Barast Ludivine

Chastel Adrien

Estival Elodie

Gaillard Christophe

**Editeur de document collaboratif pour table tactile**

Entreprise Maskott

I.U.T. Clermont-Ferrand I, Département Imagerie Numérique

Année universitaire (2013-2014)

**Remerciements**

Merci à l’entreprise Maskott pour leur implication dans ce projet.

Merci à Delphine Huguel pour l’aide indispensable et constante qu’elle nous a fournis.

Merci à Adélaïde Albouy-Kissi.

**Résumé**

L’objectif de ce projet était de réaliser un éditeur de document collaboratif sur table tactile. Cela consistait à permettre à plusieurs utilisateurs autour de la table de proposer leurs idées que se soit sous forme de texte ou d'image et de les mettre en place sous forme de cases d’idées. Chaque utilisateur devait pouvoir de son côté relier les cases, les séparer, en ajouter et en supprimer dans le but de créer des arbres d’idées qui permettraient ainsi d’organiser des données pour créer un projet. Ceci à la manière d’un logiciel de mind mapping.

Ce projet doit utiliser les capacités de la table tactile, les différents types d’interactions que chaque utilisateur peut avoir avec elle : le doigt, la paume de la main ou les tags, l’utilisation simultanée du logiciel par plusieurs personnes et conserver une bonne ergonomie de l’application. Notamment éviter que le trop grand nombre d’actions successives entraine le désordre sur la page.

Pour réaliser cette application, nous avons pris en compte les exemples déjà existants de logiciels de mind mapping pour savoir comment organiser notre travail et les différentes possibilités que nous avions pour sa mise en place. Nous avons adaptées ces informations aux capacités de la table tactile et fait face à quelques problèmes notamment l’apparition simultanée de plusieurs claviers sur la table.

L’interface permet de créer un "Node" à l’appui du doigt sur la table. On a alors le choix du format de l’idée : image ou texte. Pour chaque case, on peut créer un fils qui sera relié, la supprimer avec ses fils ou la séparer de son parent. Pour le "NodeText", on peut changer la couleur et éditer le texte à l’aide d’un clavier virtuel. Pour le "NodeImage, on a accès à un chargement d’images à partir d’un dossier du projet.

Chaque case peut être déplacée sur l’écran. Lors de l’écriture, on peut bloquer ces mouvements pour voir s’afficher une barre de défilement.

**Summary**

This project’s goal was to create a collaborative text document editor for the Surface table. We had to allow many users all around the table to give their ideas with a text or an image and put them into an idea’s tree. Any user can link the boxes, disconnect them, add or delete a node to create an idea’s tree that will organize data to create a project. This is just like a mind mapping software.

This project has to use the Surface table’s abilities, the different kinds of interactions that any user can have with this table : finger, hand’s palm or tags, software’s simultaneous use by many people, and keep good application ergonomic. Especially prevent that the number of successive actions results a disoder on the page.

To create this application, we used the existing mind mapping software’s examples as model to know how to organize our work and the different possibilities we had for achieve this project. We adapted this data to the Surface table’s capacities and faced a few problems like the creation of several keyboards in the same time on the table.

The interface allows to create a node when you press your finger on the table. Then, you can choose your idea’s format : image or text. For each box, you can create a son linked to it, delete it or disconnect it to its parent. For the nodetext, you can change the color and edit the text with a virtual keyboard. For the nodeimage, you can choose the images and change them from a folder into the project.

Any box can be moved on the screen.When you write text, you can block these moves to see and use a scrollbar. With a tag, you can save your idea’s tree to read or modify it.

Table des matières

[Introduction 1](#_Toc382532332)

[I. Contexte. 2](#_Toc382532333)

[1.1. Entreprise 2](#_Toc382532334)

[1.2. Tactiléo 3](#_Toc382532335)

[1.3. PixelSense 4](#_Toc382532336)

[II. Partie générale 5](#_Toc382532337)

[2.1. Introduction 5](#_Toc382532338)

[2.2. Mind Mapping 6](#_Toc382532341)

[2.3.1. WPF 7](#_Toc382532345)

[2.3.2. XAML 7](#_Toc382532346)

[2.3.3. C# 8](#_Toc382532347)

[2.3.4. Scatter View 8](#_Toc382532355)

[2.3.5. Entrées 9](#_Toc382532356)

[2.3.6. Outils du SDK 9](#_Toc382532357)

[2.4. Eléments du programme 10](#_Toc382532358)

[2.4.1. Nodes 10](#_Toc382532359)

[2.4.2. Fonctionnalités du NodeText 10](#_Toc382532360)

[2.4.3. Fonctionnalités du NodeImage 11](#_Toc382532361)

[2.4.4. Relations entre Nodes 11](#_Toc382532362)

[2.4.5. Gestion de fichiers 12](#_Toc382532363)

[2.4.5.1. Enregistrement 12](#_Toc382532364)

[2.4.5.2. Ouverture 12](#_Toc382532365)

[2.4.6. Clavier 13](#_Toc382532366)

[III. Partie technique. 14](#_Toc382532367)

[3.1. Introduction 14](#_Toc382532368)

[Conclusion 15](#_Toc382532369)

# Introduction

Durant notre formation de DUT informatique, il nous a été demandé de choisir et mener à bien un projet tuteuré en liaison avec une entreprise extérieure. Différents sujets nous ont alors été proposés en fonction de l’orientation que nous préférions suivre au quatrième semestre : programmation 3D ou synthèse d’images. Notre groupe ayant une affinité pour la première de ces orientations, nous avons porté notre choix sur le projet de réalisation de document collaboratif sur table tactile proposé par l’entreprise **Maskott** au Puy-en-Velay.

Ces sujets avaient pour objectif de nous faire travailler en groupe de quatre ou cinq, en relation avec une entreprise et sur des supports différents de ceux dont nous avions l’habitude. Cette situation nous a permis de nous rapprocher de notre futur travail en entreprise en nous faisant suivre et réaliser les différentes étapes de la création d’un projet en relation avec les professionnels.

Celui-ci avait pour but la création d’une application d’édition de **document collaboratif** sur **table tactile** devant être capable de permettre aux différents acteurs d’une réunion autour de la table de pouvoir proposer leurs **idées** et les **associer** entre-elles. Ces idées pouvant se présenter sous la forme de textes, images, vidéos ou sons.

Pour réaliser ce projet, plusieurs choix nous avaient été proposés : réaliser un traitement de texte dans le style de Microsoft Word, ou bien une présentation dans le type Power Point, ou encore un logiciel de documents collaboratifs de **mind mapping** comme Freemind. Nous avons opté pour cette dernière proposition car elle nous paraissait la plus évidente au vue du sujet proposé.

En effet, il s’agit d’organiser des idées et ce format propose de créer des idées sous forme de portions indépendantes que l'on peut ensuite **relier** ou séparer les unes des autres afin de créer des **arbres d’idées**.

L’autre point à prendre en compte a été le fait que la table tactile permet un **travail collaboratif** : plusieurs personnes peuvent interagir en même temps avec la table chacun de son côté. Il a fallu éviter les débordements que pouvait provoquer cette liberté d’utilisation et permettre de l’utiliser au mieux pour l’application.

# Contexte.

## Entreprise

Ce projet nous a été proposé en collaboration avec la société de développement informatique Maskott au Puy en Velay. Cette entreprise a été fondée en 2004 et conçoit, réalise et vend des solutions pédagogiques et multimédias à des établissements scolaires et structures de formation.

Cette société offre des conseils pour la planification de projets d’équipements multimédia et pour leur mise en œuvre, pour l’enseignement. Elle propose une solution globale incluant l’ensemble des équipements nécessaires à l’installation : des serveurs aux logiciels pédagogiques, en passant par les tableaux interactifs.

Maskott propose d’autres services comme la rédaction du cahier des charges, l’installation du matériel et la formation des utilisateurs (principalement des enseignants).

Cette société est donc principalement orientée sur l’installation et la création de solutions informatiques pour l’enseignement et l’intégration de matériel de technologie moderne dans un cadre pédagogique.

Leur première expérimentation tablette en école primaire, en 2010, a aussi été la première de France. Cette expérience leur a permis de se développer, mais aussi de mettre en place le consortium Tactiléo en association avec de nombreux collaborateurs importants.

## Tactiléo

Pour ce qui est de la recherche informatique, l’entreprise Maskott est également à la tête du consortium Tactileo, qui réunit à la fois l’IUT du Puy en Velay, le Commissariat à l’Energie Atomique (CEA), l’Institut Français d’Education - Ecole Normale Supérieure de Lyon (IFE), Schuch Production, l’IGN et Microsoft.

Ce projet consiste à mettre en place au sein d’une classe un « écosystème pédagogique » qui réuni et synchronise les interfaces tactiles de cette classe. Ils traitent donc aussi bien de la circulation des informations entre les différentes interfaces que de l’ergonomie de celles-ci ou de leur organisation au sein de la salle de classe. Cette organisation concerne les tables tactiles, tableaux interactifs et autres paillasses numériques.

Cette association vise à la composition d’une véritable classe immersive qui puisse stimuler les apprentissages, notamment ceux des enfants, qui apprennent plus facilement face à une pédagogie interactive comme le proposent les interfaces tactiles.

Tactileo s’adresse également à toutes les catégories d’élèves, de l’école primaire jusqu’à l’université.

A long terme, le projet devrait toucher plus de 5000 élèves et étudiants répartis dans différentes académies. Il constitue l’avenir de l’éducation avec du matériel plus moderne et plus innovant face à une nouvelle population d’étudiants et d’élèves plus que jamais sensible aux nouvelles technologies.

Ces processus de formations permettront également d’adapter les élèves, lors de leur formation, aux outils auxquels ils seront forcément confrontés dans leur vie future que se soit dans leur quotidien comme dans le cadre de leur entreprise.

Les interfaces utilisées dans le projet Tactiléo demandent également aux étudiants de travailler en groupe sur une même problématique et non chacun sur sa propre feuille ou son propre ordinateur. Ceci permet l’apprentissage d’une coordination de groupe.

## PixelSense

Le programme que l’on nous a demandé de réaliser doit donc être utilisable sur l’un des outils de prédilection de l’entreprise Maskott, la table tactile, et plus précisément la table Microsoft SUR40. Il nous a donc fallu utiliser cette interface et programmer en conséquence avec le SDK qui nous a été fourni.

Nous avons ensuite pu tester nos résultats sur l’une des tables mises à notre dispositions, l’une dans les locaux de l’entreprise Maskott et l’autre au sein l’iut.

Microsoft PixelSense ou Microsoft Surface est une table tactile conçue par Microsoft. Elle se présente à l'utilisateur comme une table dont le dessus est constitué d'une surface doté d’un affichage tactile multitouch de 30 pouces.

La table tactile permet à plusieurs personnes d’interagir en même temps avec elle. Les interactions peuvent se faire soit au doigt, soit à l’aide de blocs ou de tags.

Il existe deux versions de cette table tactile :

* Microsoft Surface 1.0, basée sur le système d’exploitation de vista.
* Microsoft SUR40 qui possède une plus grande table et la technologie PixelSense de Microsoft.

La table tactile peut recevoir plus de 50 contacts simultanément et les traiter tous en même temps, ce qui permet à plusieurs personnes de travailler et réfléchir en même temps sur le même outil interactif.

La table peut être utilisée comme une table, un mur et peut communiquer avec d’autres périphériques comme un clavier ou une tablette.

La table tactile SUR40 possède de base plusieurs applications qui permettent aux utilisateurs de tester les différentes fonctionnalités et interactions disponibles.

1. Partie générale

## Introduction



Lors de notre projet tuteuré, nous avons donc dû créer un logiciel d’édition de document collaboratif sur cette table tactile.

Pour cela, nous avons déjà pris connaissance des différentes possibilités que nous avions lors d’une réunion avec les différents protagonistes du projet, c’est-à-dire Mme Albouy-Kissi et les membres de l’entreprise Maskott concernés par notre projet. Nous avons alors étudié ces différents choix pour nous tourner vers celui qui semblait le plus approprié à ce type de document, la création d’un logiciel basé sur le mind mapping.

Ensuite, nous avons pris connaissance de la table tactile, de ses caractéristiques et de son utilisation, et des différents outils dont nous avions besoin pour réaliser ce projet. Ainsi, nous avions besoin des différents éléments de développement nécessaires pour ce type d’interface mais aussi d’un outil de simulation des outils de la table tactile car nous ne pouvions pas coder directement sur la table.

Le programme prend la forme d’un espace sur l’écran où l’utilisateur peut créer des cases d’idées appelées « Nodes » et les relier les unes aux autres. Il peut créer deux types de Nodes : des Nodes contenant du texte, appelées « NodeText » et des Nodes contenant des images appelées « NodeImage ». Ces bulles sont construites sous la forme d’arbres d’idées reliées et liables entre elles ayant des spécificités particulières selon leur contenu.

L’application utilise plus particulièrement l’interaction de la table tactile du « finger », qui est plus précis notamment pour l’utilisation du clavier virtuel. Elle utilise aussi les tags pour la sauvegarde.

## Mind Mapping

Une carte heuristique, carte des idées ou plus connu sous son nom anglo-saxon Mind Mapping est une méthode utilisée pour l’organisation de réflexions et d’informations de façon visuelle permettant de suivre aisément le cheminement associatif de la pensée.

Ce concept fut originellement pensé par Aristote. Cependant, ce fut Tony Buzan, un psychologue qui le théorisa dans les années 1970 lors de ses réflexions sur le cerveau et plus précisément sur la spécialisation hémisphérique de ce dernier.

Proche du brainstorming, il se révèle cependant organisé et permet ainsi de donner un aspect logique à une suite d’idées. Sa façon de représenter l’information de manière graphique et spatiale diffère de la représentation linéaire traditionnellement utilisée dans l’apprentissage.

En effet, notre cerveau n’empile pas les idées, mais les associe les unes aux autres créant ainsi une arborescence. C’est selon ce schéma et ce concept de ramification que s’établit le Mind Mapping.

Le support utilisé pour sa création manuelle est une feuille blanche au format paysage. Ce support vierge permet une vision périphérique de l’analyse sans pour autant être distrait par un quelconque marquage.

Au centre se place le concept de base de la réflexion, également appelé le cœur. Ce sujet est le plus souvent représenté par une image, mais il doit également être le plus précis possible afin d’orienter favorablement la suite de la pensée.

À ce nœud initial sera ajouté les branches. Ces premières branches dites principales partant du cœur correspondent aux thèmes ou rubriques dérivées et illustrant cette idée initiale. Enfin à chacune de ces branches pourront être ajoutées des branches secondaires comportant à leurs tours des données périphériques qui étoffent les précédentes.

C’est sur ces branches, principales comme secondaires, que seront écrites les informations. Le Mind Mapping se voulant concis, seuls des mots-clés seront inscrits sur ces branches. Peuvent également s’ajouter des images ou pictogrammes pour faciliter la représentation.

Enfin sera ajouté de la couleur à ce schéma. Sa valeur n’est pas uniquement décorative mais également esthétique. Apporter une couleur à une ramification l’associe aux sensations qu’elle évoque et permet une meilleure mémorisation de l’information.

2. 2. **Eléments de l’environnement de développement**

## WPF

Le WPF pour Windows Presentation Fondation est une norme graphique de Microsoft .NET 3.0. Son intégration du langage XAML lui autorise une utilisation proche d’une page HTML pour ses utilisateurs. Il permet également aux développeurs de pouvoir séparer le graphisme de l’application de ses contrôles.

Étant une surcouche logicielle à Direct X, il fournit tous les éléments nécessaires à la conception d’une interface utilisateur. Ces éléments étant entièrement vectoriels aussi bien pour le dessin que pour le texte permettent une manipulation sans soucis de pixellisation lors des redimensionnements, par exemple au cours de l’utilisation de l’application.

## XAML

Le XAML pour eXtensible Application Markup Language est un langage déclaratif développé pour la conception d’interface pour les besoins des systèmes d’exploitation de Microsoft. Étant basé sur le langage XML, il répond lui aussi à un certain schéma d’élaboration et une structure spécifique de ses fichiers.

Il permet aussi aux développeurs d’élaborer aisément leurs applications client de manière similaire à une simple application destinée au WEB. Il permet également la manipulation d’objets graphiques vectoriels en 2D ou 3D.

## C#

Le C# ou C Sharp est un langage de programmation orienté objet à typage fort. Créé par la société Microsoft et principalement par un de ses employés, déjà créateur du langage Delphi.

Il a été réalisé pour refléter au mieux l’architecture de la plate-forme Microsoft .NET. Faite pour mettre en avant le maximum des capacités de cette dernière, toutes les applications .NET s’en retrouve extrêmement dépendante.

Syntaxiquement, on peut dire qu’il est proche du Java par certains aspects, mais également du C++ comprenant certaines similitudes avec ces langages comme par exemple la surcharge des opérateurs présents en C++, mais pas en Java.

Il existe également des différences entre le C# et ces deux autres langages. Par exemple, les objets ne peuvent pas être détruits de façon explicite comme en C++. C’est le rôle du ramasse-miette, procédé qui s’occupe de détruire ces instances lors qu’il n’existe plus aucune référence pointant sur ces dernières.



## Scatter View

Le ScatterView est un élément omniprésent de l’environnement de Microsoft Surface. C’est le composant où l’utilisateur pourra ajouter, modifier ou supprimer toute sorte de données, quelle que soit leur nature (texte, image, son, vidéo).

Il autorise également de nombreux contrôles tels que :

* Le redimensionnement pour adapter ce conteneur à l’échelle qui accommoderait l’utilisateur
* Le déplacement et la rotation qui permet à l’utilisateur d’avoir accès à son contenu, peu importe sa position autour du périphérique.

## Entrées

Avec ces éléments de développements s’ajoutent une série d’entrées. Parmi elles, on retrouve :

* Les Blobs ou « taches ». Ce sont des formes elliptiques ayant une certaine aire, assez importante, les différenciant ainsi de ses consœurs. Leur détection se fait par toucher ou par survol sur le périphérique. Une fois reconnue, il lui est attribué un identifiant, une position et une orientation.
* Les Fingers ou doigts sont considérés comme blobs particuliers. Ce qui les différencie de ces dernières sont ses aires plus petites et également sa détection uniquement effectuée au toucher.
* Les tags, enfin sont des éléments imprimés de petites tailles, représentant un motif noir sur une surface blanche. Chacun d’eux représente une valeur entière et n’est détectable que par toucher.

## Outils du SDK

L’input Simulator est un outil fournit par le SDK Surface. Marchant sous le système d’exploitation Seven, cet instrument permet aux développeurs de simuler toutes les interactions mises à disposition par un périphérique surface.

Il reproduit également les entrées définies précédemment, à savoir les fingers, les blobs et les tags. Cet élément est indispensable si l’on ne dispose pas d’une table Surface à disposition ou pour tester rapidement son code. Cependant, il n’est pas infaillible.

C’est pour cela que s’ajoute à lui parmi les outils fournit par le SDK l’Input Visualizer qui complète le précédent. Son rôle est d’afficher les informations associées à chaque entrée. Enfin le Surface Stress quant à lui permet de générer aléatoirement des entrées. Cet instrument fournit alors un test de robustesse pour l’application.

## Eléments du programme

## Nodes

Le Node est l’élément principal de l’application et fait office de branches. D’un design simple, il est muni d’un menu qui guidera l’utilisateur pour la réalisation de ses tâches.

Sa création se fait par simple appui prolongé sur la surface du périphérique. Une fois le contact continu détecté, un cercle de chargement apparait à l’emplacement du doigt. L’animation de chargement indique alors à l’utilisateur la durée minimum d’appui requis pour la création du Node.

L’interruption du contact à la fin de l’animation provoque la création du nœud à la position désignée. Le maintien de l’appui, une fois le délai dépassé entraine la disparition du cercle. Cependant, au relâchement, aucun Node ne sera créé. Cette action est mise en place par prévention pour ainsi empêcher la création de nœuds non souhaités lors d’un contact prolongé involontaire.

Le menu accompagnant ce Node résume en partie les fonctionnalités disponibles mais diffère légèrement selon le type de nœud qui est créé. En effet, il est mis à disposition de l’utilisateur deux genres de nœuds : le NodeText et le NodeImage.

## Fonctionnalités du NodeText

*Add NodeImage* créera un fils du précédent node dont le contenu sera visuel.

*AddTextNode* créera le fils du précédent node dont le contenu sera un texte. Ce fils lui est lié selon la logique du mind mapping, ce qui fait de lui une branche du parent. Le sens de la relation est notifié par le sens de la flèche présente sur le lien : le père pointe le fils.

En revanche, la fonction *Separate from parent* permet de supprimer cette relation père-fils pour en faire deux Nodes libres.

*Remove this* a pour charge de supprimer l’idée et ses ramifications. Une confirmation de l’action est bien sûr adressée à l’utilisateur pour lui signaler l’étendue de son acte.

La fonctionnalité *Color choice* permettra à l’utilisateur de changer l’affichage du Node, notamment sa coloration grâce à la palette de couleurs fournie.

*Edit* est son action principale. Elle permet l’ajout, la modification et la suppression du contenu textuel du Node. À son appel est sollicité le clavier virtuel implémenté accompagné de ses opérations élémentaires telles que la mise en majuscule pour une simple lettre ou pour le texte entier si verrouillage de la touche Caps.

## Fonctionnalités du NodeImage

Les fonctions *Add NodeImage, AddTextNode, Separate from parent* et *Remove this* ont un comportement en tout point identique pour ce Node que pour homologue.

Ce qui diffère dans ce menu est la disparition des fonctionnalités Edit et Color choice étant inutile dans ce cas. À leur place se trouve *Image* choice. Cette tâche autorise l’utilisateur de choisir l’image qui sera contenue dans cet élément. Un déroulement d’image horizontale est mis à disposition pour le choix de l’image. Deux boutons l’accompagnent également, l’un pour la validation de l’image sélectionné et l’autre pour son annulation.

## Relations entre Nodes

La relation entre les nodes peut être assurée lors de sa création comme définie précédemment dans leur menu par la fonction *Add NodeText*. Cependant, si l’utilisateur désire unir deux idées il en a la possibilité grâce à l’appendice au sommet du node. Cet élément répond à une action de drag and drop.

En effet, pour créer cette relation, il suffira d’appuyer sur le composant et de déplacer le doigt tout en maintenant la pression jusqu’au node cible. Une fois à proximité d’un node, un périmètre apparait alors indiquant à l’utilisateur sa zone d’action.

L’opération de lien ne sera réalisée que si le drop se fait dans cette région délimitée. L’action créera alors une relation père-fils, le père étant le node d’où débute le drag et le fils le node concerné par le drop.

## Gestion de fichiers

Pourvoir mettre ainsi le fil de ses pensées sur un support est un avantage. Et pouvoir en garder une trace est un aspect bénéfique. C’est pour cela que l’application est également munie d’un système de sauvegarde pour les utilisateurs désireux de garder une trace de l’exposé de leur réflexion.

Son fonctionnement est simple et est déclenché par l’ajout d’un tag, élément précédemment décri. À la détection de cette entrée est alors mis à la disposition de l’utilisateur un menu de gestion de fichier. Ce dernier comporte les trois tâches standards de conservation ou d’ouverture de fichiers.

## Enregistrement

Les deux fonctions garantissant l’enregistrement son Save As et Save. Lors de la première sauvegarde du fichier, quelle que soit l’option choisit pour effectuer cette opération, il sera demandé à l’utilisateur le nom de sauvegarde. Cette Mind Map répondra dorénavant au nom qui lui été attribué.

Ainsi, si l’action Save est appelée, la sauvegarde sera faite sous le même nom, écrasant alors la sauvegarde précédente. Néanmoins, si c’est la tâche Save As qui est sollicitée, il sera demandé de renommer la Mind Map. Elle sera alors conservée sous cette nouvelle appellation. Sera alors disponible pour l’usager l’ancienne sauvegarde ainsi que la nouvelle. Tous ces fichiers sont conservés dans un dossier prévu par l’application.

## Ouverture

La troisième tâche, Open est celle chargée de l’ouverture des ficher. À son appel, elle ouvre la liste des Mind Map conservé dans le répertoire de sauvegarde de l’application. L’utilisateur peut alors choisir lequel sélectionné et corriger ce choix grâce au bouton de sélection.

## Clavier

Dans ce genre d’application, le clavier représente un élément indispensable et omniprésent. Seulement la Surface 40 ne fournit qu’un unique clavier pour ses opérations d’édition.

Vient alors la notion d’utilisation multiple dans notre application. En effet, le programme doit permettre à un certain nombre d’utilisateurs de pouvoir interagir en même temps pour pouvoir éditer le contenu des nœuds. Or, avec un unique clavier, seul l’un d’entre eux pourra interagir tandis que les autres auraient à attendre leur tour.

C’est pour répondre à cette problématique qu’il a été décidé d’une ré-implémentation d’un clavier. Contrairement à celui fournit par le périphérique, un clavier est affecté à un Node pour permettre une action d’édition simultanée par ses utilisateur sans avoir à se disputer ce périphérique virtuel.

La disposition de ce clavier est semblable à un clavier quelconque. À la droite, l’on retrouve le pavé numérique comprenant ses chiffres et tous les opérateurs numériques les plus courants. Sur la gauche, l’on retrouve les touches de saisie de texte comprenant :

* Les lettres courantes
* Les lettres avec diacritiques les plus courantes (accents)
* La ponctuation
* Les symboles les plus courants (parenthèses, guillemets…)
* Une touche de tabulation
* Une touche de suppression
* Une barre d’espace
* Une touche de verrouillage de majuscules, le changement de couleur de cette touche indique son état
* Une touche Close servant à fermer le clavier
* Une touche Cadenas qui permettra d’immobiliser le clavier ainsi que la node associée pour garantir une édition sans déplacement de l’ensemble dû au touch, et entrainera également l’apparition d’une barre de défilement si le texte est suffisamment long.

# Partie technique.

## Introduction

La création de ce logiciel d’édition de documents collaboratifs a nécessité l’utilisation de quelques outils de programmation tels que XAML ou C. Les connaissances du groupe sur ces outils ont du être approfondies.

Il a également fallu se familiariser avec la table tactile et les outils qui lui sont propres, en particulier les interactions. Nous avons pu utiliser un simulateur, qui nous permettait de faire des tests directement sur nos surfaces de développement. Cependant, nous nous sommes très vite rendus compte que la simulation ne vaut pas la table elle-même.

Pour chaque classe il a fallu créer une partie xaml, qui contenait un aspect plutôt graphique de tailles, dispositions et couleurs, et une partie en C# qui contenait les interactions et liens entre les différentes cases, ainsi que les caractéristiques spécifiques (plus particulièrement pour la différenciation des nodeimage et nodetext).

Le programme a donc été découpé en plusieurs classes, chacune contenant un élément de l’application et définissant ses caractéristiques.

## Classes

### ScatterCustom

La classe ScatterCustom contient la description de base des nodes. Dans cette partie sont décrits l’apparence de tous les types de nodes ainsi que les fonctions qui leur sont associées. Elle hérite de ScatterViewItem, classe du ScatterView qui définit les éléments du programme.

Le XAML contient la taille de la boite node, ses caractéristiques graphiques, et le menu qui lui est associé de base :

* AddNodeText : qui permet de créer un fils de type texte lié au node courant
* AddNodeImage : qui permet de créer un fils de type image lié au node courant
* RemoveThis : qui permet de supprimer le node courant ainsi que ses fils.
* SeparateFromParent : qui permet de séparer le node courant de son parent.

Le code en C# contient la logique des interactions avec les éléments décrits dans le XAML. Il permet de gérer les liens entre les différents nodes, l’affichage du menu, les actions de chaque bouton du menu, les déplacements des nodes et l’héritage.

### NodeText

La classe NodeText ne contient pas de partie XAML. Elle hérite de ScatterCustom et reprend les caractéristiques de base données par le XAML de ScatterViewItem qui contient les éléments communs à tous les nodes. De même, elle reprend les fonctions et le menu de ScatterViewItem en y ajoutant ses propres particularités.

Le NodeText ajoute des boutons au menu du node :

* Color choice : qui permet de définir la couleur du node et de sa police. Cette modification permet de vieux
* Edit : qui fait apparaitre un clavier virtuel et permet d’écrire du texte dans le node courant.

La classe définit également une portion dans laquelle se fait l’écriture de type SurfaceTextBox, une barre de défilement qui n’apparait que lorsque l’on bloque le clavier virtuel et permet d’écrire des textes de plus de six lignes.

### NodeImage

La classe NodeImage ne contient pas de partie XAML. Elle hérite de ScatterCustom et reprend les caractéristiques de base données par le XAML de ScatterViewItem qui contient les éléments communs à tous les nodes. De même, elle reprend les fonctions et le menu de ScatterViewItem en y ajoutant ses propres particularités.

Le NodeImage ajuste le menu de base à ses caractéristiques en ajoute le bouton « Image choice » qui ouvre une fenêtre permettant de choisir l’image à partir d’un dossier d’images interne au programme.

Cette classe redimensionne le node en fonction de la taille de l’image désignée et créé une liste d’images à partir des images contenues dans le dossier. Elle permet de récupérer les informations concernant chaque image pour l’afficher et adapter le node.

### SurfaceWindow1

La classe SurfaceWindow1 hérite de SurfaceWindow et contient toutes les informations sur le fichier courant. C'est-à-dire qu’elle contient le nombre de nodes de chaque type, leur contenu et leurs relations. La classe gère également les autres éléments du programme, comme les cercles de chargement, les menus de sauvegarde et de création de node et les interactions des utilisateurs avec l’interface.

Le XAML contient des informations sur l’image de fond de l’application et le menu de création de node.

Le code en C# contient les fonctions gérant l’apparition et le temps d’affichage du menu de création de nodes, du cercle de chargement de ce même menu et du menu de sauvegarde au niveau du point d’interaction de l’utilisateur avec la table tactile. Il comporte également les fonctions de gestion de la fermeture, de sauvegarde et d’ouverture d’un projet.

Cette classe peut créer une sauvegarde du projet courant et de tous ses éléments ainsi que de leur contenu et leurs liens de parentés. Il permet de conserver le projet tel qu’il a été créé et de pouvoir le rouvrir par la suite. Le choix du nom de la sauvegarde est possible.

SurfaceWindow1 permet à l’utilisateur de créer une node à l’endroit qu’il souhaite d’un simple appui du doigt sur l’écran de la table tactile. Elle dessine les liens entre les nodes symbolisés par des flèches vertes et retient si le fichier a été modifié depuis la dernière sauvegarde.

La classe SurfaceWindow1 gère en général toutes les interactions de l’utilisateur qui concernent directement l’interface de base. C’est une gestion des actions sur le document dans sa globalité et non spécifiquement pour un élément du fichier.

### PaletteCouleurs

La classe PaletteCouleurs hérite de ScatterViewItem et est utilisée dans le NodeText. Cette classe permet l’affichage de la palette de couleurs qui modifie la couleur du NodeText ainsi que de sa police.

Le XAML contient la taille de la palette et une grille de cases de couleurs chacune associée à un nom, ainsi que deux boutons : l’un pour valider la modification de couleur et l’autre pour annuler le changement.

Le code en C# décrit les fonctions s’appliquant sur cette palette de couleurs, c’est-à-dire l’action qui se produit lors de l’appui sur chaque case de couleur ou sur les autres boutons de la palette ainsi que la définition des couleurs de base ou courante (modifiée) du node concerné.

### LoadCircle

La classe LoadCircle concerne le cercle de chargement qui apparait lors de l’appui du doigt d’un utilisateur sur un endroit vide du document. Ce cercle symbolise le temps durant le doigt doit rester enfoncé pour permettre l’apparition du menu de création des nodes. Cette classe hérite aussi de ScatterViewItem.

Le XAML décrit un cercle bleu d’opacité qui augmente avec le temps et apparait complètement au bout d’une seconde et demie. Il définit également un fragment de cercle qui fait un tour du cercle bleu dans ce même laps de temps. Ces deux actions sont réglées ici pour débuter une demi-seconde après le début de l’appui. Ainsi, le temps de chargement du menu est de deux secondes, ce qui évite qu’un node n’apparaisse à chaque contact même non intensionnel avec la table.

Le code en C# créé seulement un constructeur qui permettra l’utilisation du LoadCircle dans la classe SurfaceWindow1 et une fonction qui détecte le chargement.

### MenuCreation

La classe MenuCreation hérite de ScatterViewItem. Cette classe sert à l’affichage et à l’utilisation du menu de création des nodes. Cet élément apparait lors d’un appui sur un espace vide de l’écran après la fin du chargement indiqué par le cercle de chargement. Le menu principal reste alors affiché durant un temps défini dans la classe SurfaceWindow1.

Le fichier XAML contient les caractéristiques graphiques du menu ainsi que les trois boutons :

* Text : permet la création d’un node de type TEXT
* Image : permet la création d’un node de type IMAGE
* X : ferme le menu de création et annule l’apparition d’un nouveau node

Le fichier C# contient les fonctions liées à chacun des boutons créés dans le XAML qui permettront l’utilisation du menu de création gérée par la classe SurfaceWindow1.

### MenuPrincipal

La classe MenuPrincipal hérite de ScatterViewItem. Elle gère la création et les caractéristiques du menu principal ou menu de sauvegarde. Ce menu apparait à l’appui d’un tag sur l’écran et permet alors la création d’une nouvelle sauvegarde, la sauvegarde ou l’ouverture d’un fichier existant. Les sauvegardes conservent les nodes, leur contenu et les héritages. Cette classe est utilisée par la classe SurfaceWindow1 grâce à son constructeur définit dans le fichier C#.

Le fichier XAML gère l’apparence graphique et créé les trois boutons du menu de sauvegarde :

* Save as : permet de créer une nouvelle sauvegarde du document courant dont on peut spécifier le nom à l’aide d’un clavier virtuel qui apparait.
* Save : permet d’enregistrer le fichier courant sans forcément créer une nouvelle sauvegarde su il existe déjà une sauvegarde de ce document.
* Open : permet d’ouvrir un fichier déjà existant pour le consulter ou le modifier.

### ListeSauvegarde

La classe ListeSauvegarde gère la liste de documents déjà enregistrés que l’on peut charger depuis le programme. Elle hérite de ScatterViewItem et apparaît lorsque l’utilisateur appuie sur le bouton « Open » du menu principal.

Le fichier XAML contient l’affichage de la liste de documents enregistrés présents dans le dossier du projet. L’affichage est divisé en deux parties. En haut, il y a une partie qui permettra l’affichage des noms des fichiers enregistrés qui seront chargés en cas d’appui d’un utilisateur sur leur nom. En dessous, il y a un bouton « annuler » qui permet de ne pas charger de document existant et de rester sur le document courant.

Le code en C# contient les fonctions permettant l’acquisition des noms des fichiers sauvegardés existant dans le dossier, la création d’une liste et son affichage dans la partie destinée à cet effet. Il gère aussi les actions du programme en cas d’appui sur le bouton annuler et reçoit le nom du fichier que l’utilisateur a sélectionné et qui sera chargé dans la classe SurfaceWindow1.

### ListeImage

La classe ListeImage hérite de ScatterViewItem. Elle gère la liste des images accessibles depuis le programme et contenues dans un dossier spécifique. Cet élément apparait lorsque l’on appui sur « Image Choice » dans le menu d’un node de type « Image ». Cette classe permet à l’utilisateur de sélectionner une image qui sera affichée dans le NodeImage.

Le code XAML contient la taille et les autres aspects graphiques du ListeImage. Il définit un espace où sera affichée une liste verticale des images qui pourront être chargées depuis le document courant.

Le fichier C # gère les fonctions d’initialisation de la liste d’images qui créé une liste à partir des images disponibles dans un dossier donné. Les fonctions vérifient si le fichier ajouté à la liste est bien une image et gèrent l’appui sur la touche d’annulation de l’élément ListeImage.

### ClavierVirtuel

La classe ClavierVirtuel hérite de ScatterViewItem puisqu’il s’agit d’un élément du document. Le clavier virtuel créé dans cette classe est utilisé aussi bien pour l’édition de texte que pour l’écriture du nom du fichier de sauvegarde du document collaboratif réalisé sur la table tactile. Ce clavier a été créé pour contenir les touches principales le plus couramment utilisées dont les lettres de l’alphabet latin, les chiffres et quelques caractères spéciaux dont les accents, parenthèses,…

Le fichier XAML définit l’aspect graphique du clavier virtuel : la position des touches et leur taille pour obtenir un clavier esthétique. Il contient également les noms et caractères associés à chaque touche.

Le code en C# créé les fonctions qui gèrent l’appui sur chaque touche du clavier :

* L’appui sur les touches normales entraine l’écriture du caractère qui leur est associé.
* L’appui sur la touche « close » ferme le clavier et réactive les mouvements du node si ceux-ci avaient été bloqués.
* L’appui sur la touche « BackSpace » supprime l’élément précédent. L’appui long sur cette touche entraine la suppression successive des éléments précédents jusqu’à l’arrêt de l’appui.
* L’appui sur la touche « Caps » entraine que l’écriture des caractères suivants se fera en majuscule jusqu’à un nouvel appui sur cette touche.
* L’appui sur « Cadenas » bloque les mouvements du node et permet l’apparition d’une barre de défilement si le texte est plus long que la taille du node.

# Conclusion

Ce projet nous a permis de travailler sur un nouvel outil, la table tactile, et de programmer pour une application sur une interface différente de celle de l’ordinateur que l’on a eu l’habitude d’utiliser durant notre formation à l’IUT.

Nous avons du faire face de ce fait à différents problèmes comme les nombreuses utilisations simultanées du programme ou le passage de la machine virtuelle à l’interface réelle, qui ont impliqués notamment de ne pas capter des clics de souris ou des appuis clavier, mais des interactions manuelles. Et les différences de réactions entre les deux plateformes. Il nous a également fallu prendre en compte les différents types d’actions possibles avec la table tactile comme l’appui d’un seul doigt ou bien d’une paume de main.

Ce projet va permettre à différents membres d’une équipe d’organiser leurs idées en les présentant sous forme de textes, images, vidéos ou sons et de les lier les unes aux autres pour pouvoir créer des solutions pédagogiques dans le cadre de leurs activités. Tout ceci restant dans le cadre de la société Maskott puisque ce programme fonctionne sur la table tactile.

Pour poursuivre ce projet, on pourrait pouvoir transférer le résultat de notre travail sur une autre interface comme un tableau interactif ou une tablette pour avoir une vision générale de l’arbre d’informations ou que chacun puisse en avoir une version. Ceci pourrait aussi permettre de transférer une idée à partir d’un périphérique extérieur vers l’arbre d’idées réalisé par le groupe.