse les images retransmises par la caméra afin de repérer les doigts et les objets. Grace à Tuio, le tracker va pouvoir encoder les données et les retransmettent à l'application cliente. Il existe déjà plusieurs applications permettant la détection comme "CCV" et "Reactivision" mais pas de réel application cliente utilisant ce protocole appart quelques simples démonstrations.

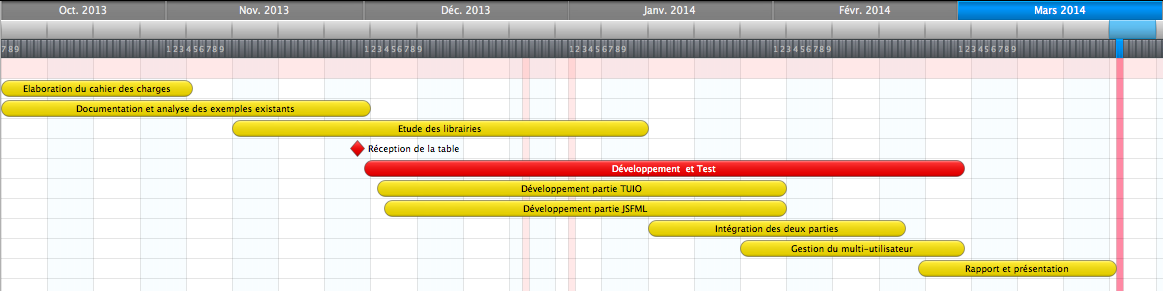
#### 2.3.3.2. JSFML:

Macintosh HD:Users:benkiranemohamedali:Desktop:jsfml.pngSFML est un open Framework multimédia basé sur OpenGL qui permet de gérer simplement l'audio, l'image, les fenêtres et le réseau. A la base codé en C++, il se décline sur beaucoup d'autre langage notamment JAVA : JSFML. Il n'y a pas beaucoup d'application qui l'utilise mais, grâce à la documentation très fournit en C++ et aux différents tutoriels ainsi que la Java doc,  nous avons pu nous l'utiliser facilement.

## 2.4. Répartition des tâches

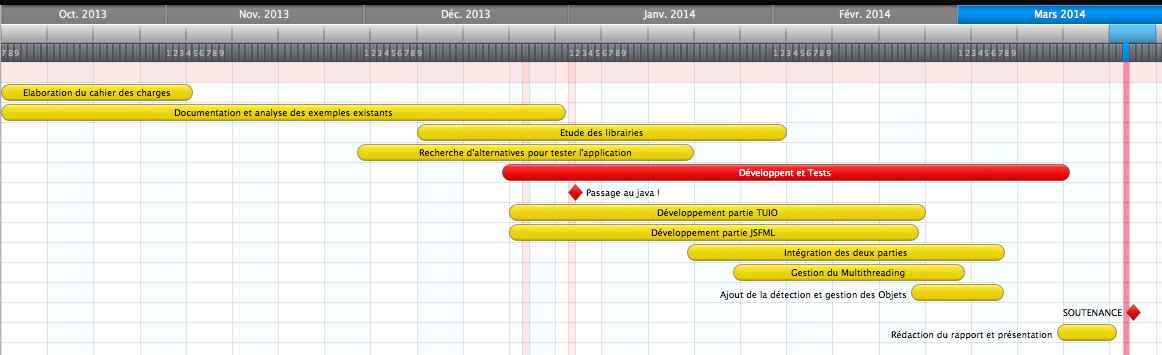
Nous avons choisi de diviser l'équipe en deux binômes. DA SILVA CAMPOS et BENKIRANE étant les personnes de meilleurs compétences techniques, nous avons choisi se partager en deux groupes et donc de se diviser les tâches en deux. Ainsi nous avons pu avancer en parallèle sans être retardé par un membre du groupe ni de mettre quelqu’un de côté. Les groupes étaient donc : DA SILVA CAMPOS Anis et TEBOUL Linda / BENKIRANE Mohamed Ali et DIALLO Amadou. Le premier binôme a travaillé sur la gestion de la détection des doigts mais et l’implémentation de toutes les méthodes en relation avec TUIO, pendant que le deuxième binôme travaillait plus sur la gestion des images, de l’affichage jusqu’à la mise dans le conteneur. On a donc pu trouver le moyen idéal de se partager les tâches pour réaliser une base à notre application qui est totalement fonctionnelle.

### 2.4.1. Diagramme de GANTT prévisionnel :



### 

### 2.4.2. Diagramme de GANTT réel :



# 

# 3.    Développement

## 3.1. Mise en place des librairies

La première étape du développement fut de mettre en place les librairies nécessaires au projet : Tuio et JSFML.

Pour commencer nous nous sommes inspirées des codes sources présents notamment sur le site de TUIO. La plus part de ces application ont été codées en C++ avec en plus des librairies comme SDL et OpenGL. Dans un premier temps nous avons commencé  par comprendre comment étaient structurée une application utilisant ce Framework. La première difficulté fut de dissocier le code responsable de l'affichage et celui de la réception des données du tracker. La seconde fut de porter ce code en java suite à la décision de changer pour plus de portabilité.

La réception des données se fait grâce aux classes TuioClient et TuioListener. La première permet de réceptionner et de décoder les messages Tuio émis par le tracker dans un port donné (3333 par défaut). Le client va ensuite générer des évènements qui seront interceptés par toutes les classes implémentant l'interface TuioListerner et répertoriées dans le client.

La classe TuioClient fournit une liste de TuioCursor et une liste de TuioObject .Dans TUIO les doigts sont représenté par la classe TuioCursor et les Objets part la classe TuioObject .Ces classes contiennent des  données comme les coordonnées, la vitesse et l'accélération. Afin de représenter des éléments sur l'écran, il faut leur donner une forme.

Pour l'affichage nous utilisons JSFML. Son utilisation est semblable à toutes les implémentations d'OpenGL, c'est-à-dire, dans un thread dédié à l'affichage, Il faut créer et configurer une fenêtre, puis dans une boucle de dessin dessiner les objets à l'écran. Pour résumer une boucle de dessin c'est :

**RenderWindow window = new RenderWindow();**

**window.create();**

**while (window.isOpen()) {**

**window.clear();**

**window.draw();**

**window.display();**

**}**

La méthode **draw** prend comme argument des formes de base comme des Cercles (**CercleShape**) et des Rectangle (**RectangleShape**) et plus généralement tous les  éléments implémentant l'interface "**Drawable**" qui demande d'implémentation d'une méthode : **draw().** Pour afficher une image il faut appliquer une texture, l'image, sur un rectangle : le **Sprite**.

Pour finir, l'intégration de ces deux librairies au projet passe par deux simples fichiers "**.jar**" à placer dans le dossier du projet.

## 3.2. Du mono-thread au multithread

### 3.2.1. Le mono

Dans un premier temps nous avons implémenté un programme qui gère une image. Apres quelque recherche, nous avons compris que cette gestion devait se faire en parallèle de l'affichage. En effet, il faut un "Thread'" dédié à l'affichage et un "Thread" dédié à la gestion de l'image. Les "Threads" sont des processus qui fonctionnent en parallèles mais qui partagent le même code et les même données.

C’est pourquoi, nous avons créé la classe **GesteImage** à qui en associe une image et qui va, en boucle, repérer les TuioCursor présents dans l'image. Si il y a un curseur, alors en appellera la fonction déplacer et si il y en a deux se sera zoomer-pivoter.

### 3.2.2. Le multi thread pour les Image

Cependant, si cette première implémentation fonctionnait, et donc qu'avec une image tout allait bien, à partir de deux images nous avons rencontré deux soucis :

- Premièrement : en déplaçant une image, s’il y avait une autre image sur le parcours, elle était "happée" et se déplaçait aussi.

- Deuxièmement : le plan de l'image est fixé par l'ordre de dessin dans la boucle draw(), qui est static.

Pour résoudre le premier nous avons amélioré la synchronisation des ressources à l'intérieur de **GesteImage** par exemple avec une liste des curseurs attribués à une image. Cette liste permet aux threads de savoir quels curseurs sont disponibles ou pas et grâce à cette liste un curseur ne déplace plus qu'une image à la fois.

Pour résoudre le second il faut réordonner les images dans le plan, c’est-à-dire, la dernière image touchée doit être affichée en dernier donc être au premier plan.

Le second a été résolu par la création de deux nouvelles classes : **Image** et **ListImage**. Elles implémentent toute les deux l’interface **Drawable** et peuvent donc être affiché à l’écran (grâce à la méthode **draw()** ).

**ListeImage** est une classe conteneur qui va stocker des Images dans un ArrayList. C’est grâce à la méthode Sort() de  Collections que nous trions les Images selon leur date de dernier accès (attribut de la classe Image). Dans le **draw()** de **ListeImage**, d’abord on trie le ArrayList, puis on appelle le draw() de chaque Image.

La classe I**mage** va contenir l'image bien sûr, mais aussi un thread de **GesteImage**. Maintenant créer une image créer aussi sont thread associe et supprimer cette image passe par l'arrêt de ce thread, ce qui facilite beaucoup leurs utilisation.

### 3.2.3. Les thread système

Une fois les threads de la gestion des gestes des images mis en place, il nous manquait la possibilité de détecter des gestes dans non liées aux images. Nous avons donc créé la classe **GesteSysteme**. Son fonctionnement est très similaire à celui de GesteImage, à la différence que les TuioCurseur qui l’intéresse sont ceux qui ne touchent aucune image.

 Grace à lui, nous pouvons afficher un menu et gérer la sélection d’une de ses options.

## 3.3. Le menu



Le menu est une composition de plusieurs **rectangleShape** et de conditions. On a implémenté une méthode qui prend en paramètre un curseur (donc deux coordonnées) et une fois que le curseur est positionné sur le menu, on teste plusieurs conditions du genre :

**If ( isInFermer(Cursor) ) {**

**System.exit(0) ;**

**}else if (isInHelp(Cursor) ) {**

**help.setVisible(true) ;**

**} …**

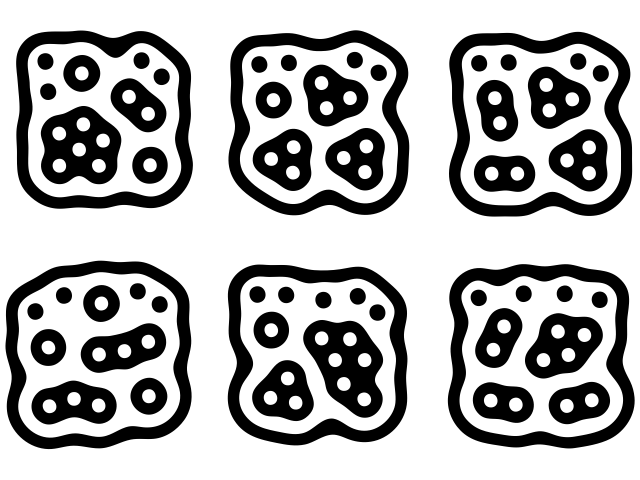
## 3.4. Le conteneur et les fiducials

### 3.4.1. Le conteneur :

Le conteneur est l'équivalent d'un dossier des explorateurs de fichier. Son rôle est de contenir des images qui pourront être sauvegardé par une sérialisation, ou bien charger par dé-sérialisation. En effet, l’utilisateur peut déplacer des images au niveau du conteneur, une fois qu’il appuie dessus (à l’intérieur du conteneur), l’image est minimisé, et se place à gauche du conteneur. L’utilisateur peut placer autant d’images qu’il veut, celles-ci seront placée l’une à côté de l’autre. Une fois qu’il veut enregistrer son dossier pour pouvoir le récupérer lorsqu’il le voudra, il n’a qu’à ouvrir le menu, et appuyer sur le bouton sauvegarder. Là un clavier est affiché à l’écran qui permettra à l’utilisateur d’entrer le nom du fichier. Le fichier est ensuite enregistré dans le répertoire courant de l’application.

Pour implémenter ce conteneur en lui-même n’est à la base qu’un **rectangleshape** qu’on place en bas. Lorsqu’on clique sur l’image à l’intérieur du conteneur, on récupère **l’URL** de l’image qu’on ajoute à un **ArrayList**. Et c’est ensuite cet **ArrayList** qui est **sérialisé**.

### 3.4.2. Les fiducials (ou Objets)

  
 Comme sur la photo ci-contre, les fiducials sont des sortes de QR Code reconnaissable par Reactivision (Logiciel de détection de doigts grâce à une caméra).

De plus, le logiciel assigne à chaque fiducials un ID qui lui est propre. Et c’est grâce à cet ID qu’on a pu gérer l’ajout de librairie d’images ou de musique. Pour tester cela, nous avons récupéré le logiciel TUIO\_Simulator, qui simule non seulement le mouvement du curseur mais aussi l’ajout de fiducials.

## 3.5. Problèmes persistants et taches non accomplies

A ce jour, certains problèmes rencontrés durant la phase de développement n'ont toujours pas étés résolus.

- En effet, l'orientation du programme est static, quel que soit la position de l'utilisateur

- L’impossibilité d’ouvrir plusieurs JFileChoser car Java ne le permet pas.

- La gestion de la souris sous WINDOWS n’est pas la même que sous MAC OS. Donc lorsqu’on ouvre le JFileChoser sur MAC cela fait bugger l’application.

- L'identification de l'utilisateur, c'est-à-dire, savoir à qui appartient tel doigt. (Au final, même s'il y a 32 doigts, l’application gère comme s’ils sont à la même personne !)

# 4. Conclusion

## 4.1. Bilan Technique

     Ce projet nous a permis tout d'abord de mettre en pratique toutes les connaissances acquises lors de notre formation. Langage java, programmation Système, AGL, réseaux ... C'est toutes ces notions étudiées lors de ces cours qui nous ont permis de réaliser ce projet. De plus, ça nous a permis d'acquérir de nouvelles compétences tel que le multi threading, la boucle d'affichage et l'étude de certaines librairies qui ne sont pas étudiées à l'IUT. De plus, nous avons appris à réaliser un projet de son analyse, jusqu’à son rendu mais aussi d’améliorer notre capacité à travailler en groupe, chacun ayant son mot à dire, il est nécessaire de développer le sens de l'écoute.

Un tel projet nous aide permet de nous entrainer à respecter les délais imposés, faire des analyses réfléchies, savoir gérer les imprévus en trouvant des solutions aux problèmes. Donc nous pensons qu’il n y a pas mieux que le projet tuteuré pour nous faciliter notre insertion dans le monde professionnel, c’est-à-dire le stage que nous devons commencer en début Avril.

## 4.2. Avenir du projet

Le projet sera surement reprit par les étudiants des promotions futures comme base pour rajouter de nouvelles fonctionnalités selon leurs choix et le temps qu'ils possèdent pour le réaliser. On leur fourni donc la **javadoc** de notre projet disponible sur le lien suivant :

**www.benkiranemedali.fr/p4t/doc/**

Ils pourront même réaliser d’autres applications en s'inspirant de celle-ci. Ce qui augmentera la valeur ajoutée de la table. Le projet est initialement dédié au multi-touch donc la première évolution serait d’adapter l'application à une table tactile en passant par le logiciel CCV (Community Core Vision) ou Reactivision.



Ils auront la possibilité d’ajouter d’autres applications ainsi qu'un menu qui guidera vers celles-ci tel que des jeux (casse brique, pong…) ou encore une table de mixage par exemple. Ils peuvent aussi gérer le côté multi-utilisateur, la division de la table en différentes parties où chaque personne a la sienne.

## 4.3. Bilan personnel

  Ce projet a été un véritable échange entre nous, c'était une aventure humaine que technologique, il a fallu faire cohabiter les membres du groupe qui ont des productivités différentes et un emploi du temps pas toujours compatibles. Mais aussi d’apprendre à nous auto former en cas de besoin d’informations.

Nous avons donc pu développer plusieurs qualités :

La responsabilité

L’autonomie

Le travail en équipe

La communication.

### 4.3.1. DA SILVA CAMPOS Anis

Grace au projet tuteuré j'ai pu mettre en œuvre mes connaissances en programmation mais aussi apprendre d'avantage. J'aime beaucoup les nouvelles technologies et ce projet fut une belle façon de s'en rapprocher.

Je regrette seulement de ne pas pouvoir tester cette application dans de réelles conditions, cela provoque un sentiment d'inachevé.

### 4.3.2. BENKIRANE Mohamed Ali

Ce projet m’a été bénéfique car j’ai pu apprendre l’utilisation de deux librairies chose qu’on n’apprend pas forcément en cours. De plus, il m'a permis d'avoir une idée sur ce qu'est un vrai projet, et comment s'y prendre, en passant par toute les étapes: analyse, conception, et rendu.

La contrainte majeure à mon avis a été celle du temps. En effet, le programme chargé du semestre 4 ne nous a pas permis de terminer ce projet dans son intégralité, mais surtout, cela ne nous a pas permis d'apporter une réelle touche personnelle à ce projet. Nous trouvons cela dommage, mais nous gardons les idées de côtés pour pouvoir les mettre en place une fois qu'on aura le temps.

### 4.3.3. DIALLO Amadou

Ce projet a été l’occasion pour moi de passer des savoirs théoriques à leur application pratique.

La phase d'analyse et de conception du projet m’a permis d’acquérir de nouvelles connaissances, notamment les librairies JSML et Tuio.

Je pense réellement que ce fut une grande expérience pour moi. Riche en apprentissage mais aussi en  expérience même si on n'a pas eu la possibilité  et le temps de terminer le projet avec l'adaptation à la table tactile.

### 4.3.4. TEBOUL Linda

En plus de bénéficier de l'apprentissage du JAVA ainsi que de nouvelles librairies, ce projet m'a permis de concevoir une application et de constater les difficultés qui peuvent apparaître comme le manque de temps, les contraintes matérielles mais aussi humaines. Les différentes étapes du projet sont plus complexes qu'on le pense, et le développement ne se résume pas seulement à coder. Ainsi, j'ai pu me rendre compte que le développement d'applications n'est pas le domaine dans lequel je souhaiterais travailler plus tard, et c'est sur ce point précis que je ne regrette pas le choix que j'ai fait.

# 5. Sources

**Javadoc : http://docs.oracle.com/javase/6/docs/api/**

**TUIO :** [**www.tuio.org**](http://www.tuio.org)

**SFML :** [**www.sfml-dev.org/**](http://www.sfml-dev.org/)

**JSFML :** [**www.jsfml.org**](http://www.jsfml.org)

**Comunity core vision :** [**www.ccv.nuigroup.com**](http://www.ccv.nuigroup.com)

**Reactivision :** [**www.reactivision.sourceforge.net**](http://www.reactivision.sourceforge.net)