Application mobile : Visualisation 3D de la cornée

Alexandre Nicaise

2 septembre 2015

Table des matières

1 Introduction

1.1 Département Informatique et Recherche Opérationnelle

Fondé en 1966, le Département Informatique et Recherche Opérationnelle (DIRO) fut le premier département d'informatique créé au Québec et le troisième au Canada. Le DIRO est situé dans le pavillon André Aisenstadt qui fait partie de l'Université de Montréal (UdeM). L'UdeM a été fondée en 1978 et a été la première Université francophone de Montréal. Elle fait partie des quatre établissements supérieur de Montréal au Québec. Selon la firme QS (Quacquarelli Symonds), l'université de Montréal se classe au 33e rang des meilleures universités du monde en recherche opérationnelle. Elle fait également belle figure en informatique, prenant place parmi le groupe de tête constitué de 150 universités d'excellence.

Durant ce stage, j'ai pu intégrer l'équipe du DIRO dirigé par Jean Meunier. Il s'intéresse à l'analyse et au traitement numérique d'images et de vidéos dans un contexte médical. Ici nous allons plutôt nous intéresser à l'œil et plus précisément à la cornée.

1.2 Contexte Biologique

1.2.1 L'œil

L'œil est l'organe de la vision, c'est à dire qu'il capte la lumière afin que le cerveau puisse l'analyser et ainsi interagir avec son environnement. Il possède trois membranes opaques (sclérotique, choroïde et rétinienne) et quatre milieux transparents (cornée, humeur aqueuse, cristallin et humeur vitrée) (Figure : 1). Il est aussi protégé par plusieurs structures annexes comme les paupières, les cils et les sourcils.

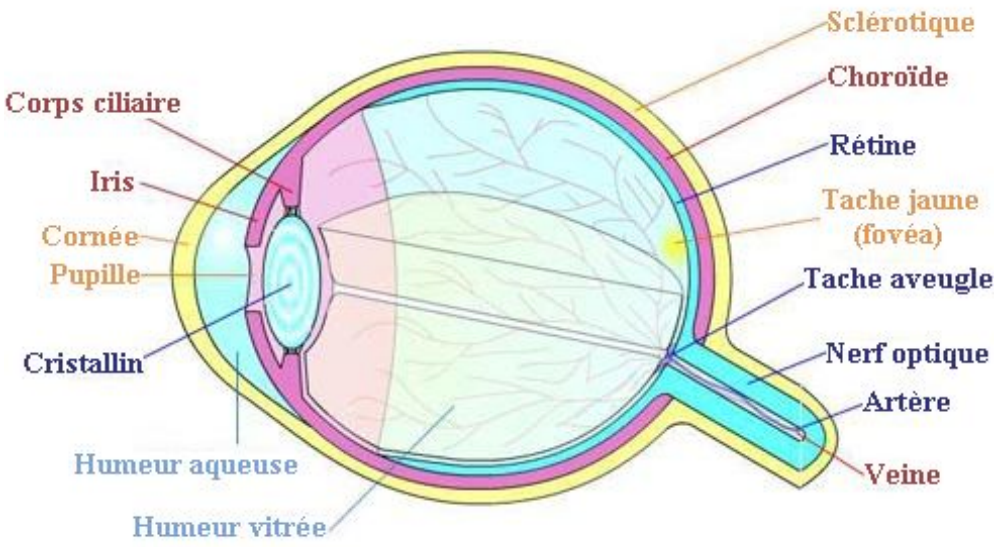

FIGURE *1 – l'oeil*

L'œil est souvent comparé à un appareil photo. Il est constitué d'un système optique (un objectif) et d'une surface sensible à la lumière. Le système optique est constitué de

la cornée et du cristallin. Il permet de capter les rayons lumineux provenant d'un objet source et de les dévier afin qu'ils convergent en un même point, idéalement situé dans le plan de la surface photosensible. Cette surface est située au sein de la rétine et est recouverte de cellules appelées photorécepteurs. Les photorécepteurs sont soit des cônes permettant la vision diurne, soit des bâtonnets permettant la vision nocturne. Ensuite nous quittons le domaine de l'appareil photo et passons dans le traitement de l'information. Les photorécepteurs créent un influx nerveux en réaction à leur exposition aux rayons lumineux. L'influx nerveux passe dans le nerf optique pour atteindre l'aire visuelle du cerveau. Le cerveau se charge ensuite de déchiffrer l'information contenue dans l'influx nerveux afin de permettre l'interaction avec l'environnement. (Figure : 2)

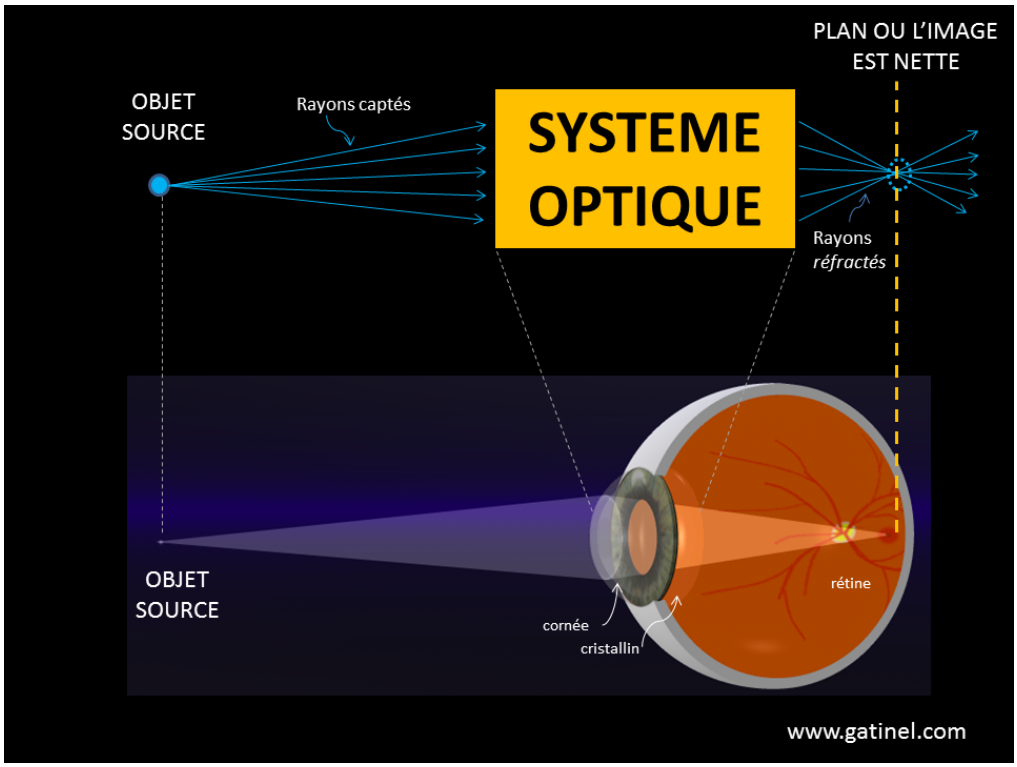

FIGURE 2 – *Présentation du mécanisme de la vision*

Ici nous nous intéresserons plus à un des éléments du système optique : la cornée.

1.2.2 La cornée

La cornée est un composant oculaire essentiel au fonctionnement de la vision : elle est la première structure que rencontre la lumière qui pénètre l'œil. Son rôle principal est de faire converger les rayons lumineux incidents vers le cristallin, avant de rencontrer la rétine et d'enclencher la cascade visuelle en créant l'influx nerveux. La cornée assure les deux tiers du pouvoir optique (focalisation de la lumière) des structures oculaires, le tiers restant étant dévolu au cristallin. Ce pouvoir optique découle de deux caractéristiques de la cornée :

- une courbure plus importante que celle du cristallin non accommodé. Les accommodations du cristallin sont des modification effectué sur celle-ci afin de modifier la réfraction du cristallin et ainsi amélioré la netteté de l'image.

- le contact avec l'air ambiant, offrant la plus grande différence d'indice de réfraction aux rayons lumineux incidents par rapport au cristallin qui est au contact d'un liquide et donc avec un indice de réfraction moindre.

La cornée est un tissu non vascularisé et transparent, de géométrie courbe. Elle est semblable à une coupole hémisphérique au contact direct de l'air. Elle couvre environ un cinquième de la surface de l'œil et a un diamètre d'environ 11 mm. La cornée est légèrement plus épaisse en périphérie (0.6 mm) par rapport à son centre (0.5 mm). Pour qualifier l'épaisseur, j'utiliserai le terme de pachymétrie. Le rayon de courbure est d'environ 6.5 mm pour la surface postérieure et varie de 7 à 9 mm pour la surface antérieure. Pour permettre à l'information lumineuse d'être transmise à la rétine sans déperdition qualitative et quantitative trop préjudiciable!!simplifie la phrase, prend des synonymes moins prise de tête!!, la cornée doit demeurer transparente, et conserver une certaine régularité de courbure. Seulement 4% à 6% de la lumière incidente est réfléchie par la surface cornéenne. Cette propriété de réflexion est mise à profit pour la réalisation de l'examen de topographie cornéenne.

1.2.3 Topographie de la cornée

La topographie permet de recueillir des informations relatives à la courbure ou au relief (élévation) de la cornée, grâce à la projection et l'analyse du reflet d'un motif lumineux éclairant ou balayant la cornée. Les images recueillies sont analysées de façon automatisée!!sont automatiquement analysées!! par un logiciel!!precise le logiciel ou vire par un logiciel!!, et des cartes en couleurs sont fournies au praticien pour interprétation. Elle!!rapel topographie!!est courante lors d'un examen ophtalmologique, pour réaliser un diagnostic ou un suivi. En effet, des déformations peuvent être liées par exemple, à des pathologies, des traumatismes, une chirurgie, ou simplement à l'âge. Dès lors, la possibilité d'estimer et de caractériser cette déformation permet d'apporter une information pertinente au médecin pour aider à son diagnostic.

Lors du!!de mon!!stage, j'ai utilisé des données obtenues via l'Obscan II!!le logiciel/la machine Obscan!! (Figure 2), un appareil qui permet les mesures!!de mesurer!! d'élévation de la cornée. Il est capable de mesurer la partie antérieur ainsi que la partie postérieur de la cornée avec une marge d'erreur de l'ordre du micron. Les données peuvent se présenter sous la forme d'une grille 101x101 contenant les valeurs d'élévation et!!rm et!! qui ont été prises toute les 0.1 mm. A partir de cette matrice d'élévation, il est possible de construire des maillages qui seront utilisés par la suite pour la visualisation en 3 dimensions de la cornée.

La cornée étant presque sphérique, un moyen simple et efficace de visualiser l'aspect de sa surface est d'utiliser une référence sphérique afin d'étudier ses différences par rapport à une sphère.!!differentie etude et visualisation!!

Afin d'affiner leurs diagnostics les praticiens ont besoin de mieux connaître l'aspect de la surface de la cornée. Pour cela, on utilise le fait qu'elle soit presque sphérique afin de la comparer avec une sphère en faisant la différence des deux!!couper phrase!!. Pour cela,

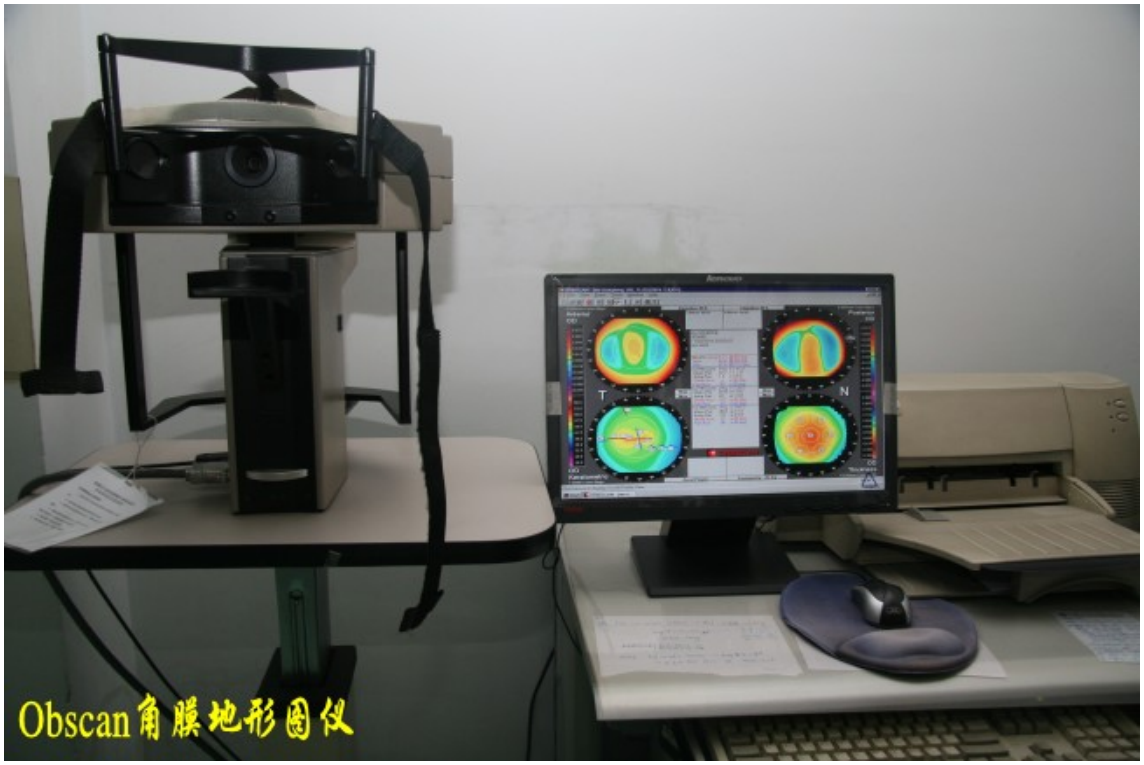

FIGURE *3* – *Obscan II*

on calcule donc la "Best Fit Sphere" (BFS) qui est la sphère qui correspond le mieux à la matrice d'élévation, puis on effectue la différence entre les deux surfaces. Cette différence va permettre de construire une carte semblable à celles utilisées par les géologues pour représenter les reliefs de la terre. En effet, afin de mettre en évidence ces "reliefs" ces différences sont représentées selon un jeu de couleur standard. Les différences positives seront associer des couleurs chaudes (points à l'extérieur de la BFS) et les différences négatives (points à l'intérieur de la BFS) à des couleurs froides. Il est nécessaire de calculer deux BFS différentes, une pour la face antérieur et une pour la face postérieur.

La BFS est calculée en minimisant la somme des distances au carré entre la sphère et chaque point de la cornée, ce qui donne l'expression suivante à minimiser :

$$f(c, R) = \sum_{k=1}^{n} \sqrt{(x_k - x_c)^2 + (y_k - y_c)^2 + (z_k - z_c)^2} - R^2$$

Avec k un point de la cornée, c le centre de la BFS, R son rayon et n le nombre de point de la cornée.

L'association d'image représentée par la figure 5 permet la visualisation des différentes étapes de construction de la carte topographique de la cornée.

1.3 Problématique du stage

Le but de mon stage a été de construire une application mobile permettant de visualiser une cornée en trois dimensions. Ce projet a été accueilli à bras ouvert !!ca se dit pas dans un rapport !! par les praticiens de l'hôpital. En effet, cela faciliterais l'accès aux différentes cartes représentant les cornées de leurs patients. Avec une simple clique il

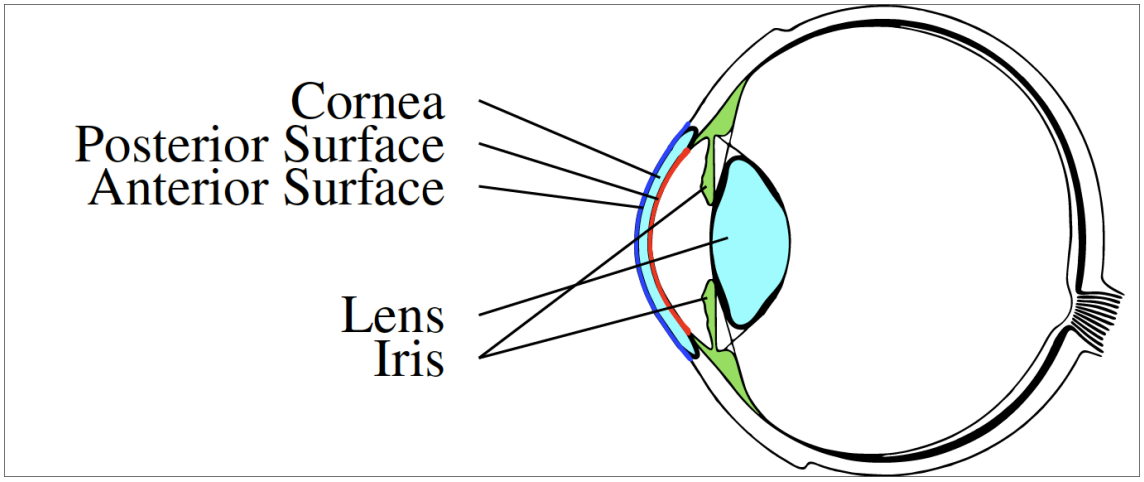

FIGURE *4 – Section d'un œil : la surface antérieur de la cornée est représenter en bleu foncé, la face postérieur en rouge et la pachymétrie en bleu clair*

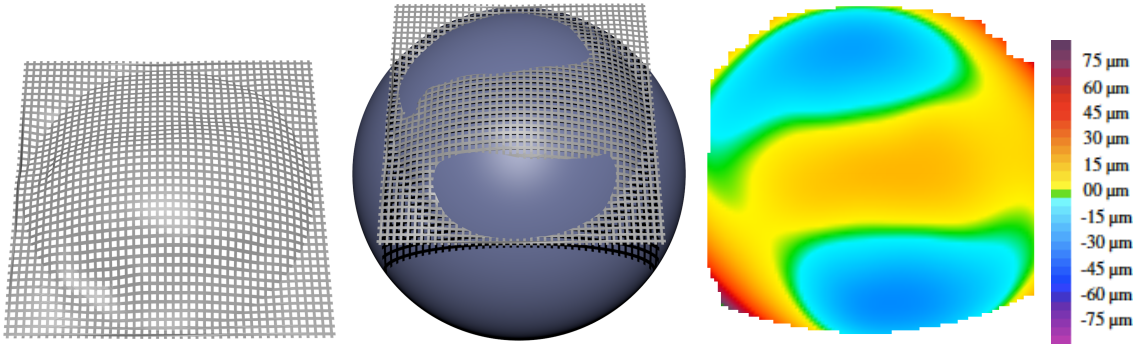

FIGURE *5 – Présentation de la construction de la carte d'élévation d'une cornée. A gauche nous pouvons voir la matrice 101x101 obtenue par l'obscan. L'image suivante est la superposition de la BFS (Best Fit Sphere) et de cette matrice. La troisième image est l'ajout des couleurs afin de visualiser les relief de la cornée selon une palette de couleur située juste à côté. On peut voir que les couleurs froides sont situées en-dessous de la BFS et les couleurs chaude au-dessus. Le vert symbolise l'égalité entre la BFS et la cornée.*

serait toucher du doigt!!!!!peut pas corriger les fautes sur ce secteur!!!!! il leur serais possible d'accéder au dossier du patient et donc à l'affichage de leur cornée. Le nombre de photocopie serais donc limitée!!comprend toujours pas!! et ils pourraient la transporter n'importe où. Ce projet a été débuté l'année dernière par Ouri Saban un étudiant du master CCI. Mon but a été d'utiliser une bibliothèque appelé Visual Tool Kit!! (VTK) vire la suite!!!, qu'on nommera VTK par la suite, afin de visualiser la cornée sur des plateforme mobile.!!!presente pas le plan si c'est juste pour dire mqt met res et discu!! Tout d'abord, je présenterais le matériel et les méthodes que j'ai utilisé tout au long de mon stage, puis je continuerais par les résultats et je finirais par la discussion et la conclusion.

2 Matériels et méthodes

2.1 Données disponibles

Pour mes tests, je me suis servi d'un document texte fournis par Arnaud Polette ! !fusionne les 2 phrase ! !. Il contient toutes les informations de la cornée obtenue par le biais de l'obscan. Elles sont sous la forme de plusieurs matrice 101x101 contenant les valeurs d'élévations et dont les valeurs de X et de Y sont situées de -5 à +5 mm séparé par des pas de 0.1 mm.

Ces matrices permettent de visualiser :

- la face postérieur
- la face antérieur
- la différence entre la postérieur et la BFS
- la différence entre la antérieur et la BFS
- la pachymetry
- la tangentiel de la face antérieur
- la tangentiel de la face postérieur

Je ne me suis servis que des matrices représentant la face antérieur, postérieur, la BFS, la pachymetry. En plus de ces données, ces fichiers informent sur le rayon et le centre des sphères utilisées comme BFS. Nous avons donc tout les éléments nécessaire à la visualisation en trois dimension de la cornée. ! !reformuler, langage pas adapté a un rapport M2 ! !

2.2 VTK (Visualisation Toll Kit)

Pour effectuer la visualisation en trois dimensions, j'ai décidé d'utiliser un logiciel très utilisé par les chercheurs : VTK (Visual Tool Kit) créé par Kitware. C'est un logiciel open-source permettant l'infographie 3D ! ! !met trois dimensions partout ou 3D partout ! !, le traitement d'image et la visualisation. Il est constitué d'une librairie de classe en C++ et plusieurs couches de surfaces interprétées incluant le Tcl/Tk, Java, et Python. En effet, nous avons été séduit par une application open-source utilisant VTK appelé KIWIVIEWER fonctionnant sur android et ios. Pour le portage ! !ca se dit ca ? ! ! sur android, j'ai aussi utilisé les exemples mises à disposition sur git-hub. Pour apprendre à utiliser VTK, un ensemble d'exemples est mis à disposition ainsi qu'un wiki ! ! non pas wiki.. ! ! bien structuré. ! ! paragraphe peut être mieux présenté ! !

Pour la visualisation j'ai utilisé les éléments de vtk suivants :

- vtkPoints → stocke les coordonnées obtenue par le biais des matrices d'élévation
- vtkFloatArray → affecte un "scalar" à une coordonnée. Un scalar est une valeur qui permettra la représentation sous forme de couleurs. Ici nous utiliserons les valeurs d'élévations.

- vtkCellArray → construit le maillage. Le maillage est la représentation d'une surface (ici la cornée) en la subdivisant en un ensemble de polygones. Il est composé de sommets, connectés les uns aux autres par des faces ou facettes de forme polygonale. Lorsque toutes les faces sont des triangles, on parle de maillage triangulaire (trimesh), ou de triangulation selon les domaines. Les maillages par quadrilatères sont aussi très courants. En 3 ! ! trois/3D ! ! dimensions, il est aussi possible d'utiliser des maillages volumiques, qui relient les sommets par des tétraèdres, des hexaèdres et des prismes. Ici, j'ai utilisé le maillage triangulaire.
- vtkPolyData → regroupe les objets vtkPoints, vtkFloatArray et vtkCellArray.
- vtkPolyMapper → permet la transformation de la structure de données contenue dans vtkPolyData en primitives OpenGL. Les primitives OpenGL sont des points, des lignes ou des triangles. Il permet aussi l'association de couleurs qui peut être contrôler en spécifiant une table de couleurs (lookup table) et un intervalle.
- vtkActor → sert à positionner et orienter l'objet, à le colorer et choisir des propriétés graphiques.
- vtkRenderer → lance les algorythmes nécessaire au rendu de l'objet
- vtkRenderWindow → spécifie une fenêtre qui affiche le rendu
- vtkRenderWindowInteractor → permet l'intéraction avec l'objet en faisant bouger la caméra.

La figure 6 permet de visualiser la construction du mapper ! !angliscisme ! ! à partir de la matrice d'élévation.
Pour utiliser VTK, nous pouvons utiliser plusieurs langage tel que le le java, le python, le C, le C# ou encore le C++. Étant donné que les exemples mis à disposition pour android sont codé en C++, j'ai donc utilisé ce langage afin de créer mon propre programme.

2.3 Langage de programmation : C++

Le C++, descendant du C, est un langage de programmation compilé apparu en 1983 et ayant subit plusieurs révision depuis, dont la dernière était en 2014. Avec lui, nous pouvons programmer sous les paradigme :

- procédurale → permet de décomposer le code en sous fonction pouvant être réutilisé plusieurs fois dans le programme voir s'appeler lui-même (récursivité)
- orienté objet → permet la représentation de principes réelle et de ces d'interactions via des objets informatique et ces méthodes.
- générique → permet la définition d'algorithme identique opérant sur des données de types différents.

Pour coder sous ce langage plusieurs choix étaient à ma disposition : un environnement de développement (IDE), un éditeur de texte et CMAKE (cross platforme make).

Dans un premier temps, j'ai utilisé un IDE (code : :block) qui est un ensemble d'outils permettant de faciliter le travaille du développeur. Il possède donc un éditeur de texte destiné à la programmation, des fonctions permettant de démarrer le compilateur ou

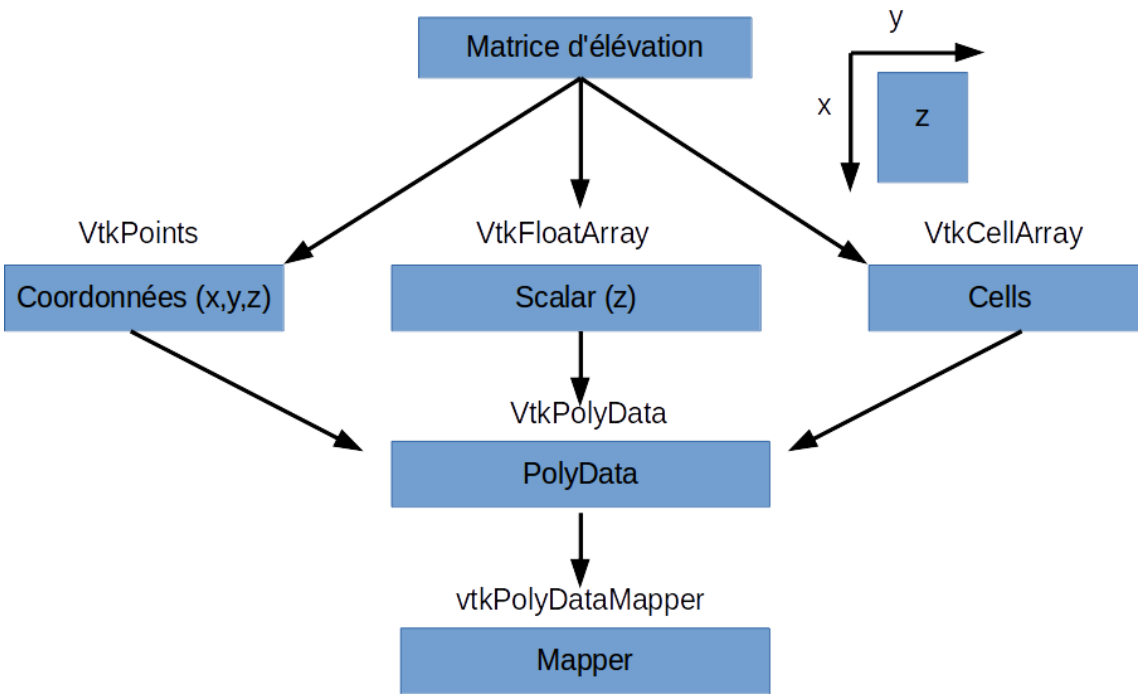

FIGURE 6 – *Construction de l'objet polydataMapper à partir d'une matrice d'élévation. Un schéma de la matrice est représenté en haut à gauche de la figure. A partir de cette matrice 3!!trois!! éléments sont extraits : les coordonnées stockées dans un vtkPoints, les valeurs qui seront associées à une couleurs stockée dans un vtkFloatArray et les cellules représentant le maillage dans un vtkCellArray. Ces 3!!trois!! éléments sont ensuite associés dans un vtkPolyData qui servira ensuite à construire le mapper.!!peut être plus clair!!*

l'éditeur de lien par simple pression sur un bouton et un débogueur en ligne. Le débogueur en ligne permet d'exécuter le programme en construction ligne par ligne.

Dans un deuxième temps, j'ai donc utilisé un éditeur de texte (Sublime texte) couplé à CMAKE. Sublime texte est un éditeur de texte permettant une auto-compilation ainsi qu'une palette de couleur pour plusieurs langages de programmation tel que le C++. CMAKE est un logiciel de compilation multiplateforme. CMAKE va utiliser des fichier appeler "CMAKELists.txt", commun pour toutes les plateformes, afin de générer des makefiles. Les makefiles sont utilisés par le programme make pour exécuter un ensemble d'actions, comme la compilation d'un projet, l'archivage de document, la mise à jour de site, etc ... Ici on l'utilise pour la compilation du projet en C++.

Comme nous l'avons vu plus haut, le C++ est un langage multiplateforme donc qui permet de construire des applications android.

2.4 Programmation mobile

3 Résultat

3.1 Les classes

Comme je l'ai dis plus haut, le C++ est un langage de programmation orienté objet. J'ai donc créé plusieurs classes lors de la construction de mon programmes.

Tout d'abord, nous avons les classes représentant les données de la cornées : DataCornee, ParserTopos et DataPachymetry.
DataCornee : représente les données d'une cornée en leur donnant un nom, comme "Surface anterior", et avec un tableau en deux dimension contenant les données d'élévation de la cornée.
ParserTopos : représente les données contenue dans un fichier topos (cf plus haut) grâce à une liste d'objet "DataCornee". Chaque matrice de données peut être appelée séparément avec leur nom, leur numéros ou par des méthodes. Cet objet permet aussi d'afficher les noms des données et contient les rayons des différentes BFS et les coordonnées de leur centres respectifs. Il stocke aussi les coordonnée réelle utilisé en x et en y (de -5 à 5mm par pas de 0.1).
DataPachymetry : reprend les données contenues dans topos afin de reconstruire les données de la pachymétrie. Pour cela, je lui donne soit deux matrices représentant les surfaces antérieures et postérieures de la cornée soit l'objet ParserTopos. Il construit ensuite la matrice de coordonnée qui sera utilisée dans la visualisation de la pachymétrie. Nous verrons cela un peu plus tard.

Ensuite, nous avons les classes permettant la visualisation de la cornée en utilisant VTK : ColorElevationMap, SurfaceElevationMatrice, SurfaceScalarElevationFromMatrice, PachymetryVisualisation, PachymetryVisualisationFromMatrice, VolumeVisualisation.
ColorElevationMap : permet de stocker les objets VTK utiliser pour la visualisation (vtkPoints, vtkCellArray, vtkFloatArray, vtkPolydata, vtkPolyDataMapper, vtkActor). Elle contient aussi les méthodes nécessaires à la construction des cellules utilisées dans le maillage. Cette classe est la principale, c'est à dire qu'elle sera héritée par les autres classes utilisant VTK.
SurfaceElevationMatrice : hérite de ColorElevationMap, elle permet la visualisation d'une surface via une matrice d'élévation (surface antérieur ou postérieur de la cornée). Dans cette classe, on ajoute des méthodes permettant d'extraire les coordonnées et les "scalars" à partir d'une matrice car celles-ci ne sont pas présentes dans ColorElevationMap.
SurfaceScalarElevationFromMatrice : hérite de SurfaceElevationMatrice, elle permet la visualisation d'une surface via 2!!deux!! matrices. La première est une matrice d'élévation, la deuxième représente les valeurs des différences entre la BFS et sa surface associé. Dans cette classes, j'ai fais en sorte d'adapter une des méthodes de la classe hériter!!heriter?!!!afin qu'elle prenne en compte les deux matrices au lieu d'une.
VolumeVisualisation : hérite de ColorElevationMap, elle permet de créer un objet de

VTK permettant de donner une impression de volume. Dans cette classe on ajoute toutes les méthodes nécessaires à la construction du volume (cf :plus loin).
PachymetryVisualisation : hérite de ColorElevationMap, elle permet de visualiser les données de l'objet DataPachymetry.
PachymetryVisualisationFromMatrice : hérite de ColorElevationMap, permet de visualiser la pachymétrie de la cornée contenue dans un objet ParserTopos.

J'ai aussi créé d'autres classes dont certaines représentent des notions tel que les points, les triangles ou encore les vecteurs. Les autres classes ne représentent pas des objets mais contiennent un ensemble de méthodes utiles dans certain cas. Par exemple, j'ai construit une classe UtilsVector qui permet d'avoir accès à des fonctions affichant des vecteur en 2!!deux!! dimensions, en 1!!une!!dimension, de float, de int, ...

3.2 visualisation des faces antérieur et postérieur

3.3 Le volume

3.4 La pachymétrie

4 Discussion

4.1 Organisation de l'application

4.2 Difficulté de la programmation mobile

Séduit par cette application, je l'ai étudié dans l'espoir de pouvoir y baser ma propre application. Malheureusement, cette application n'était plus mise à jours régulièrement. En effet, VTK a décidé d'arrêter son développement afin d'inclure la partie android et ios dans VTK en lui même. Donc une grande partie de mon stage a été de trouver comment porter VTK dans une application android. Peut de documentation peut être trouvé sur ce sujet. ! !paragraphe a reformuler ! !

4.3 Environnement utilisé pour coder

J'ai préféré utiliser l'association entre Sublime text et CMAKE. En effet, les exemples de VTK utilisaient déjà CMAKE donc leur "CMAKELists.txt" ont pu me servir de base à la construction de celui de mon projet. De plus, étant débutant en C++ j'ai trouvé que cette manière de faire me permettais de mieux comprendre la construction d'un projet en C++.

5 Conclusion