École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique

RAPPORT DE PROJET DE FIN D'ETUDE

Projet :	ScanSchem: développement d'une application multi-touch semi-automatique pour la réingénierie de documents techniques	
Emetteur :	Christophe Guinchard	Coordonnées : EPU-DI
Date d'émission :	20/05/2011	

Validation

Nom	Date	Valide (O/N)	Commentaires

Historique des modifications

Version	Date	Description de la modification
00	09/05/2011	Version initiale sans la partie implémentation
01	20/05/2011	Version finale

Introduction

Le projet suivant a été réalisé dans le cadre de mon projet de fin d'étude à l'école Polytech'Tours. Ce projet a pour but de valider notre parcours ingénieur par la mise en pratique de l'élaboration complète d'un projet de développement, avec rédaction du cahier des spécifications, implémentation de la solution et rédaction d'une documentation.

Le projet sur lequel j'ai été amené à travailler a été proposé par Mathieu Delalandre, enseignant chercheur de l'école, en partenariat avec la société Pos-Industry, entreprise travaillant sur le développement de logiciels utilisés dans le domaine industriel. Ce projet consiste à travailler sur la problématique de la réingénierie des documents techniques. En effet, aucune méthode efficace et totalement automatique n'existe à ce jour et ce projet va porter sur une méthode semi-automatique temps réel qui devra nécessiter un minimum de sollicitations d'un opérateur. On va en particulier principalement s'intéresser à la reconnaissance de symboles sur une image à partir d'une base de symboles SVG.

Concernant les moyens de mise en œuvre, le projet se découpe en deux parties : le développement d'une bibliothèque d'indexation en C++, permettant de récupérer les informations nécessaires à la reconnaissance de symboles, et la partie développement de l'interface en Objective C, pour la technologie tactile d'Apple, l'IPad.

Précision tout de même que ce projet a été abandonné par Pos-Industry au tout début de son déroulement. Les travaux ont donc été réalisés indépendamment.

Sommaire

I – Présentation du projet

1 Contexte 8

2 Solution proposée 8

3 Objectifs 8

4 Contraintes de développement 9

II – Présentation de la technique de reconnaissance de formes

1 Avantage des données vectorielles 10

2 Technique de reconnaissance des formes : le template matching 10

3 Solution pour réduire la complexité : pré-calcule des données 10

III – Vue d'ensemble du système

1 Bibliothèque d'indexation 12

1.1 Echantillonnage 12

1.2 Calcul de la transformée des distances 13

2 Application IPad 13

IV – Fonctionnalités du système

1 Bibliothèque d'indexation 15

1.1 Fonctionnalité 1 : Parsage des fichiers SVG 15

1.2 Fonctionnalité 2 : Echantillonnage des formes 15

1.3 Fonctionnalité 3 : Squelettisation 15

1.4 Fonctionnalité 4 : Calcul de la transformée de distance 15

2 Application IPad 15

2.1 Fonctionnalité 5 : Chargement des données 15

2.2 Fonctionnalité 6 : Interactions utilisateur 16

2.3 Fonctionnalité 7 : Reconnaissance du symbole 16

V – Organisation prévisionnelle

1 Découpage du projet en tâches 17

1.1 Tâche 1 : Etude de la technologie tactile 17

1.2 Tâche 2 : Proposition d'une interface et démonstration visuelle de la future application 17

1.3 Tâche 3 : Etude et implémentation de la bibliothèque d'indexation 17

1.4 Tâche 4 : Implémentation de l'interface de l'application 17

1.5 Tâche 5 : Intégrer le moteur de reconnaissance de forme à l'interface 17
1.6 Tâche 6 : Test de l'application 17
2 Planning prévisionnel 17

VI – Méthodes théoriques

1 Bibliothèque de d'indexation 18
1.1 Echantillonnage 18
1.1.1 Parsage du fichier SVG 18
1.1.2 Echantillonnage des formes 18
1.2 Calcul de la transformée de distance 22
1.2.1 Squelettisation 22
1.2.2 Calcul de la carte de transformée de distance 22
2 Application IPad 23
2.1 Interaction utilisateur 23
2.1.1 Solution 1 23
2.1.2 Solution 2 24
2.1.3 Solution 3 24
2.1.4 Choix de la solution 25
2.2 Reconnaissance de symbole 25

VII – Implémentation

1 Bibliothèque d'indexation 26
1.1 Echantillonnage 26
1.1.1 Diagramme de classe 26
1.1.2 Utilisation 27
1.1.3 Description des classes 27
1.1.4 Bibliothèque utilisée 27
1.2 Transformée de distance 28
1.2.1 Diagramme de classe 28
1.2.2 Utilisation 28
1.2.3 Description des classes 28
1.2.4 Bibliothèque utilisée 29
2 Application IPad 29
2.1 Diagramme des classes 29
2.2 Description des classes 29

2.2.1 SchemScanDelegate 29
2.2.2 SchemScanController 29
2.2.3 ViewSelection 29
2.2.4 DtTable 29
2.2.5 Symbole 29

VIII – Avancement du projet aujourd'hui

1 Comparaison plannings prévisionnels et effectifs 30
2 Etat d'avancement 31

Conclusion

Bibliographie

I – Présentation du projet

1 Contexte

Ce projet est accès autours de réingénierie de documents techniques. En effet, pour des raisons de propriété intellectuelle, de coûts et de moyens, les documents techniques sont souvent transmis sous format papiers, fax ou documents numérisés. Il en résulte chez les entreprises et les milieux académiques, un grand nombre de documents papiers stockés et un appauvrissement de la qualité à chaque diffusion.

Pour résoudre ce problème, la solution idéale serait d'être capable d'automatiser un système de reconstitution de ces documents techniques dans un format vectoriel. Pour cela, des recherches ont été entreprises. Malheureusement, à ce jour, aucune technique vraiment viable n'a pu être proposée en raison d'un trop grand nombre de variabilités selon la qualité et le formatage du document. Les entreprises ont donc souvent recourt à des techniques entièrement manuelles de réingénierie de ces documents, qui sont des solutions longues et couteuses et de qualité limitée.

Comme les solutions du tout automatique ne sont pas suffisamment viables pour être utilisées, une solution semi-automatique est donc envisagée. En effet, les recherches dans ce domaine ont permis de mettre en évidence des méthodes de reconnaissance de formes sur des éléments individuels tout à fait viables (symboles, textes, lignes, etc.). Il ne s'agit donc pas d'une solution au problème mais d'une manière de compenser les défaillances des techniques automatiques.

Ces éléments doivent donc pouvoir être pré-localisés par l'utilisateur. Pour cela, les technologies tactiles pourraient fournir une interface intuitive et conviviale permettant de considérablement faciliter les opérations de réingénierie de ces documents techniques.

2 Solution proposée

Pour répondre à la problématique de ce projet, on propose de découper le projet en 2 parties :

- Développement d'une bibliothèque d'indexation : qui aura pour but de calculer toutes les informations nécessaires pour la reconnaissance de symboles hors de l'application IPad et ce pour respecter la dimension temps réel.
- Développement de l'interface de démonstration IPad

3 Objectifs

Ce projet se découpe en plusieurs objectifs :

- **Prise en main de la technologie tactile** : Ceci inclus l'apprentissage du langage natif et de l'environnement proposé.
- **Développement de la bibliothèque d'indexation** : Celle-ci est basée sur des algorithmes existant et doit être implémentée sous forme d'une bibliothèque C++ pour être réutilisable dans d'autres applications futures.
- **Mise en place d'une interface complète de démonstration** : Cette interface a pour but de pouvoir démontrer les performances de la solution semi-automatique. Pour le moment, seule la fonctionnalité de reconnaissance des symboles est impérativement nécessaire.

4 Contraintes de développement

Contraintes liées au projet :

- ***Matériels*** : Utilisation de la technologie tactile d'Apple, l'IPad
- ***Langages de programmation imposés*** :
    - *Interface* : Objective C pour l'IPad
    - *Bibliothèque d'indexation* : C++
- ***Environnements de développement*** : Xcode et Microsoft Visual Studio 2008
- ***Algorithmes imposés*** : L'algorithme du moteur de reconnaissance sera basé sur les travaux du document « Sanchez, G. & Llados, J. Blurred Shape Model for binary and grey-level symbol recognition Pattern Recognition Letters ».
- ***Délais de réalisation*** : Fin des PFE.

II – Présentation de la technique de reconnaissance de formes

1 Avantage des données vectorielles

Le principal but de la réingénierie des documents industriels et de pouvoir générer des données vectorielles à la place des données BTMAP. En effet, le format BITMAP comporte de nombreux inconvénients pour le stockage d'informations graphiques :

- La qualité du document est très variable selon la méthode d'acquisition et la qualité ira toujours en se détériorant à chaque manipulation.
- Les données sont beaucoup plus volumineuses
- Le format n'est pas exploitable pour réaliser des modifications

Le format vectoriel en comparaison propose un stockage d'informations structurelles qui permettent de toujours conserver une qualité constante, quel que soit les opérations réalisées sur celles-ci, et les données stockées sont très peu volumineuses. Les données vectorielles peuvent aussi être modifiées à tout moment (déformation, ajout d'éléments structurels, etc.).

2 Technique de reconnaissance des formes : le template matching

La technique de reconnaissance des formes que nous allons utiliser est le template matching. Cette technique graphique consiste à prendre une partie d'une image et de la comparer à un template d'images vectorielles pour déterminer une correspondance.

Dans notre cas, l'utilisateur doit pouvoir sélectionner un symbole du schéma BITMAP et celui-ci sera mis en correspondance avec une base de symboles vectoriels. Pour cela, on va tester un ensemble points de l'élément vectoriel et les mettre en correspondance avec la transformée de distance correspondant à la partie de l'image sélectionnée. On va ainsi pouvoir obtenir un écart-type des distances pour chaque symbole qui permettront de faire un classement de correspondance des symboles.

Cette technique a pour avantage d'être particulièrement robuste face aux éléments parasites présents sur l'image. Cependant, l'inconvénient est qu'elle est aussi très gourmande en temps de calcul à cause de sa complexité.

En effet, cette mise en correspondance nécessite de calculer au préalable la transformée de distance de l'image BITMAP, mais aussi d'échantillonner les images templates vectoriels pour récupérer un ensemble de points. Ces opération son très gourmandes en temps de calcul et ne sont pas applicables pour la réalisation d'une application temps réel.

3 Solution pour réduire la complexité : pré-calcule des données

Une solution que nous proposons pour réduire la complexité du template matching est de pré-calculer les données qui seront réutilisées pendant l'opération de mise en correspondance.

La première chose que l'on peut pré-calculer est la transformée de distance. En calculant la transformée de distance sur l'image BITMAP complète, on pourra récupérer par la suite la zone correspondante au symbole pour la mise en correspondance.

Nous pouvons aussi pré-échantillonner une liste de points pour chacune des images vectorielles.

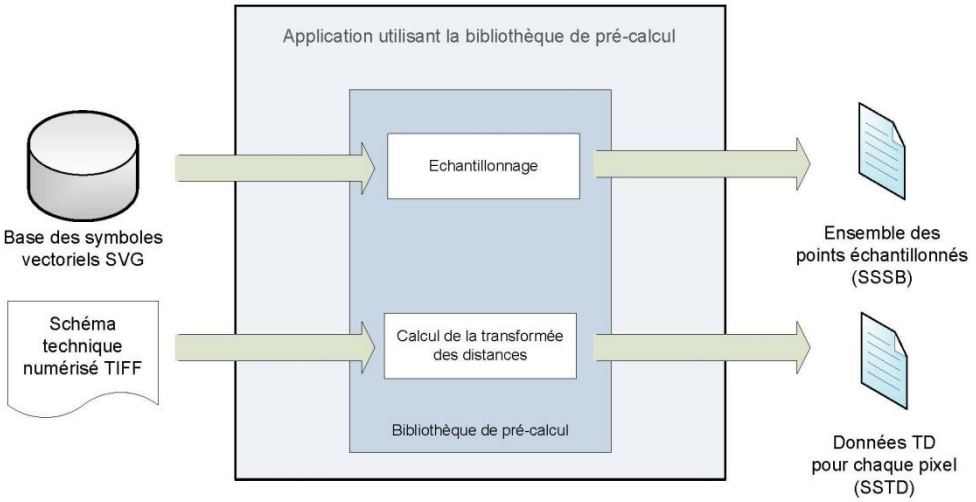
L'ensemble de ces données étant pré-calculées, il ne restera plus qu'à réaliser l'opération de mise en correspondance dont la complexité est beaucoup moins importante et est variable en fonction du nombre de symboles et du nombre de points sur les symboles.

On appellera ces données pré-calculées, **les données indexées**.

III – Vue d'ensemble du système

L'environnement du projet se découpe en deux parties distinctes. La première consiste à réaliser l'indexation avec la bibliothèque développée. La seconde englobe toute l'application IPad utilisant le résultat de ces calculs.

1 Bibliothèque d'indexation

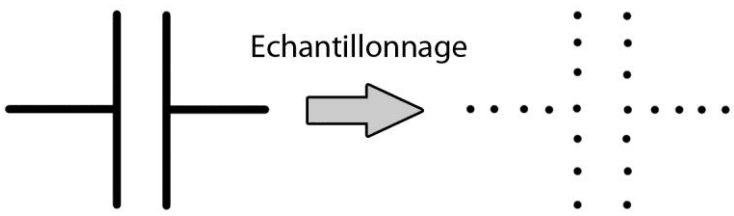


Cette étape est réalisée avant même le lancement de l'application IPad sur une machine externe. Elle consiste à utiliser une application qui exploitera la bibliothèque d'indexation. On va donc utiliser deux types d'informations ici :

1.1 Echantillonnage

Avec les symboles SVG, la bibliothèque va échantillonner ceux-ci pour obtenir une liste de points. Le nombre de points sera variable en fonction d'une valeur correspondante à la distance entre points qui sera indiquée par l'utilisateur lors de l'appel de la bibliothèque. Le fichier de sortie portera l'extension *.SSSB et contiendra la liste de points par symbole. On peut également envisager la possibilité d'y inclure les informations sur les formes composant le symbole pour pouvoir redessiner ceux-ci.

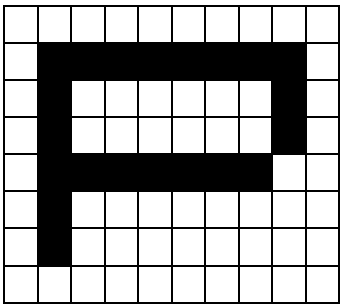
Exemple :



1.2 Calcul de la transformée des distances

A partir d'une image TIFF binaire, la bibliothèque va calculer la transformée des distances de l'image complète. La transformée de distances est une carte indiquant pour chaque pixel, sa distance du pixel noir le plus proche. Le fichier de sortie portera l'extension *.SSTD et sera un fichier binaire contenant chaque pixel avec sa distance associée enregistrée sur 2 octets.

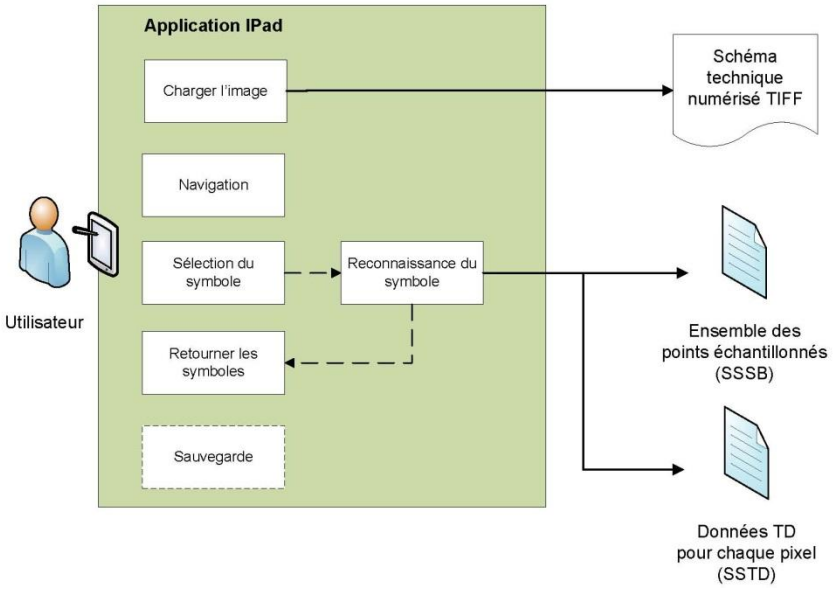
Exemple :



Transformée de distance

2	1	1	1	1	1	1	1	1	2
1	0	0	0	0	0	0	0	0	1
1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	0	1
1	0	0	0	0	0	0	0	1	2
1	0	1	1	1	1	1	1	2	3
1	0	1	2	2	2	2	2	3	4
2	1	2	3	3	3	3	3	4	5

2 Application IPad



L'application IPad va exploiter les données indexées précédemment pour réaliser la reconnaissance de symboles en temps réel. Au lancement de l'application, l'utilisateur va donc charger 3 fichiers :

- L'image TIFF binaire représentant le schéma technique scanné qui sera affiché en arrière-plan dans l'application

- Les points échantillonnés (fichier SSSB) de la base de symboles à reconnaitre
- La transformée de distance (fichier SSTD) correspondant à l'image TIFF chargée.

Une fois ces fichiers chargés, l'utilisateur pourra naviguer librement sur le schéma. Il pourra aussi sélectionner un symbole. Dès le moment de la sélection, l'application va réaliser l'opération de reconnaissance de symbole. La zone sélectionnée sera donc utilisée pour définir sur quelle zone on doit considérer la transformée de distance. A partir de cette zone, on pourra faire l'opération de « matching » consistant à parcourir l'ensemble des points de chaque symbole et récupérer la distance de Manhattan sur la carte de transformée de distance. Par somme de toutes ces valeurs, on pourra déterminer une valeur d'écart-type qui nous permettra de classer les symboles par ordre de similitude. Le meilleur symbole reconnu sera ensuite affiché à l'utilisateur. Si celui-ci ne correspond pas, il pourra prendre le suivant de la liste.

IV – Fonctionnalités du système

1 Bibliothèque d'indexation

1.1 Fonctionnalité 1 : Parsage des fichiers SVG

- **Description** : Récupération des données contenues dans un fichier SVG
- **Données d'entrée** : Le fichier SVG
- **Données de sortie** : L'instance d'une classe Symbole contenant les informations nécessaires
- **Priorité** : Préférable mais pas obligatoire (résultat pouvant être généré extérieurement pour la démonstration).

1.2 Fonctionnalité 2 : Echantillonnage des formes

- **Description** : Calculer une liste de points pour un symbole
- **Données d'entrée** : L'instance d'une classe Symbole (générée par la fonctionnalité 1)
- **Données de sortie** : Une liste des points pour le symbole (écrit dans un fichier SSSB)
- **Priorité** : Préférable mais pas obligatoire (résultat pouvant être généré extérieurement pour la démonstration).

1.3 Fonctionnalité 3 : Squelettisation

- **Description** : Calculer le « squelette » d'un schéma en ne conservant que les pixels correspondant aux formes structurelles sur l'image.
- **Données d'entrée** : Image binaire
- **Données de sortie** : Image binaire sans les pixels inutiles
- **Priorité** : Préférable mais pas obligatoire (résultat pouvant être généré extérieurement pour la démonstration).

1.4 Fonctionnalité 4 : Calcul de la transformée de distance

- **Description** : Calcul de la transformée de distance.
- **Données d'entrée** : Image binaire squelettisée
- **Données de sortie** : La transformée de distance (écrit dans un fichier SSDT)
- **Priorité** : Préférable mais pas obligatoire (résultat pouvant être généré extérieurement pour la démonstration).

2 Application IPad

2.1 Fonctionnalité 5 : Chargement des données

- **Description** : Charger toutes les données nécessaires pour la reconnaissance des symboles. Cela inclus donc l'image TIFF du schéma technique, le fichier de transformée de distance (SSDT) et la liste de points des symboles échantillonnés (SSSB).
- **Données d'entrée** : Fichiers TIFF, SSDT, SSSB
- **Priorité** : Primordiale

2.2 Fonctionnalité 6 : Interactions utilisateur

- **Description** : Implémentation des différentes interactions utilisateur. Ceci inclus la navigation sur l'image, le zoom et la sélection du symbole.
- **Priorité** : Primordiale

2.3 Fonctionnalité 7 : Reconnaissance du symbole

- **Description** : Détermination d'une liste des symboles les plus ressemblants au symbole sélectionné
- **Données d'entrée** : Zone de sélection, symboles échantillonnés, transformée de distance
- **Données de sortie** : Liste des meilleurs symboles reconnus
- **Priorité** : Primordiale

V – Organisation prévisionnelle

1 Découpage du projet en tâches

1.1 Tâche 1 : Etude de la technologie tactile

Cette tâche consiste à étudier les technologies Apple « Objective C » et « Cocoa ». Elle occupera beaucoup de temps en début de projet pour acquérir les bases nécessaires pour réaliser les premiers tests d'implémentation d'interface. Elle se prolongera ensuite tout au long du développement de l'interface.

1.2 Tâche 2 : Proposition d'une interface et démonstration visuelle de la future application

Cette étape consiste à définir l'apparence de l'interface et les interactions de l'utilisateur avec celle-ci. Une documentation sera rédigée expliquant étape par étape comment l'utilisateur interagira avec l'interface.

1.3 Tâche 3 : Etude et implémentation de la bibliothèque d'indexation

Cette tâche comprend tout le développement de la bibliothèque d'indexation incluant donc l'échantillonnage des symboles et le calcul de la transformée de distance. Les livrables seront le moteur de reconnaissance ainsi que la documentation associée.

1.4 Tâche 4 : Implémentation de l'interface de l'application

Cette tâche consiste à implémenter toute la partie interface sans tenir compte pour le moment de la partie reconnaissance de formes. Hormis la reconnaissance, l'interface doit être fonctionnelle.

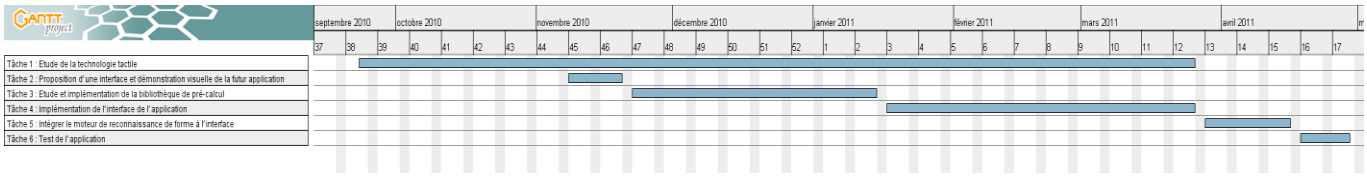
1.5 Tâche 5 : Intégrer le moteur de reconnaissance de forme à l'interface

Cette tâche consiste à intégrer le moteur de reconnaissance de forme à l'interface en utilisant les données générées par la bibliothèque d'indexation.

1.6 Tâche 6 : Test de l'application

Cette tâche consiste à réaliser l'ensemble des tests sur l'application.

2 Planning prévisionnel



VI – Méthodes théoriques

1 Bibliothèque de d'indexation

1.1 Echantillonnage

1.1.1 Parsage du fichier SVG

Pour réaliser l'échantillonnage, nous devons d'abord extraire les informations contenues dans le fichier SVG. Ces fichiers sont structurés à la manière d'un fichier XML. Il sera donc possible d'utiliser une bibliothèque pour le parser et récupérer les informations nécessaires.

Dans notre cas, nos symboles ne nécessiteront pas toutes les formes existantes proposées par le format SVG. On se propose donc de limiter l'extraction à 3 types de données :

- Les lignes (line)
- Les cercles (circles)
- Les chemins (paths)

Les chemins sont des formes plus problématiques car il s'agit d'une ensemble de formes basiques misent les une à la suite des autres. On retrouve donc dans les paths :

- Des lignes
- Des cercles
- Des Béziers quadratiques
- Des Béziers cubiques
- Des ellipses

Chacune de ces formes doivent donc être extraites et listées pour réaliser la partie suivante.

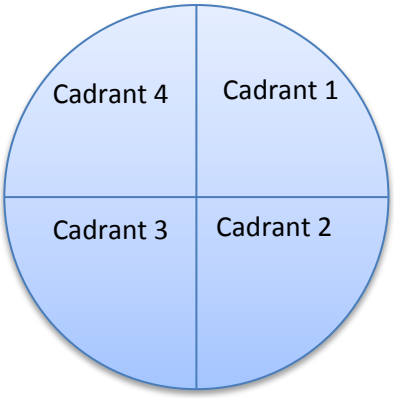
1.1.2 Echantillonnage des formes

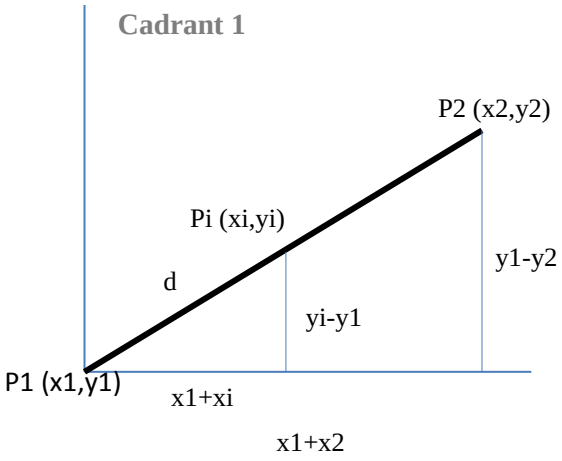
Les formes récupérées précédemment doivent ensuite être échantillonnées au cas par cas.

Les lignes :

Les lignes sont définies simplement par les coordonnées des deux points à l'extrémité de celles-ci. Par utilisation du théorème de Thalès et en travaillant séparément selon le cadrant où se trouve la ligne, on peut facilement retrouver les coordonnées d'un ensemble de points situés sur celle-ci. La méthode est assez similaire au théorème de Bresenham.

Exemple :





Dans cet exemple, on cherche xi et yi, les coordonnées du point Pi. La distance en le point d'origine et le point Pi est calculé avec le théorème de Pythagore. En suivant le théorème de Thalès :

$$\frac{x1 + xi}{x1 + x2} = \frac{d}{distance(P1, P2)} = \frac{yi - y1}{y1 - y2}$$

A partir de cette équation, on peut trouver xi et yi. Cette équation doit être légèrement adaptée pour les 3 autres cadrants.

Les cercles :

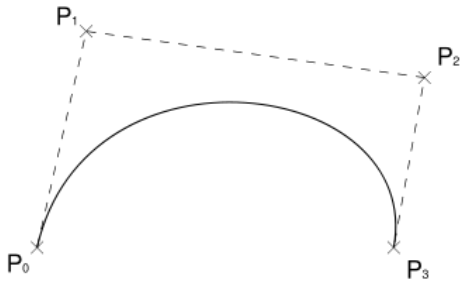
Un cercle est défini par les coordonnées de son centre et son rayon. Dans le cas du cercle, les points seront répartis selon l'angle qui les sépare. Ainsi, pour obtenir les coordonnées d'un point selon son angle, on utilise la formule suivante (avec un angle en radian) :

$$PointX = Centre\ x + (rayon * \cos(angle))$$

$$PointY = Centre\ y + (rayon * \sin(angle))$$

Les Béziers cubiques :

Les Béziers cubiques sont des Béziers à deux points de contrôle. Pour calculer les coordonnées d'un point sur celui-ci, on utilise la formule suivant (extrait de wikipedia.org) :



$$\mathbf{P}(t) = \mathbf{P}_0(1-t)^3 + 3\mathbf{P}_1 t(1-t)^2 + 3\mathbf{P}_2 t^2(1-t) + \mathbf{P}_3 t^3$$

Les Béziers quadratiques :

Similaire aux Béziers cubiques, celui-ci n'a qu'un seul point de contrôle. On pourra utiliser la formule suivante pour déterminer les coordonnées d'un point (extrait de wikipedia.org) :

$$\mathbf{B}(t) = (1-t)^2\mathbf{P_0} + 2t(1-t)\mathbf{P_1} + t^2\mathbf{P_2} \text{ , } t \in [0,1].$$

Les ellipses :

Ce sont des formes complexes dont certaines données ne sont pas directement précisées et doivent être calculés au préalable. Les données fournies par SVG sont :

- $(x1, y1)$, Les coordonnées du point d'origine
- $rx\ et\ ry$, taille des deux axes du l'ellipse, aussi appelées axe majeur et axe mineur
- φ, l'angle de rotation par rapport à l'axe X
- f_A, une valeur binaire, le « large arc flag », indiquant si l'ellipse fera plus de 180 degrés ou moins.
- f_S, une valeur binaire, le « sweep flag », indiquant le sens de rotation de l'ellipse
- $(x2, y2)$, les coordonnées du point final.

Le calcul est réalisé en plusieurs étapes d'après la méthode énoncée sur le site x3.org :

- Calcul de $(x1', y1')$:

$$\begin{pmatrix} x_1{}' \\ y_1{}' \end{pmatrix} = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \cdot \begin{pmatrix} \frac{x_1 - x_2}{2} \\ \frac{y_1 - y_2}{2} \end{pmatrix}$$

- Calcul de (cx', cy') :

$$\begin{pmatrix} c_x{}' \\ c_y{}' \end{pmatrix} = \pm \sqrt{\frac{r_x^2 r_y^2 - r_x^2 (y_1{}')^2 - r_y^2 (x_1{}')^2}{r_x^2 (y_1{}')^2 + r_y^2 (x_1{}')^2}} \begin{pmatrix} \frac{r_x y_1{}'}{r_y} \\ -\frac{r_y x_1{}'}{r_x} \end{pmatrix}$$

On prend le signe positif si $f_A \neq f_S$, sinon, le signe négatif.

- Calcul de (cx, cy) correspondant au centre de l'ellipse :

$$\begin{pmatrix} c_x \\ c_y \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \cdot \begin{pmatrix} c_x{}' \\ c_y{}' \end{pmatrix} + \begin{pmatrix} \frac{x_1 + x_2}{2} \\ \frac{y_1 + y_2}{2} \end{pmatrix}$$

- Calcul de l'angle de départ $\theta 1$, et de l'angle total de l'ellipse $\Delta\theta$:

Sachant que l'angle entre deux vecteurs U et V est calculé de la manière suivante :

$$\angle\left(\vec{u}, \vec{v}\right) = \pm \arccos \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \ \|\vec{v}\|}$$

, on fait le calcul suivant :

$$\Delta\theta \equiv \angle\left(\begin{pmatrix} \frac{x_1{}' - c_x{}'}{r_x} \\ \frac{y_1{}' - c_y{}'}{r_y} \end{pmatrix}, \begin{pmatrix} \frac{-x_1{}' - c_x{}'}{r_x} \\ \frac{-y_1{}' - c_y{}'}{r_y} \end{pmatrix}\right) \bmod 360°$$

A partir de ces nouvelles données, on peut calculer les coordonnées d'un point sur l'ellipse selon l'angle où il se trouve :

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \cdot \begin{pmatrix} r_x \cos\theta \\ r_y \sin\theta \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$$

1.2 Calcul de la transformée de distance

On découpe cette étape en deux parties : la squelettisation et le calcul de la transformée de distance.

1.2.1 Squelettisation

Les schémas numérisés ne sont pas directement exploitables. Les tracés auront toujours une épaisseur et des imperfections. La squelettisation va nous permettre de lisser les formes pour obtenir uniquement la forme structurelle que nous pourrons exploiter.

Pour se faire, nous utiliserons l'algorithme de squelettisation de Tohmé. Celui-ci consiste à parcourir l'image binaire et à vérifier la configuration de chaque pixel. Si le pixel parcouru correspond à un des 16 cas énoncé par Tohmé, le pixel est éliminé (passe à 0). Les 16 cas de Tohmé sont les suivants :

X	1	X
1	P=1	0
X	1	X

X	0	X
1	P	1
X	1	X

X	1	X
0	P	1
X	1	X

X	1	X
1	P	1
X	0	X

0	0	0
0	P	0
X	1	1

0	0	0
0	P	0
1	1	X

1	1	X
0	P	0
0	0	0

X	1	1
0	P	0
0	0	0

0	0	X
0	P	1
0	0	1

0	0	1
0	P	1
0	0	X

X	0	0
1	P	0
1	0	0

1	0	0
1	P	0
X	0	0

X	0	0
1	p	0
X	1	X

0	0	X
0	p	1
X	1	X

X	1	X
0	P	1
0	0	X

X	1	X
1	P	0
X	0	0

1.2.2 Calcul de la carte de transformée de distance

Ce calcul se fait en parcourant tous les pixels de l'image en deux passages. Le premier parcours se fait de gauche à droite, de haut en bas. Pour calculer la distance associée aux pixels, on regarde la valeur du voisin au-dessus et de celui à gauche. On utilisera la valeur de distance la plus petite et on lui ajoute 1. On répète cette opération pour le reste du parcours.

Le second parcours se fera de droite à gauche, de bas en haut. On fait la même opération qu'au premier parcours mais cette fois-ci, on s'intéressera à la distance associé des voisin du bas et de droite.

2 Application IPad

2.1 Interaction utilisateur

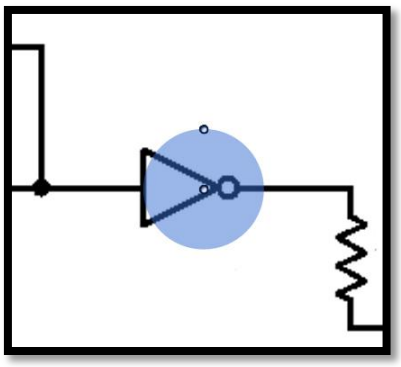
L'application développée est multi-touch utilisant les fonctionnalités de la technologie IPad. Avant de débuter le développement de l'interface, il est nécessaire de définir un modèle.

Il est ainsi nécessaire de bien définir les interactions utilisateur afin de veiller à respecter les standards existant pour les manipulations sur IPad et pour s'assurer que chaque commande soit distincte pour ne pas entrainer une action non voulu par l'utilisateur. Voici les différentes possibilités d'interactions proposées et celle qui a été choisie finalement.

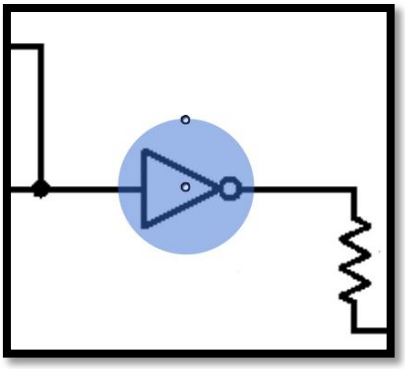
2.1.1 Solution 1

Déplacements : Faire glisser le schéma avec un seul doigt.
Zoomer/dézoomer : Eloigner ou rapprocher deux doigts sur l'écran
Sélection : Elle se fait en plusieurs étapes :

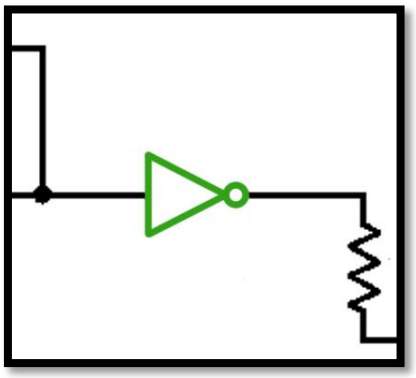
1. Faire une double tape d'un doigt sur le symbole à reconnaitre. Un cercle bleu va apparaitre à l'endroit où la double tape a été faite. Ce cercle bleu représente la zone sélectionnée. Dès l'apparition de ce cercle, le moteur de reconnaissance retourne immédiatement les symboles reconnus.



2. Dans le cas où la reconnaissance s'est mal effectuée car le cercle n'est pas bien placer, il faut déplacer les petit rond se trouvant au centre et sur une extrémité du cercle bleu en les faisant glisser. Le rond sur centre permet de situer le centre sur cercle et le cercle sur l'extrémité permet de définir le rayon du cercle.



3. Quand le bon symbole apparait dans la barre des symboles reconnus, après avoir sélectionné celui qui correspond au symbole sur l'image, un aperçu sur symbole reconnu se superposera sur le schéma technique.



Rotation : Faire glisser le rond de l'extrémité du cercle dans le sens horaire ou antihoraire autour du rond central.

Avantage : Toutes ces manipulations sont conformes aux normes de manipulation de la plupart des applications IPad/IPod, elles sont donc instinctives.
Inconvénient : Cette solution exige beaucoup de manipulations qui doivent être répétées pour tous les symboles (> 5 actions). Elle est donc trop couteuse en temps.

2.1.2 Solution 2

Déplacements : Faire glisser le schéma avec un seul doigt.
Zoomer/dézoomer : Eloigner ou rapprocher deux doigts sur l'écran
Sélection : Faire une tape maintenue sur l'écran pendant 1 seconde, puis placer un second doigt. Un cercle apparaitra, les deux doigts représentant le diamètre. Il suffit d'ajuster le cercle au symbole.
Rotation : Pendant la sélection (avec 2 doigts) placer un 3^{ième} doigt sur l'écran et le faire glisser vers le haut ou vers le bas.

Avantage : Peu de manipulations nécessaires
Inconvénient : Risque de conflit avec la fonction zoom si l'utilisateur ne maintien pas son doigt assez longtemps. De plus, cette attente ralentie inutilement le processus.

2.1.3 Solution 3

Déplacements : Faire glisser le schéma avec un seul doigt.

Zoomer/dézoomer : Prévoir deux boutons supplémentaires sur l'interface '+' et '-' pour zoomer et dézoomer.
Sélection : Placer deux doigts sur l'écran. Un cercle apparaitra, les deux doigts représentant le diamètre. Il suffit d'ajuster le cercle au symbole.
Rotation : Pendant la sélection (avec 2 doigts) faire tourner la main.

Avantage : Manipulations simples et rapides
Inconvénient : Incompatibilité avec la fonction de zoom de la plupart des applications IPad/IPod.

2.1.4 Choix de la solution

Bien qu'il soit important de respecter les normes de manipulation Apple, le temps de manipulation est la partie la plus crucial de ce type d'application. Pour cette raison, la solution 3 semble être le meilleur choix. Certes, la manipulation pour la sélection est normalement réservée au zoom, cependant, l'utilisateur va ajuster le zoom uniquement à l'ouverture du document, il n'est donc pas nécessaire d'en faire une manipulation rapide. La sélection au contraire doit être opérée pour chaque symbole.

2.2 Reconnaissance de symbole

La reconnaissance de symbole se fait lorsque l'utilisateur sélectionnera un symbole. La reconnaissance s'effectuera en récupérant les coordonnées des zones sélectionnées sur l'image BITMAP. Elles seront ensuite utilisées pour déterminer la partie de la transformée de distance qui sera utilisée. On aura donc deux types de données : la transformée de distance partielle et la liste des points échantillonnées par symbole. On va donc mettre en correspondance ces deux données pour calculer un écart-type par symbole.

Pour cela, nous allons récupérer une distance pour chacun des points échantillonnées. En effet, par transformée affine, nous allons pouvoir adapter les points échantillonnées à la zone de sélection. Puis, en récupérant la distance correspondante à chacun des points, on va pouvoir calculer une moyenne et un écart-type pour chaque symbole.

Pour terminer, on compare tous ces écarts-type et on aura une liste des symboles par ordre de correspondance.

4	3	2	3	4	5	4	3	2	3	4
3	2	1	2	3	4	3	2	1	2	3
2	1	0	1	2	3	2	1	0	1	2
2	1	0	1	2	3	2	1	0	1	2
2	1	0	1	2	3	2	1	0	1	2
2	1	0	1	2	2	2	1	0	1	2
2	1	0	1	1	1	1	1	0	1	2
2	1	0	0	0	0	0	0	0	1	2
2	1	0	1	1	1	1	1	0	1	2
2	1	0	1	2	2	2	1	0	1	2
2	1	0	1	2	3	2	1	0	1	2
2	1	0	1	2	3	2	1	0	1	2
2	1	0	1	2	3	2	1	0	1	2
3	2	1	2	3	4	3	2	1	2	3

<u>Points rouges :</u>
Somme = (3*2)+(2*14)+(1*6) = 40
Nombre de points = 22
Moyenne = 1,82
Ecart-type = 0,59

<u>Points bleu</u> :
Somme = (3*2)+(2*4)+(1*6) = 20
Nombre de points = 15
Moyenne =1,34
Ecart-type = 0.98

Donc le symbole rouge est le plus proche.

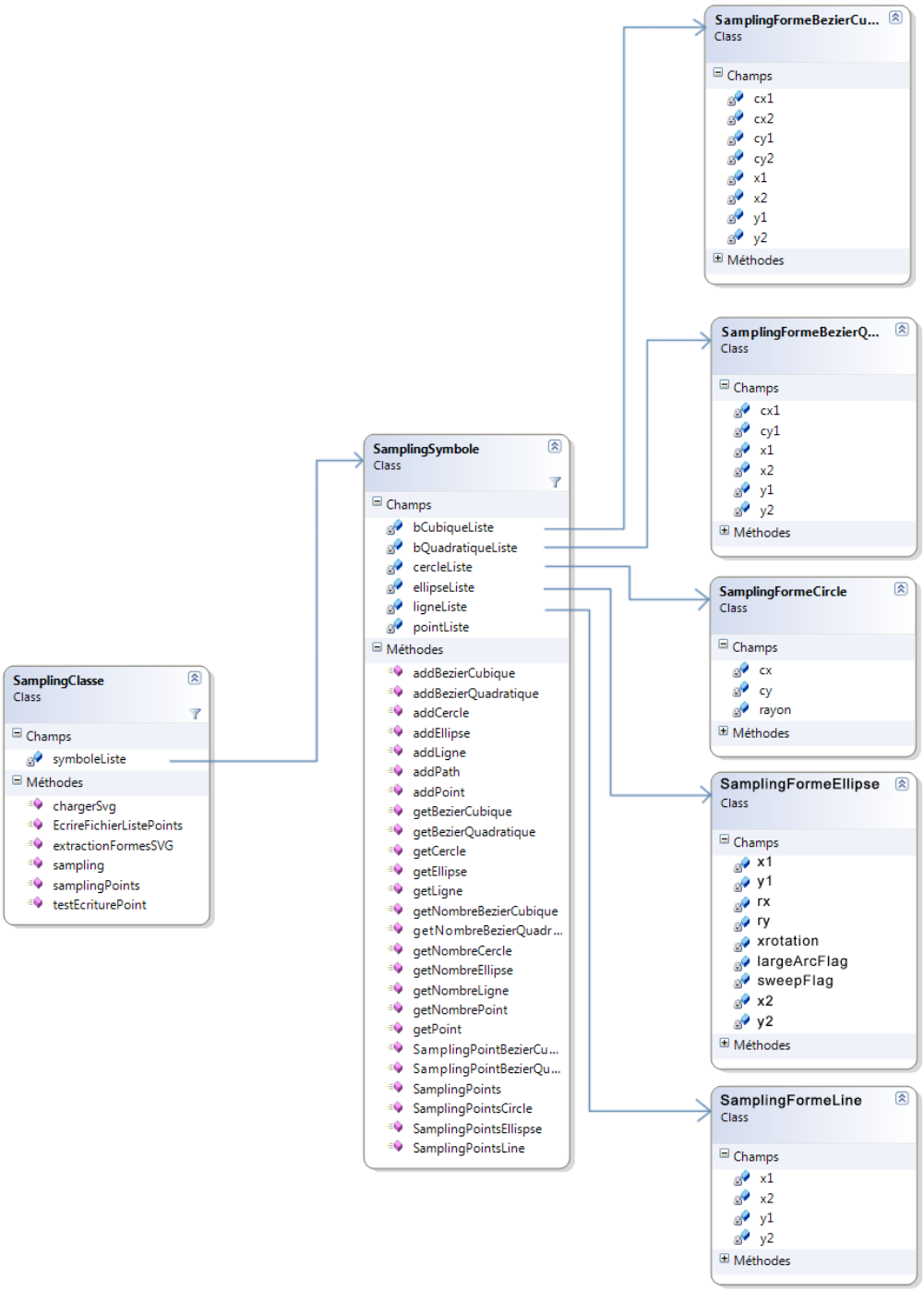
VII – Implémentation

1 Bibliothèque d'indexation

Les deux parties de la bibliothèque d'indexation (transformée de distance et échantillonnage) sont complètement indépendantes. La bibliothèque a été entièrement écrite en C++.

1.1 Echantillonnage

1.1.1 Diagramme de classe



1.1.2 Utilisation

Pour réaliser une opération de sampling, il faut créer une instance de la classe « SamplingClasse ». A partir de celle-ci, on va pouvoir utiliser la méthode « sampling » en précisant en paramètre le dossier contenant les fichiers SVG, la distance entre les points à échantillonner et une valeur booléenne indiquant si on souhaite que les sous dossiers soient pris en compte pour récupérer les fichiers SVG. En cas de succès, la fonction retourne 1, sinon, elle retourne 0. On peut ensuite écrire le fichier SSSB avec la méthode « EcritureFichierPoint ».

```
SamplingClasse   monInstance ;

int ret = monInstance.sampling("c:/MesSVG/", 8, false) ;
if( !ret) printf("Une erreur s'est produite pendant l'échantillonnage") ;

ret = monInstance.EcritureFichierListePoint("c:/fichier.sssb") ;
if( !ret) printf("Une erreur s'est produite pendant l'écriture") ;
```

1.1.3 Description des classes

1.1.3.1 SamplingClasse

Il s'agit de la classe principale qui devra être instanciée. Elle va permettre de récupérer l'ensemble des fichiers SVG qui se trouve dans le dossier spécifié, et de générer des instances de la classe « SamplingSymbole » qui seront listées dans un vecteur. Elle permet également d'écrire les fichiers SSSB contenant les données échantillonnées.

1.1.3.2 SamplingSymbole

Cette classe représente chaque symbole extrait à échantillonner. Elle contient des vecteurs d'éléments graphiques composant les symboles. Les points échantillonnés sont stockés dans le vecteur pointListe.

Elle contient également un ensemble de méthodes pour ajouter et récupérer ces éléments graphiques, mais aussi des méthodes pour échantillonner ceux-ci.

1.1.3.3 SamplingForme…

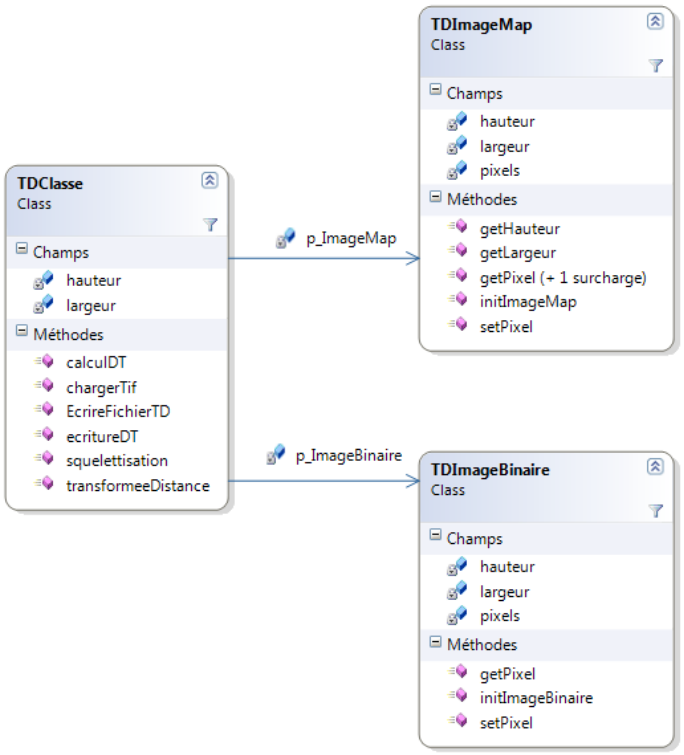
Ce sont des classes représentant l'ensemble des éléments graphiques qui seront échantillonnés. On retrouve donc les lignes, les cercles, les Béziers cubiques et quadratiques et les ellipses.

1.1.4 Bibliothèque utilisée

Pour réaliser le parsage du fichier SVG, j'ai utilisé la bibliothèque « tinyxml » qui est gratuite et libre. Elle permet de perser les fichiers XML en général mais fonctionne aussi sur les fichiers SVG qui respectent les même normes.

1.2 Transformée de distance

1.2.1 Diagramme de classe



1.2.2 Utilisation

Pour calculer la transformée de distance, il suffit d'instancier la Classe TDClasse, puis d'appeler la méthode « transformeeDistance » en spécifiant comme argument le chemin vers le fichier TIFF binaire. Cette méthode va charger l'image, calculer la squelettisation, et calculer la transformée de distance.

```
TDClasse   monInstance ;

int ret = monInstance.transformeeDistance("c:/fichier.tiff");
if( !ret) printf("Une erreur s'est produite pendant le calcul de la transformée de distance") ;

ret = monInstance.EcritureFichierTD("c:/fichier.ssdt") ;
if( !ret) printf("Une erreur s'est produite pendant l'écriture") ;
```

1.2.3 Description des classes

1.2.3.1 TDClasse

Il s'agit de la classe principale qui doit être instanciée. Elle permet de récupérer un fichier TIFF binaire pour faire toutes les opérations de calcul : squelettisation et transformée de distance. Pour finir, on va pouvoir écrire le fichier SSDT.

1.2.3.2 TDImageBinaire

Cette classe contient l'image binaire chargée du fichier TIFF.

1.2.3.3 TDImageMap

Cette classe contient les distances associées à chacun des pixels de l'image.

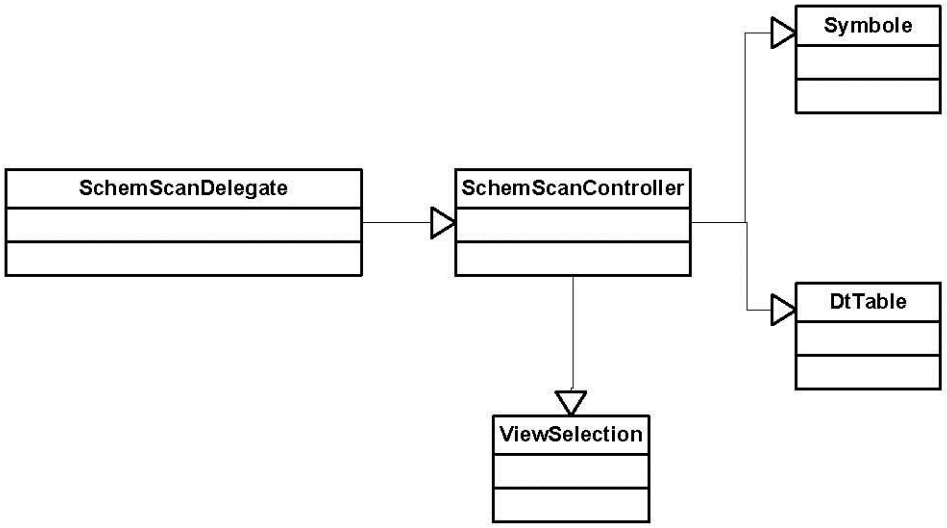
1.2.4 Bibliothèque utilisée

Pour le chargement de l'image TIFF binaire, j'ai utilisé la bibliothèque d'Adobe, « libtiff » qui est gratuite et libre.

2 Application IPad

L'application IPad a été écrite avec Xcode qui est optimisé pour l'Objective C mais permet également l'utilisation du C et C++.

2.1 Diagramme des classes



2.2 Description des classes

2.2.1 SchemScanDelegate

Il s'agit de la classe principale de l'application. Elle va charger le contrôleur à son ouverture.

2.2.2 SchemScanController

C'est le contrôleur de l'application. Il va assurer la gestion de la vue, mais aussi la gestion des données et de l'interface.

2.2.3 ViewSelection

C'est la vue de l'application. A chaque appel, elle va se recharger pour se mettre à jour. Il contient une liste d'éléments que l'on met à jour et qu'elle doit afficher.

2.2.4 DtTable

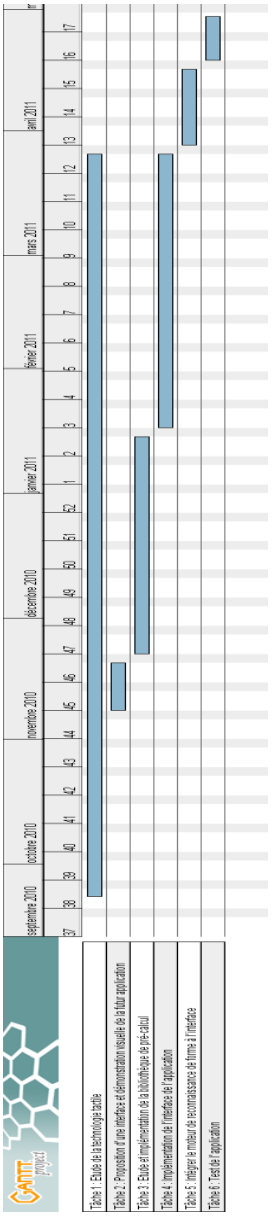
Classe contenant les données de la transformée de distance.

2.2.5 Symbole

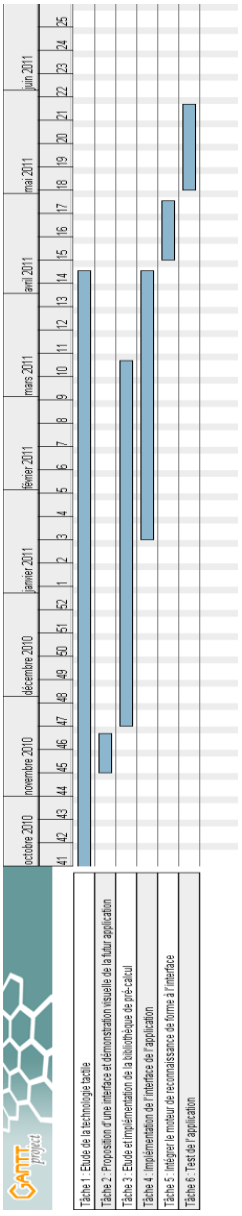
Classe contenant la liste des points par symbole.

VIII – Avancement du projet aujourd'hui

1 Comparaison plannings prévisionnels et effectifs



Planning prévisionnel



Planning effectif

Par rapport au planning prévisionnel, quelques retards ont eu lieu. En effet, de nouvelles modifications ont été nécessaires sur la bibliothèque d'indexation par rapport à la première version terminée dans les temps. Pour cette raison, j'ai commencé le développement de l'interface en parallèle de la finalisation de la bibliothèque sur mon temps libre pour rattraper le retard. L'apprentissage des technologies Apple a aussi nécessité plus de temps que prévu et j'ai continué cette apprentissage tout au long du développement de l'interface. Mais s'agissant d'un langage de

développement nouveau pour moi, le développement de l'application a pris bien plus de temps que prévu.

2 Etat d'avancement

Aujourd'hui, le développement de la bibliothèque d'indexation est achevé, cependant l'application IPad est encore en développement. Sur celle-ci, l'ensemble des classes et méthodes sont déjà implémentées. L'application est donc en cours de finalisation et sera testée sous peu sur IPad.

A l'issu de ce test, une documentation de test sera rédigée réunissant les tests réalisés sur la bibliothèque d'indexation et l'application IPad. Les documentations doxygene seront également remisent avec les codes sources.

Conclusion

Ce projet a été pour moi l'occasion d'étudier les technologies Apple, mais aussi les technologies tactiles en général. En effet, ce sont des technologies qui sont de plus en plus utilisées dans tous les domaines. J'ai également pu travailler sur la problématique tu temps réel, et sur les possibilités offertes pour réduire la complexité des algorithmes.

Il y a effectivement eu des retards au cours de ce projet, et malgré le fait que l'application soit encore en developpement, la version finale de logiciel de démonstration SchemScan pourra être présentée à Pos-Industry sous peu.

Je tiens à remercier Mathieu Delaladre pour avoir proposé ce sujet mais aussi pour l'aide précieuse qu'il m'a apporté tout au long de son développement.

Bibliographie

- http://www.wikipedia.org
  - Site utilisé pour retrouver les formules de calcul de points sur les courbes de bézier

- http://www.libtiff.org/
  - Site proposant le téléchargement de la bibliothèque libtiff ainsi que sa documentation

- http://www.grinninglizard.com/tinyxml/
  - Site proposant le téléchargement de la bibliothèque tinyxml ainsi que sa documentation

- http://perso.telecom-paristech.fr/~maitre/BETI//skel_4_8/algo.html
  - Site présentant l'algorithme de Tohmé pour la squelettisation d'images binaires

- http://www.w3.org/TR/SVG/
  - Documentation détaillée de W3 sur les fichiers SVG

- http://mathieu.delalandre.free.fr/projects/sesyd/index.html
  - Exemples de fichiers SVG et TIFF utilisées pour les tests

- http://developer.apple.com/devcenter/ios/index.action
  - Site officiel d'Apple pour le développement IOS fournissant les documentations, des codes exemples et des manuels