14/04/2014

Eye tracker pour visualiser des images

Projet d’approfondissement

Master of Science HES-SO in Engineering

Orientation : Technologies de l’information et de la communication (TIC)

Sous la direction de

Prof. Olivier, Hüsser

Haute école ARC

William Droz

Accepté par la HES‑SO//Master (Suisse, Lausanne) sur proposition de

Prof. xxx, conseiller du projet d’approfondissement

Lausanne, le … 20 ??

|  |  |
| --- | --- |
| Prof. xxx  Conseiller | Prof.  Responsable de la filière …. |

Table des matières

[1 Introduction 5](#_Toc387676702)

[1.1 Définition de l’oculométrie 5](#_Toc387676703)

[1.2 Cahier des charges 5](#_Toc387676704)

[2 État de l’art 6](#_Toc387676705)

[2.1 Les bonnes pratiques 6](#_Toc387676706)

[2.1.1 Ce qu’il ne faut pas faire 6](#_Toc387676707)

[2.2 Les différentes technologies 7](#_Toc387676708)

[2.2.1 Les lunettes 7](#_Toc387676709)

[2.2.2 Les caméras - vidéo-oculographie 7](#_Toc387676710)

[2.2.3 électro-oculographique 7](#_Toc387676711)

[2.2.4 Galvanométrique 8](#_Toc387676712)

[2.2.5 Reflet cornéen 8](#_Toc387676713)

[2.3 Domaines d’applications 8](#_Toc387676714)

[2.3.1 Médical 8](#_Toc387676715)

[2.3.2 Conception d’interfaces graphiques 8](#_Toc387676716)

[2.3.3 Les jeux 8](#_Toc387676717)

[2.3.4 Militaire 8](#_Toc387676718)

[2.4 Les nouveautés à attendre 9](#_Toc387676719)

[2.4.1 Tobii pour les jeux 9](#_Toc387676720)

[3 Planification 9](#_Toc387676721)

[3.1 Méthode de développement 9](#_Toc387676722)

[3.2 Planification initiale 9](#_Toc387676723)

[3.3 Planification après la première itération 9](#_Toc387676724)

[3.4 Les déviations 9](#_Toc387676725)

[4 Analyse 10](#_Toc387676726)

[4.1 Fonctionnement du EyeX SDK 10](#_Toc387676727)

[4.1.1 Interactor 11](#_Toc387676728)

[4.1.2 Query 11](#_Toc387676729)

[4.1.3 Events 11](#_Toc387676730)

[4.1.4 Déroulement d’un scénario simple 12](#_Toc387676731)

[4.1.5 Limitations 13](#_Toc387676732)

[4.2 Choix des technologies 14](#_Toc387676733)

[4.2.1 C++ avec Qt 14](#_Toc387676734)

[4.2.2 C# avec WPF 15](#_Toc387676735)

[4.2.3 Conclusion 15](#_Toc387676736)

[4.3 Diagrammes des cas d’utilisations 16](#_Toc387676737)

[5 Conception 17](#_Toc387676738)

[5.1 Diagrammes de classes 17](#_Toc387676739)

[5.1.1 Première partie 17](#_Toc387676740)

[5.1.2 Seconde partie 18](#_Toc387676741)

[5.2 Interfaces graphiques 19](#_Toc387676742)

[5.2.1 Menu principal 19](#_Toc387676743)

[5.2.2 Menu navigation 20](#_Toc387676744)

[5.2.3 Pièges à éviter 20](#_Toc387676745)

[6 Implémentation 21](#_Toc387676746)

[6.1 Trouver quel control est « regardé » 21](#_Toc387676747)

[6.1.1 Surface absolue (technique naïve) 22](#_Toc387676748)

[6.1.2 Surface relative 22](#_Toc387676749)

[6.2 Partie visualisation des images 23](#_Toc387676750)

[6.2.1 Zoom 23](#_Toc387676751)

[6.2.2 Scroll 24](#_Toc387676752)

[6.3 Partie navigation 24](#_Toc387676753)

[7 Evolutions 25](#_Toc387676754)

[7.1 Utiliser une autre technologie que Tobii Rex 25](#_Toc387676755)

[8 Tests et vérifications 25](#_Toc387676756)

[9 Problèmes rencontrés 25](#_Toc387676757)

[9.1 Responsabilité des objets 25](#_Toc387676758)

[9.2 Lacunes C# et WPF 25](#_Toc387676759)

[10 Conclusion 26](#_Toc387676760)

[11 Références 26](#_Toc387676761)

[12 Annexes 27](#_Toc387676762)

[12.1 Code C++ pour Tobii Rex 27](#_Toc387676763)

# Introduction

L’oculométrie[[1]](#footnote-1) a un grand potentiel que de plus en plus d’acteurs tentent t’exploiter. Ce projet a pour but de démontrer ce qu’il est possible de faire avec une technologie accessible (Tobii Rex[[2]](#footnote-2)) sans pour autant en exclure d’autres.

Il a été décidé de faire une application qui permet de naviguer dans des dossiers d’images.

## Définition de l’oculométrie

*« L’oculométrie (en anglais Eye-tracking) regroupe un ensemble de techniques permettant d'enregistrer les mouvements oculaires. Les oculomètres les plus courants analysent des images de l'œil humain capturées par une caméra, souvent en lumière infrarouge, pour calculer la direction du regard du sujet. En fonction de la précision souhaitée, différentes caractéristiques de l'œil sont analysées. D'autres techniques sont basées sur les variations de potentiels électriques à la surface de la peau du visage ou encore sur les perturbations induites par une lentille spéciale sur un champ magnétique »*

Source : wikipédia révision du 10 juillet 2013

## Cahier des charges

TODO mettre le cahier des charges

# État de l’art

Voici un état des lieux de ce qu’il se fait actuellement en matière d’oculométrie. Ce chapitre abordera les bonnes pratiques à prendre en compte, une brève introduction aux techniques de captures « du regard » et terminera avec une sélection de domaines d’applications.

## Les bonnes pratiques

En matière d’oculométrie il est nécessaire de faire attention à plusieurs points. En effet, il pourrait être tentant de dire « Quand l’utilisateur cligne des yeux, alors fait l’action X ». Alors que justement, ceci est une erreur.

Chez Tobii par exemple, ils utilisent une touche d’activation (sur le clavier). Ainsi il suffit de regarder la zone que l’on désire activer, puis enfin appuyer sur cette touche.

### Ce qu’il ne faut pas faire

Voici une liste non-exhaustive des choses à éviter

|  |  |
| --- | --- |
| Piège | Raison |
| Utiliser le clignement de l’œil pour interagir | L’œil n’est pas fait pour faire des actions |
| Utiliser une gestuellement des yeux pour interagir | L’œil n’est pas fait pour faire des actions |
| Utiliser des zones d’interactions de surface faible | Les outils ne sont pas assez précis |
| Demander l’attention de l’utilisateur trop longtemps | L’utilisateur peut être distrait et/ou se fatiguer. |

## Les différentes technologies

Au fil des années, plusieurs technologies ont vu le jour avec plus ou moins de succès. Nous avons essayé de les catégoriser.

### Les lunettes

Les lunettes ne sont pas à proprement parler une technologie d’enregistrement oculaire, mais plutôt un moyen d’exploiter des techniques.

### Les caméras - vidéo-oculographie

Utiliser des caméras semblent une bonne idée. Malheureusement cela ne donne pas de bons résultats. C’est pour ça qu’on combine les caméras avec des lumières infrarouges pour amplifier la brillance de la pupille.

Les premières versions ne sont pas très sexy



Figure 1 vidéo-oculographie

Cela nous donne un aperçu des progrès réalisés avec les Tobii Rex.

### électro-oculographique

Technique qui mesure les différences de potentiels électriques provoqués par la rotation des yeux. Cette technique est très peu utilisé car ce n’est pas précis pour reconnaître où l’utilisateur regarde.

### Galvanométrique

Mesure des signaux électriques à l’aide d’un champ magnétique. Cette technique est très précise mais a le désavantage de nécessiter une lentille.

### Reflet cornéen

Cette technique ressemble un peu à celle de la vidéo-oculographie. On utilise plusieurs éclairages infrarouges afin de pouvoir détecter des variations sur le reflet de la cornée.

Les Tobii Rex utilisent cette technique

## Domaines d’applications

Voici les domaines principaux où est utilisées l’oculométrie.

### Médical

Il y a plusieurs sous-domaines, voici quelques exemples :

* Détecter des dyslexies chez un patient[[3]](#footnote-3)
* Etudier la compétence d’un chirurgien[[4]](#footnote-4)
* Evaluer des étudiants en médecine[[5]](#footnote-5)

### Conception d’interfaces graphiques[[6]](#footnote-6)

Il n’est pas toujours évident de bien concevoir une interface graphique. En posant des eyes tracker sur un certain nombre de volontaire, nous pouvons recueillir des informations précieuses qui peuvent nous permettre de modifier notre interface.

Voici quelques informations qui peuvent nous intéresser :

* Qu’est-ce que l’utilisateur regarde en premier ?
* Quel chemin parcourt ces yeux ?
* Est-ce qu’il fait des allez-retours ?
* Est-ce qu’il doit regarder longtemps le menu pour comprendre comment ça marche ?

### Les jeux

Les jeux-vidés essayent sans arrêt d’innover et de proposer des nouvelles façons de jouer.

### Militaire

Les pilotes utilisent des casques[[7]](#footnote-7) qui leurs permettent d’être plus efficaces.

## Les nouveautés à attendre

Toutes les entreprises ne sont pas bavardes sur leurs recherches et développements avec l’oculométrie. Tobii nous ont quand même renseigné sur le sujet.

### Tobii pour les jeux[[8]](#footnote-8)

Tobii veulent tenter de fournir du matériels suffisamment réactifs pour les joueurs. Car les Tobii Rex ne sont pas adaptées aux joueurs. Si ceux-ci font des gestes brusques, les yeux ne seront plus détectés pendant quelques secondes.

# Planification

Ce chapitre arborera tout ce qui touche à la planification, tel que la méthode de développement, la planification initial et après la première itération. En fin de chapitre, nous avons une analyse des déviations.

## Méthode de développement

Nous avons choisi une méthode semi-itérative. À Chaque nouvelle itération, il y a un incrément logiciel (dans le sens ajout de features) ainsi qu’une amélioration des features précédentes.

Cette technique va permettre d’avoir une application utilisable dès la première itération.

Avec plus de temps, la méthode RAD avec prototype jetable aurait été meilleure, car elle permet de se baser sur l’expérience acquis sur le prototype précédant pour améliorer le prochain. Ceci est très utile quand on ne maîtrise pas ou peu la technologie dans laquelle on développe.

## Planification initiale

## Planification après la première itération

## Les déviations

# Analyse

Ce chapitre concerne l’analyse et les réflexions effectuées afin de répondre aux exigences du projet. Nous avons analysé le fonctionnement du SDK fournit par Tobii, les choix des technologies et des diagrammes des cas d’utilisations.

## Fonctionnement du EyeX SDK

Le SDK fournit par Tobii permet de de simplifier considérablement le développement d’une application « standard ». Ce chapitre va synthétiser son fonctionnement, son but n’est toutefois par de remplacer la lecture du « developer’s Guide » fournit par Tobii.

Il y a trois composants qui interagissent entre eux :

* **EyeX Engine**
  + Sélectionne le control[[9]](#footnote-9) que l’utilisateur regarde et/ou actionne.
  + Informe l’application via un « Event » des intentions de l’utilisateur.
* **EyeX Controller**
  + Fournit au eyeX Engine les coordonnés du regard de l’utilisateur.
* **Application**
  + Indique à l’EyeX Engine les coordonnés de ces contrôles.
  + Réagit aux « Events » envoyé par l’EyeX Engine.

### Interactor

Les interactors définissent les zones à l’écran avec lesquelles l’utilisateur peut interagir. Voici un exemple avec l’interface de Google avec trois Interactors :



Figure 2 Interface de google.ch

### Query

Quand l’utilisateur regarde l’écran, l’EyeX Engine demande s’il y a des interactors dans la zone du regarde de l’utilisateur. Cette demande est adressée à l’application plusieurs fois par secondes.

L’application est sensé lui répondre une liste d’interactors.

### Events

Lors que l’utilisateur désire activer un control, il va appuyer sur la touche d’activation. Celle-ci est interceptée par l’EyeX Engine puis celui-ci détermine quel est l’interactor que l’utilisateur regarde (grâce aux query vu avant).

L’EyeX Engine va ensuite envoyé un Event à l’application l’informant quel interactor a été activé.

### Déroulement d’un scénario simple

Nous allons montrer pas à pas le déroulement sur l’interface Google vu précédemment.



Figure 3 Interface google.ch avec Interactors et le regard de l'utilisateur

Comme dirait le visiteur du futur[[10]](#footnote-10), voilà ce qui va se passer :



Figure 4 Le visiteur du futur qui dit "voilà ce qui va se passer"

1. L’EyeX Controller va informer l’EyeX Engine que où regarde l’utilisateur
2. L’EyeX Engine va faire une Query à l’application pour lui demander s’il y a des interactors dans la zone que regarde l’utilisateur (en rouge sur la Figure 3).
3. L’application va répondre avec une liste contenant les interactors 1 et 2.
4. L’utilisateur appui sur la touche d’activation
5. L’EyeX Engine détermine quel est l’interactor le plus probable entre le 1 et le 2. Il choisit le 2 car l’intersection est plus importante.
6. L’EyeX Engine envoi un Event à l’application lui indiquant que l’interactor 2 vient d’être activé.
7. L’application fait le code qui le développeur a déterminé.

### Limitations

Ce SDK simplifie le travail du développeur, sauf si celui-ci veut sortir des sentiers battus (ce qui est notre cas).

En effet, si on désire définir plusieurs comportements différents en fonctions de plusieurs touches différentes… Cela n’est pas pris en charge. Il n’est pas possible d’avoir des events du genre « Interactor X activé avec la touche A » et « Interactor Y activé avec la touche B ».

Pour notre application, nous avons donc choisit de nous en tenir aux query comme interaction avec l’EyeX Engine. Ceux-ci nous informent de la zone regardée de l’utilisateur.

## Choix des technologies

Il y a deux choix de SDK pour les Tobii Rex. L’un est en C++, l’autre en C#. Pour chaque langage, des exemples sont fourni.

Pour le C#, les exemples sont en Utilisant des « WinForms [[11]](#footnote-11)». Pour le C++ ils sont en GDI[[12]](#footnote-12).

Dans ce chapitre, nous allons tenter de comparer les deux voies possibles.

### C++ avec Qt[[13]](#footnote-13)

La première solution consiste à utiliser C++ avec le Framework Qt pour l’interface graphique. Nous n’évoquons par GDI ici, car nous avons moins de 30 ans.

Qt est la seule manière moderne de créer des interfaces graphiques en C++ que nous maîtrisons.

Voici les avantages et les inconvénients pour cette méthode :

### C# avec WPF[[14]](#footnote-14)

Les WinForms sont de moins en moins utilisés au profit de WPF, car ceux-ci sont plus malléables en séparant bien le code et l’interface graphique. Nous avons donc opté pour WPF comme technologie qui va avec C#.

Voici les avantages et les inconvénients pour cette méthode :

### Conclusion

Le projet étant dans un cadre des études, nous choisissons « **C# avec WPF** » afin de pouvoir nous perfectionner avec cette technologie qui est assez présente dans les entreprises.

Malheureusement, il y a un « coût » avec cette méthode : En effet, vu que nous ne maitrisons pas très bien la technologie, le risque de faire un logiciel peu maintenable et peu optimisé est grand.

## Diagrammes des cas d’utilisations

Voici le diagramme « UseCase » sur lequel nous allons nous baser pour la conception :

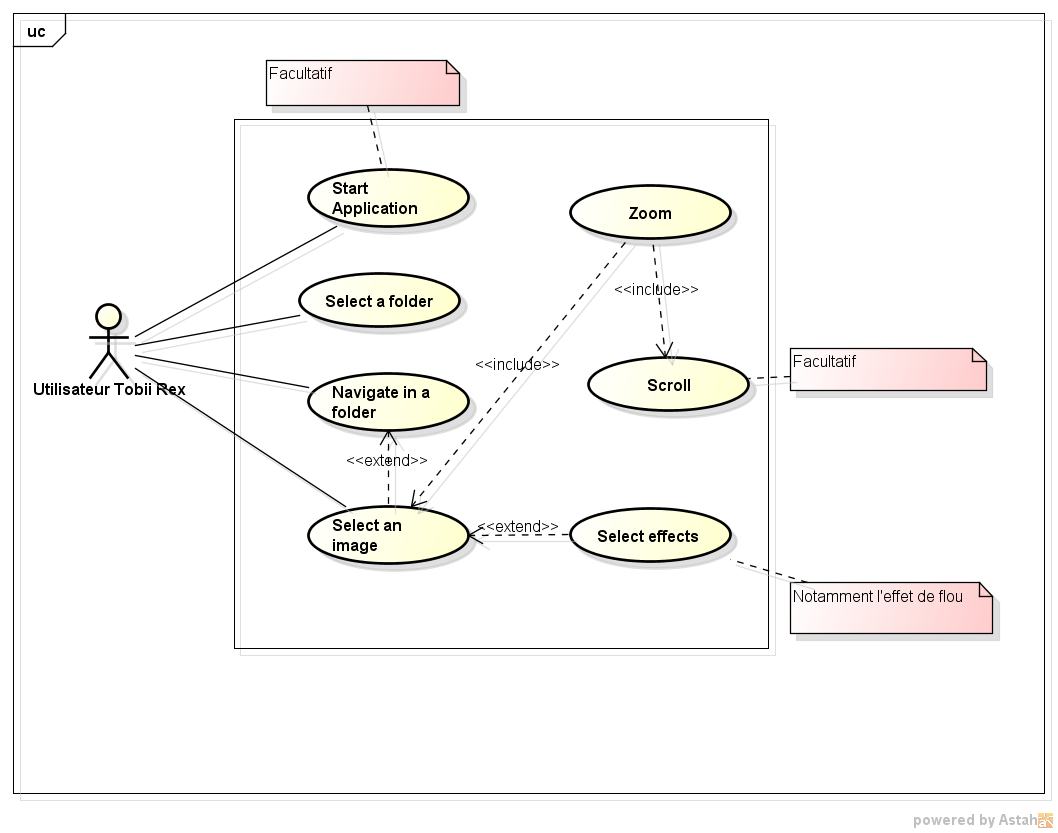


Figure 5 UseCase de l'analyse

# Conception

Ce chapitre contient les éléments qui ont guidé l’implémentation. Nous avons donc les diagrammes de classes et les interfaces graphiques

## Diagrammes de classes

Pour plus de lisibilité, nous avons séparé le diagramme en deux parties.

### Première partie

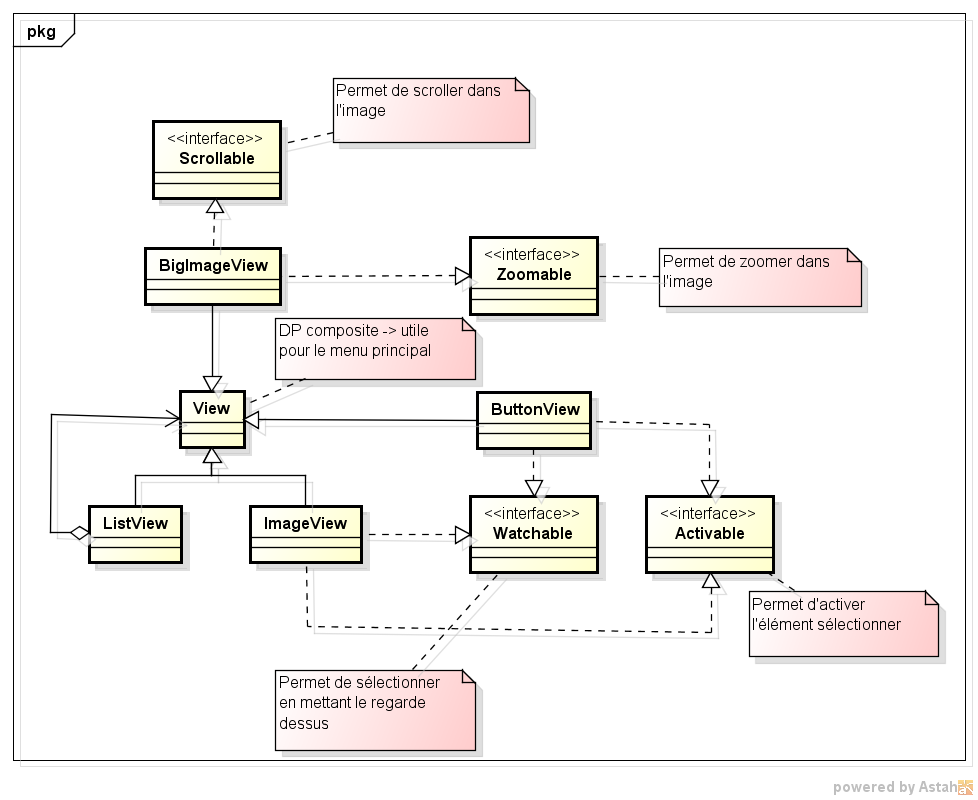


Figure 6 Diagramme de classe partie I

### Seconde partie

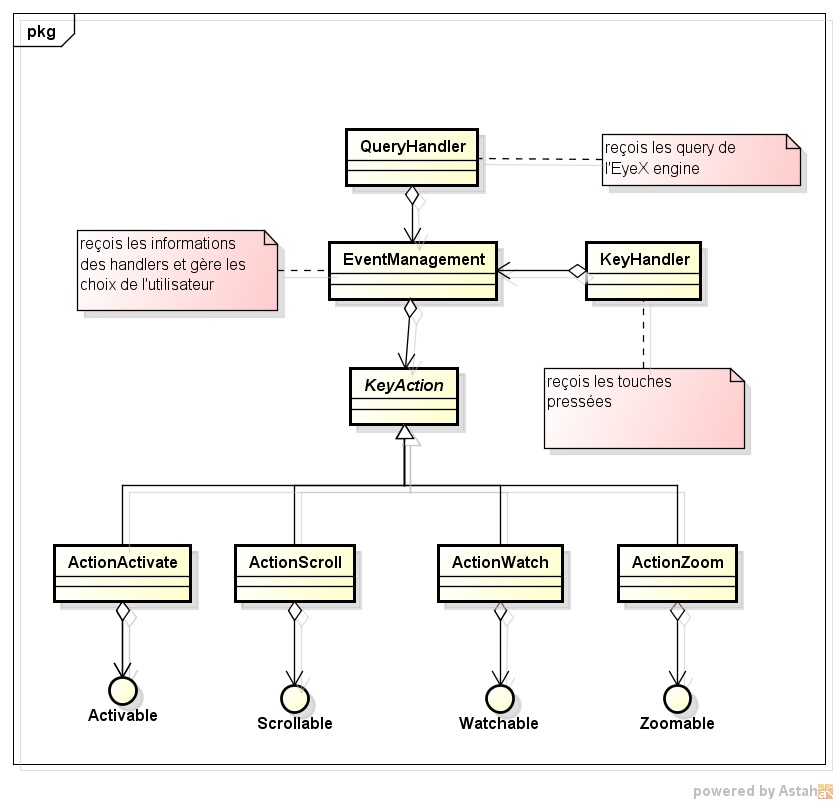


Figure 7 Diagramme de classe partie II

## Interfaces graphiques

Voici les maquettes des interfaces graphiques pour le menu principal ainsi que pour la partie de navigation.

### Menu principal

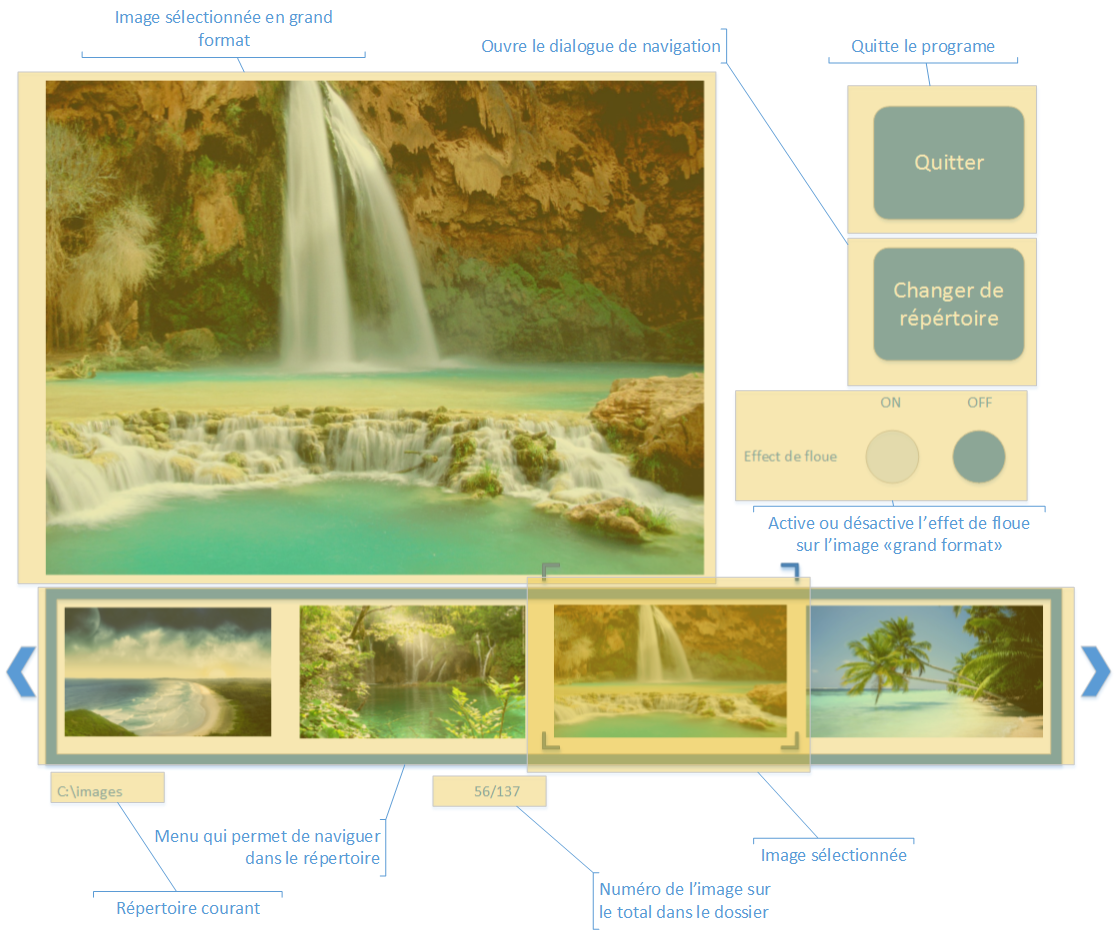


Figure 8 Maquette du menu principal

### Menu navigation

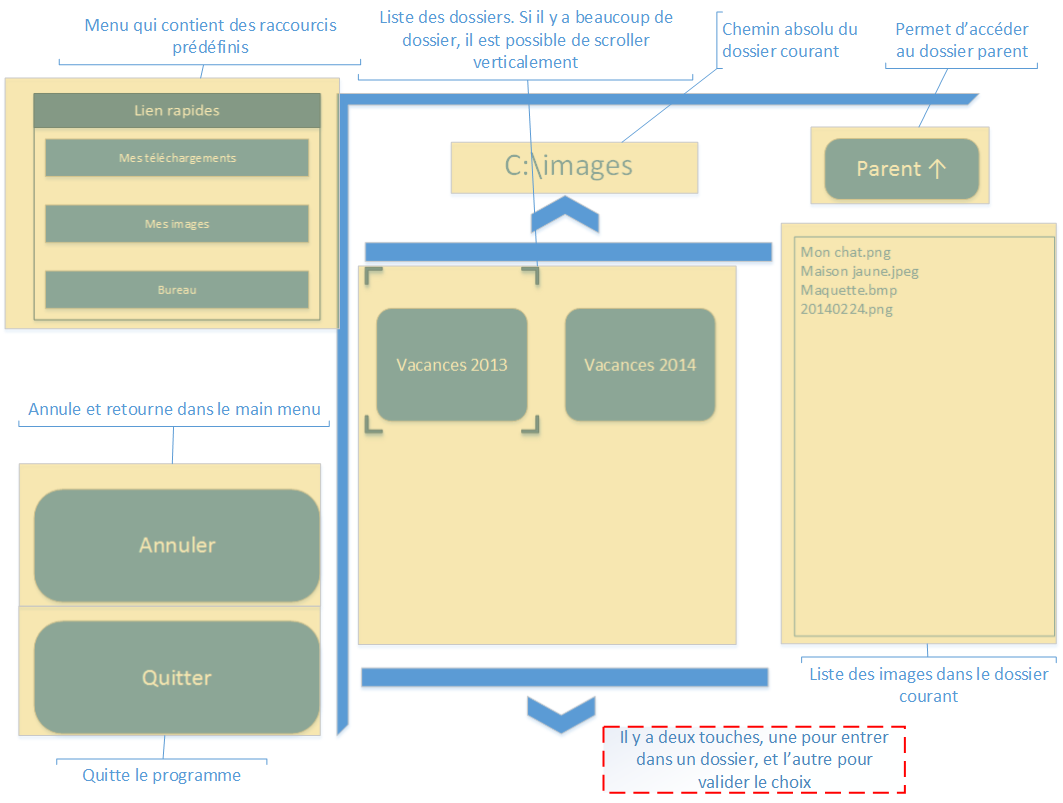


Figure 9 Menu de navigation

### Pièges à éviter

Comme on peut s’en douter, utiliser des boutons de petite taille n’est pas une bonne idée. Tobii n’a pas de taille recommandée[[15]](#footnote-15), ils indiquent juste d’éviter d’utiliser des contrôles trop petit car il y a du bruit (dans le sens imprécision).

Là où ça devient moins évident, c’est de penser à minimiser l’angle entre le Tobii Rex et le control. En effet, si nous mettons des éléments en haut de l’écran, il y aura plus d’erreur qu’en bas de l’écran.

Il y a une petite astuce qu’on peut utiliser dans certain cas pour augmenter l’efficacité de l’eyes tracking. Celle-ci consiste à utiliser une touche, par exemple ‘Q’. Quand celle-ci est appuyée, un nouveau menu apparait. Il est possible d’utiliser plusieurs touches pour plusieurs menus. Cela permet de minimiser le nombre de contrôles affichés à l’écran en même temps.

# Implémentation

Lors de l’implémentation, certaines parties méritent de figurer dans le rapport. Notamment les parties qui sont difficile à comprendre d’un simple coup d’œil, ou encore celles qui permettent des features importantes.

## Trouver quel control est « regardé »

Il n’est pas toujours facile de déterminer quel control l’utilisateur regarde. Reprenons pour exemple l’image de Google.

Figure 10 Quel est l'interactor choisit ?

Le regard est posé entre deux interactors (sur la barre de recherche et sur le bouton « Recherche Google »). Mais lequel l’utilisateur veut réellement voir ?

Nous allons parler de deux techniques possibles.

### Surface absolue (technique naïve)

Cette première technique est présentée comme naïve pour des raisons qui seront bientôt évidentes.

Pour connaître l’interactor regardé, il suffit de comparer l’intersection et choisir celui dont la surface d’intersection est la plus grande.

Exemple :   
Z = surface du regard de l’utilisateur  
a et b sont deux interactors.

Si A > B, alors l’interactor a est choisi, sinon c’est le b.

Cette technique est très simple et permet d’avoir de bons résultats. Néanmoins, il y a un inconvénient majeur. Prenons le cas suivant :



Figure 11 Bouton 1 ou Bouton 2 ?

Avec cette technique, il est très difficile d’atteindre le Bouton 2, car ça surface personnelle est beaucoup plus petite que celle du bouton 1. L’intersection sera plus grande chez Bouton 1 que chez Bouton 2.

Nous allons donc essayé de pallier ce problème avec la solution suivante.

### Surface relative

L’astuce dans cette technique est de comparer « à quel pourcentage a ou b est intersecté ».

Sur la figure 11, le bouton 2 est intercepté à ~70%, tandis que le bouton 1 est intercepté à ~20%. Dans cette technique, c’est bouton 2 qui est donc choisi-

C’est donc cette technique qui nous parait plus juste que nous avons décidé d’implémenter.

Nous avons remarqué que cette technique ressemble beaucoup à l'indice de Jaccard[[16]](#footnote-16), sauf qu’on prend en compte uniquement la surface de l’interactor au dénominateur.

## Partie visualisation des images

Le dialogue qui permet de visualiser les images d’un dossier est la partie principale du projet. Nous aborderons deux principes clés de notre implémentation : Le zoom et le scroll.

### Zoom

Le zoom est l’action qui consiste se focaliser sur un point précis de l’image et d’en augmenter le niveau de détail en l’agrandissant.

Nous avons essayé plusieurs techniques, certaines ont eu plus de succès que d’autres. Nous allons présenter deux d’entre-elles. Celle qui sera présenté en premier est moins performante que la suivante, néanmoins elle mérite d’être présentée.

Pour expliquer avec le plus de clarté, nous allons présenter un scénario simple.



Figure 12 Scénario d'un zoom

La zone ‘A’ est la surface du regard de l’utilisateur. Celui-ci veut donc zoomer dans l’image, voici comment nous allons tenter d’effectuer cette action.

Nous créons un nouveau point au centre de l’image. Ensuite nous allons déduire le vecteur qui se trouve entre le centre de l’image et le centre de la zone ‘A’, nommons celui-ci « VZOOM ».

C’est ici que va commencer à diverger les deux techniques. Pour la première le vecteur « VZOOM » va être normalisé. Ensuite nous allons faire une translation depuis le centre de l’image dans le sens du « VZOOM » normalisé en le multipliant par un scalaire qui permet de choisir la vitesse de déplacement.

Ensuite nous effectuons un changement d’échelle afin d’agrandir l’image.

Cette technique permet d’avoir un déplacement fluide est assez agréable. Toutefois celle-ci a un inconvénient majeur : la vitesse de déplacement ne prend pas en compte si l’utilisateur regarde loin ou prêt du centre de l’image. Cela rend difficile d’accéder rapidement aux endroits éloigné de centre de l’image.

Pour pallier cet inconvénient, nous avons mise au point une seconde technique. Dans celle-ci, nous de normalisons plus le vecteur « VZOOM » et le scalaire à un espace plus petit que ‘1’ afin de limite la vitesse (cela permet d’éviter des changements brusques).

Cette seconde technique permet de zoomer beaucoup plus rapidement que la première et est plus agréable à utiliser.

### Scroll

Le scroll est organisé comme le zoom, mais sans faire de changement d’échelle.

## Partie navigation

Faire ou ne pas faire…

# Evolutions

Lors du développement dans une technologie que l’on ne maitrise pas, il est difficile de faire du code « maintenable » et « évolutif ». Néanmoins, nous avons réussi à créer un code qui permet facilement de faire évoluer l’application de manière significative, sans pour autant devoir tout réécrire.

L’aspect évolutif le plus intéressant, c’est la possibilité d’utiliser une autre technologie que les Tobii Rex afin de capturer les mouvements des yeux.

## Utiliser une autre technologie que Tobii Rex

# Tests et vérifications

Étant donné que l’application à rendre est un « démonstrateur », les tests ne sont pas centraux. Toutefois, nous avons créé un protocole de test afin d’avoir une vision rapide de ce qui fonctionne.

# Problèmes rencontrés

Ce chapitre énumère les difficultés majeures rencontrées durant le projet

## Responsabilité des objets

Lors de la conception, nous voulions que chaque objet qui s’affiche à l’écran ait la responsabilité de s’afficher.

Le problème, c’est qu’en WPF c’est aux objets parents (dans le sens de la composition) qui incombent cette responsabilité.

Nous avons pu palier à ce problème en modifiant la conception de sorte que les objets retourne un « FrameworkElement » et ainsi les parents peuvent les afficher facilement.

## Lacunes C# et WPF

N’étant pas très familier avec la technologie C#/WPF, à plusieurs reprises nous nous sommes dits : « j’aurais dû faire comme ceci ou comme cela » après l’implémentation effective.

De toute évidence, nous aurions une implémentation sensiblement différente si nous devions recommencer.

# Conclusion

# Références

# Annexes

## Code C++ pour Tobii Rex

|  |
| --- |
| EyeXHost**::**EyeXHost**()**  **:** \_state**(**Initializing**),**  \_hWnd**(**nullptr**),**  \_statusChangedMessage**(**0**),** \_focusedRegionChangedMessage**(**0**),** \_regionActivatedMessage**(**0**),**  \_focusedRegionId**(-**1**),**  \_context**(**TX\_EMPTY\_HANDLE**),**  \_connectionStateChangedTicket**(**0**),** \_queryHandlerTicket**(**0**),** \_eventHandlerTicket**(**0**)**  **{**  // initialize the EyeX Engine client library.  txInitializeSystem**(**TX\_SYSTEMCOMPONENTOVERRIDEFLAG\_NONE**,** nullptr**,** nullptr**);**  // create a context and register event handlers, but don't enable the connection to the engine just yet.  // we'll enable the connection in the Init method, when we're ready to handle the  // connection-status-changed notifications.  bool success **=** txCreateContext**(&**\_context**,** TX\_FALSE**)** **==** TX\_RESULT\_OK**;**  success **&=** RegisterConnectionStateChangedHandler**();**  success **&=** RegisterQueryHandler**();**  success **&=** RegisterEventHandler**();**  **if** **(!**success**)**  **{**  SetState**(**Failed**);**  **}**  **}** |

1. http://fr.wikipedia.org/wiki/Oculom%C3%A9trie [↑](#footnote-ref-1)
2. http://www.tobii.com/en/eye-experience/buy/rex/ [↑](#footnote-ref-2)
3. http://www.club-44.ch/media-c44/c44-p-53315f6501d43.pdf [↑](#footnote-ref-3)
4. http://eyecomresearch.com/eyetrackingresearch/eye-tracking-as-a-medical-training-device/ [↑](#footnote-ref-4)
5. http://www.smivision.com/fileadmin/user\_upload/downloads/case\_studies/cs\_smi\_medicaldiagnosis.pdf [↑](#footnote-ref-5)
6. http://www.tobii.com/fr/eye-tracking-research/global/research/usability/ [↑](#footnote-ref-6)
7. http://en.wikipedia.org/wiki/Helmet-mounted\_display [↑](#footnote-ref-7)
8. http://www.engadget.com/2014/01/03/tobii-steelseries-eye-tracking-game-accessory/ [↑](#footnote-ref-8)
9. Appelé Interactor dans le jargon Tobii [↑](#footnote-ref-9)
10. http://www.levisiteurdufutur.com/ [↑](#footnote-ref-10)
11. http://en.wikipedia.org/wiki/Windows\_Forms [↑](#footnote-ref-11)
12. http://en.wikipedia.org/wiki/Graphics\_Device\_Interface [↑](#footnote-ref-12)
13. http://qt-project.org/ [↑](#footnote-ref-13)
14. http://en.wikipedia.org/wiki/Windows\_Presentation\_Foundation [↑](#footnote-ref-14)
15. http://developer.tobii.com/community/forums/topic/design-of-gui-buttonobject-sizes/ [↑](#footnote-ref-15)
16. http://fr.wikipedia.org/wiki/Indice\_et\_distance\_de\_Jaccard [↑](#footnote-ref-16)