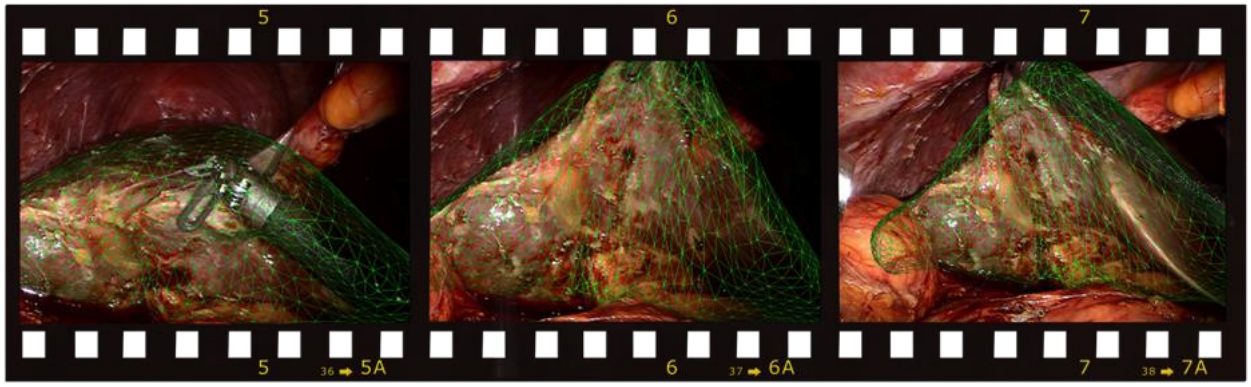
Delberghe Calvin

Mboup Papa Momar

2013/2014

Rapport de Projet de Fin d'Etudes

IMA5SC



Réalité augmentée pour la chirurgie

Tuteurs: **Jeremie Dequidt, Nazim Haouchine**

POLYTECH LILLE

REMERCIEMENTS

Nous tenons tout d'abord à remercier nos deux tuteurs Mr DEQUIDT et Mr HAOUCHINE pour toute l'aide qu'ils ont pu nous apporter, et les conseils précieux qu'ils on pu nous prodiguer durant ce projet.

Nous tenons également à remercier tous les professeurs du département IMA pour ces trois années d'études, pour leurs disponibilités et leurs conseils apportés durant notre cursus de formation d'ingénieur.

De même, nous remercions notre secrétaire et le chef du département respectivement Madame HIGUINEN et Monsieur ROJO.

Sommaire

Introduction

Notre projet de fin d'étude se porte sur le domaine de la médecine et plus spécialement dans le cadre d'opérations chirurgicales abdominales.

Dans ce rapport, vous trouverez tout d'abord l'explication détaillée du projet, son contexte ainsi que la problématique posée. Ensuite, nous vous exposerons notre planning ainsi que la gestion du projet. Dans la troisième partie, nous vous présenterons les différentes réalisations effectuées ainsi que l'explication du programme. Enfin, un bilan personnel sera présenté de même qu'un récapitulatif par rapport aux objectifs de départ du projet.

I - Le projet

A- Contexte et problématique

Comme précisé en introduction, notre projet consiste à assister les chirurgiens lors de chirurgies de l'abdomen, dites laparoscopiques. La chirurgie laparoscopique est une technique chirurgicale minimalement invasive où le chirurgien réalise une opération de l'abdomen par de petites incisions où sont placés des trocars qui permettent le passage de fins instruments chirurgicaux. Dans le cadre de ce projet, il s'agit de réaliser l'ablation de la(des) tumeur(s) présente(s) dans le foie du patient. La localisation des tumeurs ainsi que de l'ensemble du réseau vasculaire peut etre calculé grâce à un scanner préopératoire du patient. Cependant, ils ne peuvent pas être visualisés par la caméra lors de l'opération chirurgicale. Ainsi, le chirurgien ne peut en aucun cas connaitre la localisation de la tumeur ainsi que sa profondeur par rapport à la surface du foie.

Figure 1: Chirurgie laparoscopique

B- Mission

La mission qui nous a été confiée est d'aider les chirurgiens à mieux repérer la(les) tumeur(s) par rapport à la surface du foie durant ces opérations grâce à la réalité augmentée. Pour celà, des travaux récents de l'équipe de recherche Shacra ont permis de suivre avec une bonne précision le mouvement de structures internes du foie pendant une opération laparoscopique. Leur objectif consiste à utiliser ces travaux pour mélanger le flux vidéo fourni par l'endoscope(caméra) aux informations calculées par la simulation(représentation du foie et la tumeur en 3D) pour fournir aux chirurgiens des informations utiles sur la position des structures internes du foie lors de la manipulation.

La mission sera de proposer des solutions d'affichage de la distance séparant le foie de la tumeur. Différentes pistes pourront être explorées pour ne pas surcharger le flux vidéo tout en fournissant les informations importantes (position, profondeur, distance à la surface du foie...). La fusion de la vidéo et de notre application ne fait pas partie des objectifs du projet. En effet, elle sera réalisée par l'équipe de Shacra.

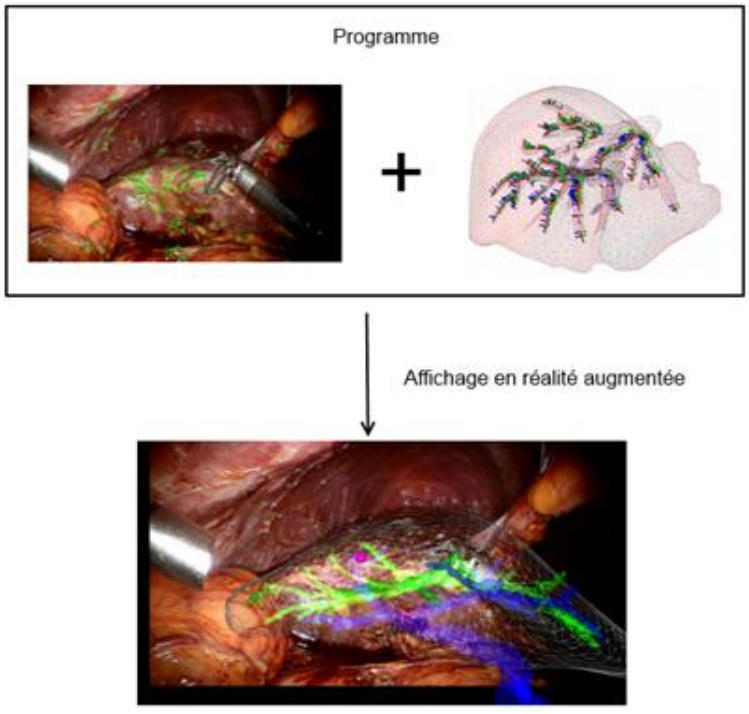


Figure 2: Exemple de vidéo + modélisation 3D

II. Gestion de projet et cahier des charges

Le projet se décompose en deux parties: l'une pour le premier semestre et la deuxième partie pour le second semestre.

Le cahier des charges du projet qui nous a été confié était le suivant:

- Objectif: Développement d'outils de réalité augmentée pour des opérations de chirurgie abdominale.
- Réaliser une application permettant l'affichage d'un foie et d'une tumeur à l'aide de données existantes.
- Trouver une solution créative afin de représenter la distance séparant la tumeur de la surface du foie en temps réel.

L'objectif qui nous a été fixé pour le premier semestre était de réaliser la base de l'application permettant un affichage statique en 3D de données à un instant donné, et ensuite d'implémenter une méthode afin de calculer la distance séparant la tumeur de la surface du foie dans les trois directions orthogonales X, Y, et Z.

Nous nous sommes donc répartis les tâches. Après avoir suivi des tutoriels d'apprentissage, l'un de nous a pris en charge la classe principale de notre programme, se chargeant de mettre en lien les différentes classes et méthodes autorisant l'affichage des données, tandis que l'autre s'est occupé de coder les méthodes spécifiques permettant la récupération des données des fichiers, et de calculer la distance entre la tumeur et la surface du foie.

Pour la deuxième partie du projet, il s'agissait tout d'abord de terminer les objectifs du premier semestre ensuite proposer différentes solutions d'affichage de la distance entre la tumeur et la surface du foie.

Vous pourrez trouver en annexe un diagramme prévisionnel GANTT que nous avions réalisé pour la seconde partie du projet. Vous pourrez noter des différences entre le diagramme prévisionnel GANTT et le diagramme UML présenté plus loin, notamment sur les différentes classes que nous avons implémentées (ou non), qui sont dues à des choix que nous avons réalisés pendant le déroulement de la seconde partie du projet.

III. Réalisation

A- Choix des technologies

Pour la bonne réalisation de notre projet, il nous a été indiqué dans le cahier des charges que nous devrions travailler dans le langage C++ en utilisant la librairie OpenGL. OpenGL permet à un programme de déclarer la géométrie d'objets sous forme de points, de vecteurs, de polygones, de bitmaps et de textures. OpenGL effectue ensuite des calculs de projection en vue de déterminer l'image à l'écran, en tenant compte de la distance, de l'orientation, des ombres, de la transparence et du cadrage.

Afin d'apprendre à utiliser OpenGL, nous avions à notre disposition l'adresse d'un site contenant des tutoriels sur l'utilisation d'OpenGL. Cependant, ces tutoriels dataient déjà de quelques années et le site n'avait plus été actualisé depuis quelques temps. De plus ces tutoriels utilisent une ancienne version d'OpenGL. Nous avons donc décidé de rechercher des tutoriels nous proposant un apprentissage d'une version plus actuelle de cette librairie. Nous avons trouvé sur le site OpenClassrooms (anciennement connu comme LeSiteDuZero) un tutoriel aussi bien détaillé sur les différentes fonctionalités d'OpenGL que sur le précédent site, mais proposant cette fois la version la plus récente de cette librairie.

Le choix de l'IDE sous lequel travailler s'est fait tout naturellement durant la réalisation des tutoriels, qui proposaient d'utiliser le logiciel CodeBlocks qui est très simple d'utilisation et qui possède tout ce qui nous intéresse afin de mener à bien notre projet. De plus le tutoriel expliquait les bases extensives du fonctionnement de cet IDE, nous permettant ainsi d'être plus productifs lorsque nous avons commencé à coder notre programme.

B- Affichage du foie et de la tumeur en 3D

Pour représenter une structure en 3D, nous allons utiliser un ensemble de triangles. Chaque triangle est constitué de 3 vertices qui sont ses sommets. Un vertex est tout simplement un point en trois dimensions : x, y et z. Les vertices sont récupérées à partir de fichiers **.obj** . Un fichier OBJ est un format de fichier contenant la description d'une géométrie 3D. De fait, le but est de relier les trois sommets (vertices) formant un triangle et ensuite de relier tous les triangles pour obtenir au final notre structure en 3D.

Pour connaître quelles vertices associer entre elles, il nous faudra aussi l'ensemble des faces pour chaque structure à représenter. Ce qu'on appelle une face est en fait simplement une suite de 3 entiers qui indiquent quelles vertices (que l'on a récupéré du même fichier **.obj**) associer pour chaque triangle. Ainsi, l'ensemble constituera notre structure en 3D.

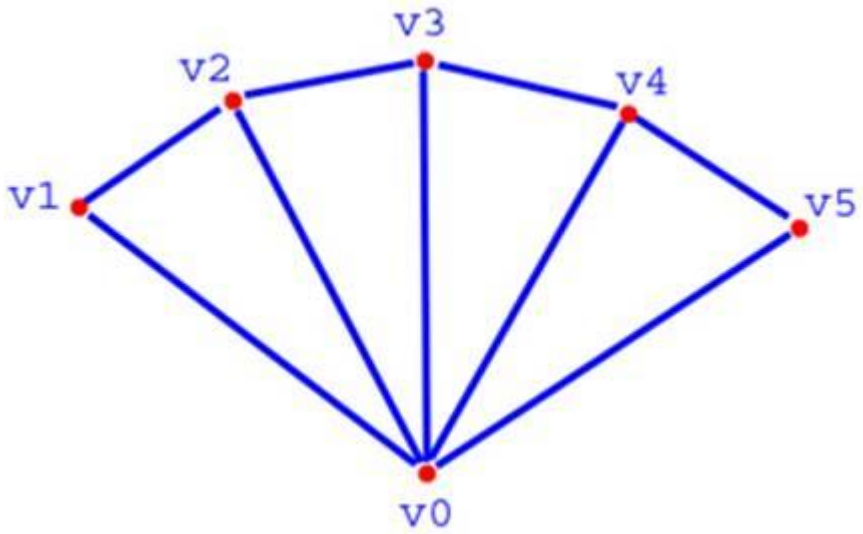


Figure 3: Triangles tracé à partir de vertices.

La principale difficulté rencontrée pour la modélisation des structures en 3D était celle du nombre de vertices du foie. En effet, nous avions un nombre de vertices dix fois plus élevé pour le foie par rapport au nombre de vertices de la tumeur que nous affichions sans problème. Il se produisait trop d'échanges de données entre la carte graphique (qui gère l'affichage) et la RAM (qui stocke les données), ce qui empèchait l'affichage du mesh du foie. Afin de pallier à ce problème, nous avons utilisé les VBO (Vertex Buffer Object).

Un VBO est un objet OpenGL qui contient des données relatives à un modèle 3D comme les vertices, les coordonnées de texture, les normales (pour les lumières), ... Sa particularité vient du fait que les données qu'il contient se trouvent non pas dans la RAM mais directement dans la carte graphique.

Les VBO permettent de gagner un temps considérable à OpenGL en évitant de faire des aller-retours inutiles entre la RAM et la mémoire graphique :

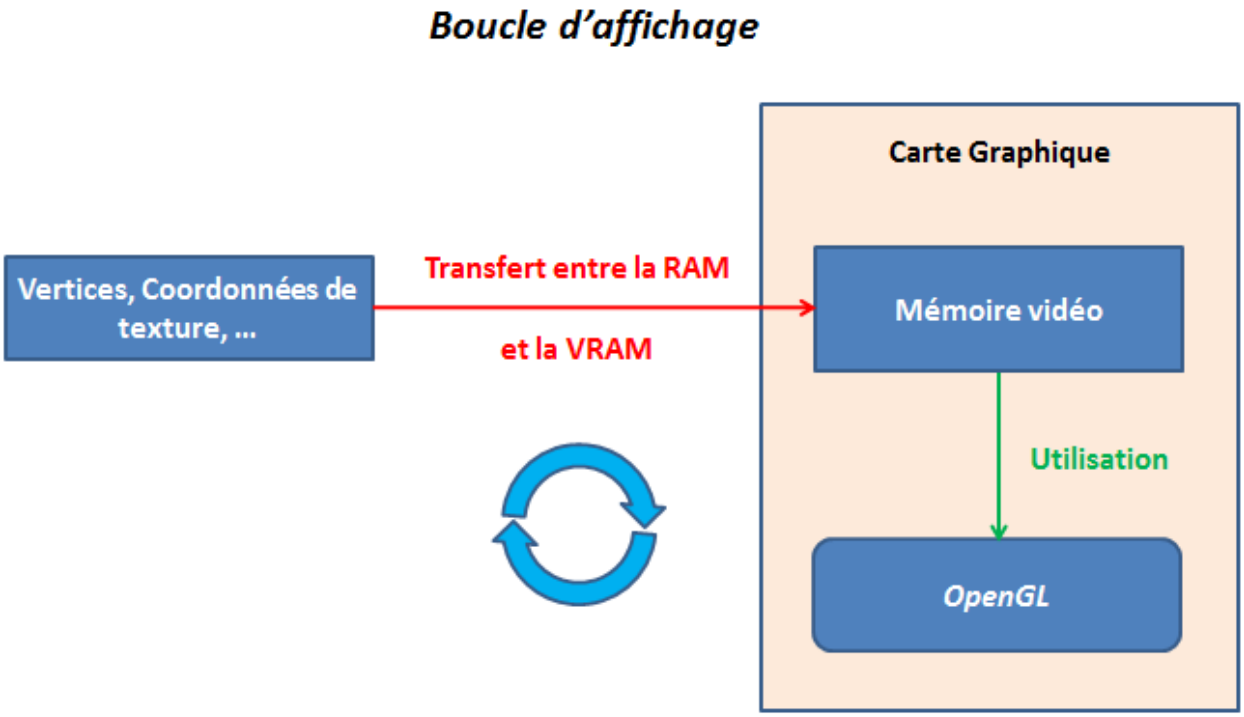


Figure 4 : Boucle d'affichage sans les VBO

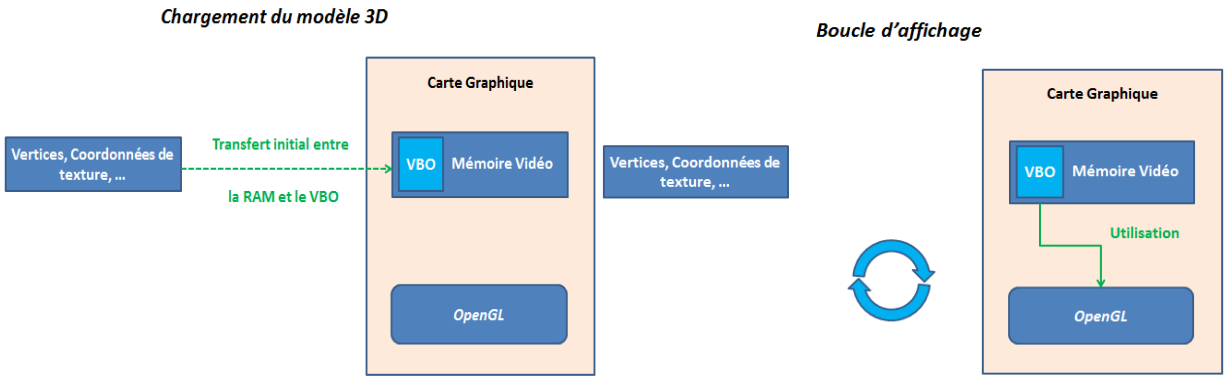


Figure 5 : Affichage avec les VBO

Nous avons grâce aux VBO pu réaliser l'affichage du foie sans plus de problèmes.

On peut voir en **Figure 6** le résultat de la représentation du foie et de la tumeur en même temps :

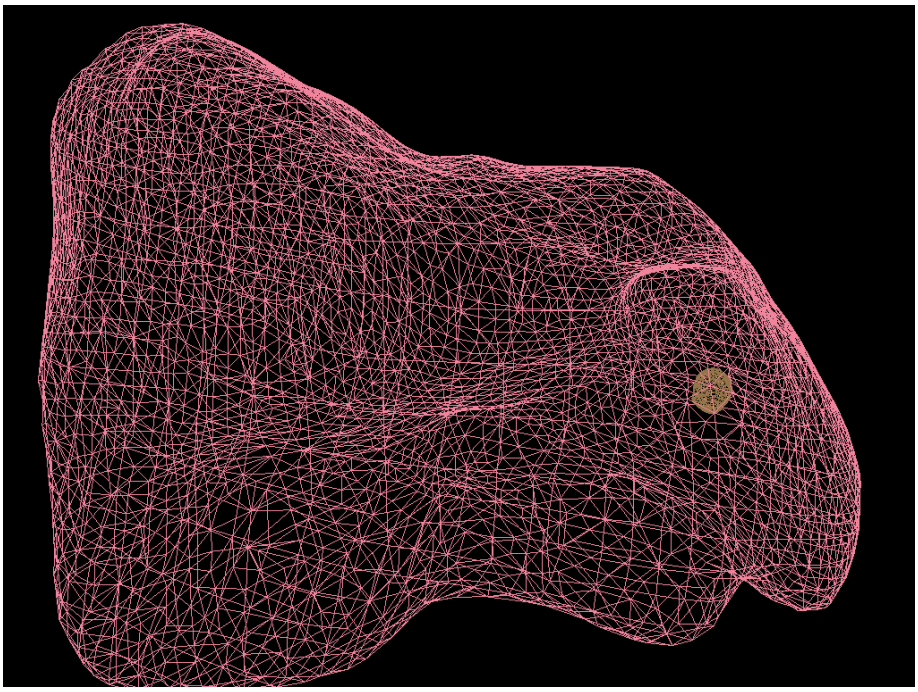


Figure 6: Le foie et la tumeur en 3D

En rose, on a représenté le foie et en marron la tumeur qui se trouve à l'intérieur de ce dernier.

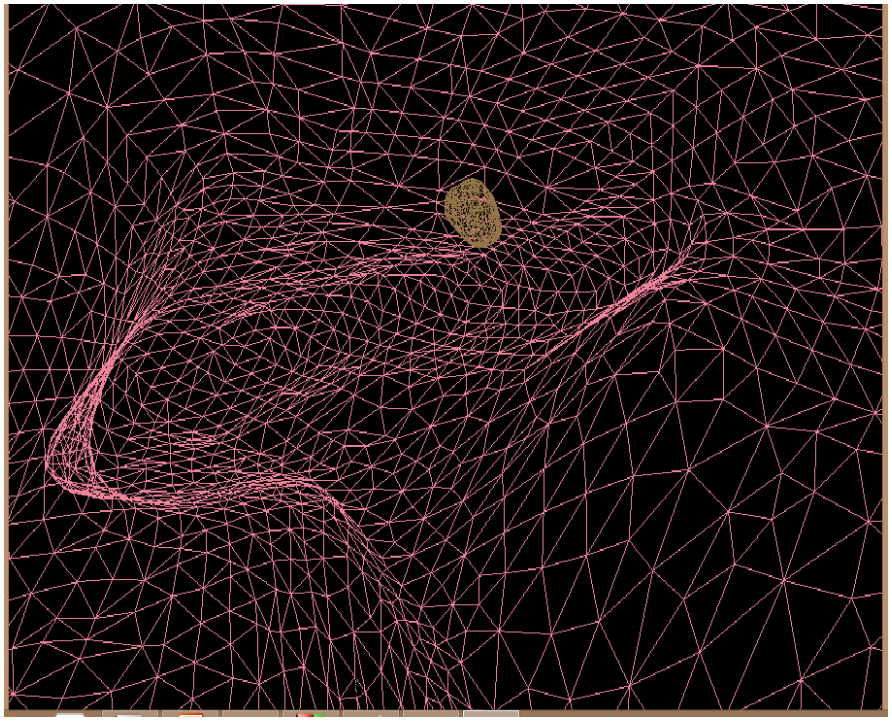


Figure 7: A l'intérieur du foie

Pour arriver à cette représentation, nous avons choisi de créer deux classes distinctes : **Liver.cpp** et **Tumor.cpp**.

On peut voir en **Figure 8** le diagramme UML de notre application avec les différentes classes utilisées par cette dernière.

Figure 8: Diagramme UML

Voici un bref résumé de chacune des classes et de leurs fonctions :

- La classe **Camera.cpp** s'occupe de gérer tous les placements et les orientations que la caméra va prendre.
- La classe **Input.cpp** s'occupe de gérer tous les évènements clavier et souris, les rendant plus simples d'utilisation.
- La classe **Shader.cpp** permet de charger et de compiler le shader en charge de l'affichage de couleurs sur notre représentation.
- La classe principale: **SceneOpenGL.cpp** qui met en relation les différentes classes, permet la création de la scène, et dans laquelle nous avons codé les différentes fonctions nécessaires au programme, notamment la fonction de calcul des distances.
- La classe **Texture.cpp** permet de charger des textures à la place des couleurs.
- La classe **Tumor.cpp** permet de récupérer les vertices de la tumeur et de l'afficher.
- La classe **Liver.cpp** permet de récupérer les vertices du foie et de l'afficher.
- La classe **DistanceDisplay.cpp** permet d'afficher les distances séparant la tumeur de la surface du foie suivant les trois axes x, y et z.

C- Calcul de la distance entre la tumeur et le foie

Afin de récupérer la distance séparant la tumeur et le foie, on réalise des tests d'intersection rayon-triangle avec les triangles composant les structures 3D du foie et de la tumeur. En effet, en réalisant ce test entre un rayon partant du barycentre de la tumeur suivant la direction *X*, *Y* ou *Z*, et les triangles constituant le foie d'une part, et la tumeur d'autre part, on obtient deux distances (toujours selon *X*, *Y*, ou *Z*). La première, distance entre la paroi du foie et le barycentre de la tumeur, la seconde, distance entre la paroi de la tumeur et le barycentre de celle-ci. En soustrayant la seconde à la première, on obtient la distance séparant la paroi du foie et la paroi de la tumeur selon un axe prédéfini (*X*, *Y* ou *Z* de notre repère). Il suffit ensuite de réaliser le calcul une fois pour chaque axe.

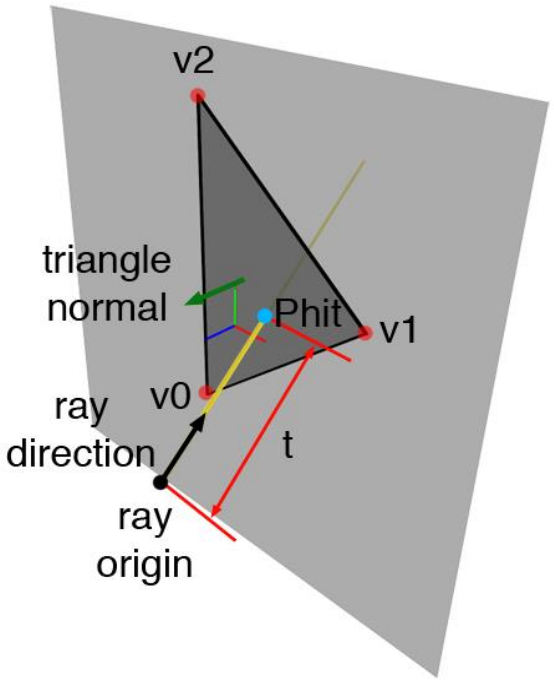


Figure 9: Point d'intersection rayon/plan du triangle

Ce calcul se réalise en plusieurs étapes mathématiques présentées ci-dessous :

Afin de savoir s'il y a intersection entre un rayon et un triangle, il est plus aisé de commencer tout d'abord par vérifier s'il y a intersection entre ce même rayon et le plan contenant le triangle.

Pour cela, on calcule le vecteur normal du plan. Le vecteur normal du plan s'obtient en faisant le produit vectoriel de deux vecteurs non colinéaires du plan. On connaît 3 points du plan (les points constituant le triangle en train d'être testé), que l'on nommera ici *v0*, *v1* et *v2*. On peut donc calculer le produit vectoriel de *v0v1* et *v0v2* par exemple, pour obtenir le vecteur normal *N(A,B,C)*. Une fois le vecteur normal obtenu, on réalise le produit scalaire entre *N* et le rayon. Si le résultat est nul, le plan et le rayon sont parallèles donc il n'y a pas intersection. Sinon on a confirmation que le rayon et le plan entrent en intersection.

Cela permet ensuite, si intersection il y a, de calculer les coordonnées du point d'intersection *P* entre le rayon et le plan. P est défini de cette façon:

$P = O + t * D$

-O l'origine (ici le barycentre de la tumeur)

-D le vecteur directeur du rayon

-t la distance entre O et P

L'équation du plan contenant le triangle est celle-ci:

$A * x + B * y + C * z + d = 0$

-*A, B* et *C* sont les composantes du vecteur normal au plan *N(A,B,C)*.

-*(x,y,z)* un point du plan.

-*d* la distance de l'origine (du repère) au plan, suivant la direction du vecteur normal au plan.

Grâce aux deux formules précedentes, on en déduit le calcul de t :

$t = - (N \bullet O + d) / (N \bullet D)$

Une fois *t* calculé, on peut aisément obtenir les coordonnées de *P*. On peut maintenant passer au deuxième test à réaliser, qui est le test d'appartenance d'un point à un triangle.

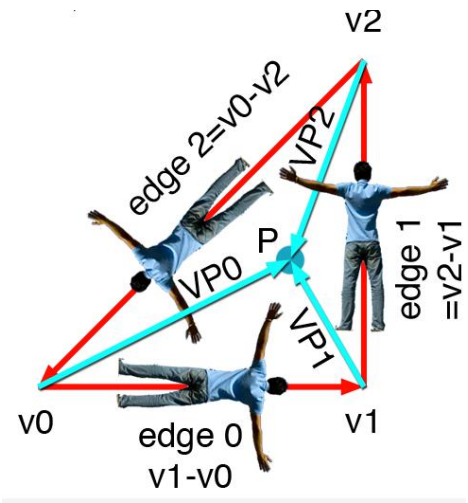


Figure 10 : Test d'appartenance au triangle

Le principe de ce test est le suivant:

On calcule les vecteurs partant des points du triangle et allant vers *P* : *v0P, v1P, v2P.*

On calcule de même les vecteurs correspondants aux côtés du triangle : *v0v1, v1v2, v2v0*.

On calcule le produit vectoriel de chacun de ces vecteurs avec les vecteurs contenant *P:*

u = v0v1 ^ v0P, v = v1v2 ^ v1P, w = v2v0 ^ v2P.

Si *P* appartient au triangle, les vecteurs *u*, *v* et *w* ainsi calculés seront colinéaires de même sens avec le vecteur normal au triangle N. (voir **Figure 11**)

On réalise donc les tests suivants: (produit scalaire)

N • u, N • v, N • w

Si le résultat est positif, P est placé à l'intérieur du triangle par rapport au coté testé. (u teste le coté v0v1, v teste v1v2, w teste v2v0)

Si les trois tests sont positifs, cela signifie que P est bien à l'intérieur du triangle.

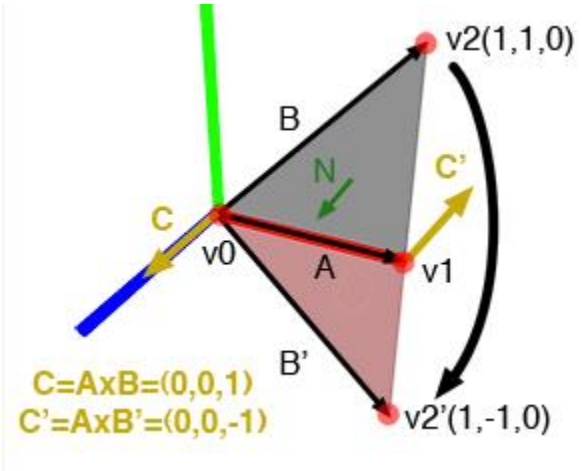


Figure 11 : Exemple représentatif du test: Deux produits scalaires par rapport a un coté v0v1

Une fois ces tests concluants, on récupère la distance voulue avec le calcul suivant :

*Distance = t * D(x,y,z)*

Ceci nous donne une distance selon les 3 axes, grâce à une comparaison du vecteur directeur et des vecteurs X, Y, et Z, on récupère uniquement la distance voulue :

*Distance = t * D.x* ou *t * D.y* ou *t *D.z*

D- Proposition de solutions pour représenter les distances

Les solutions que nous allons proposer se divisent en deux catégories.

Propositions implémentées:

- Affichage des points d'intersection calculés:

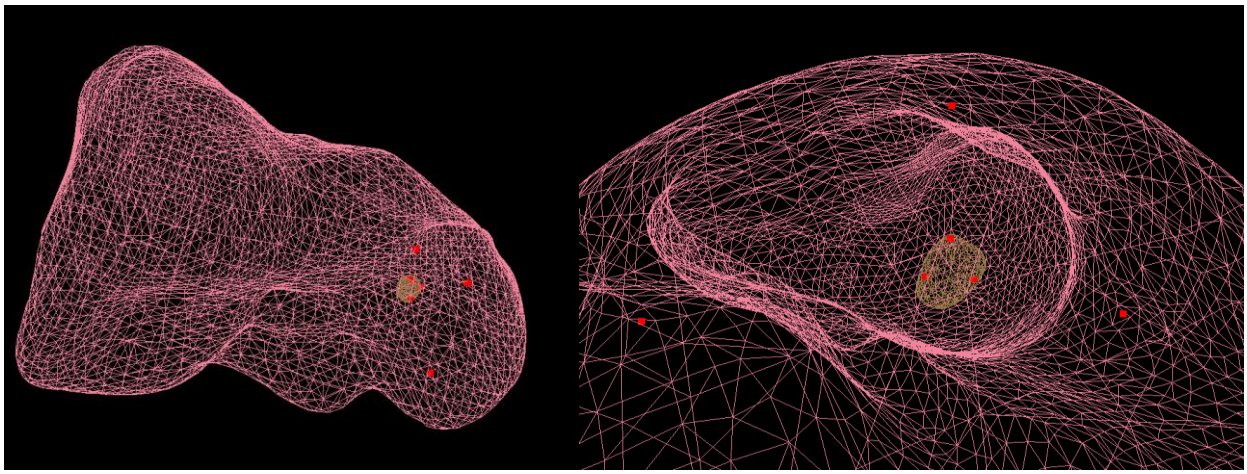


Figure 12 : Affichage des points d'intersection

- Affichage des distances avec des lignes:

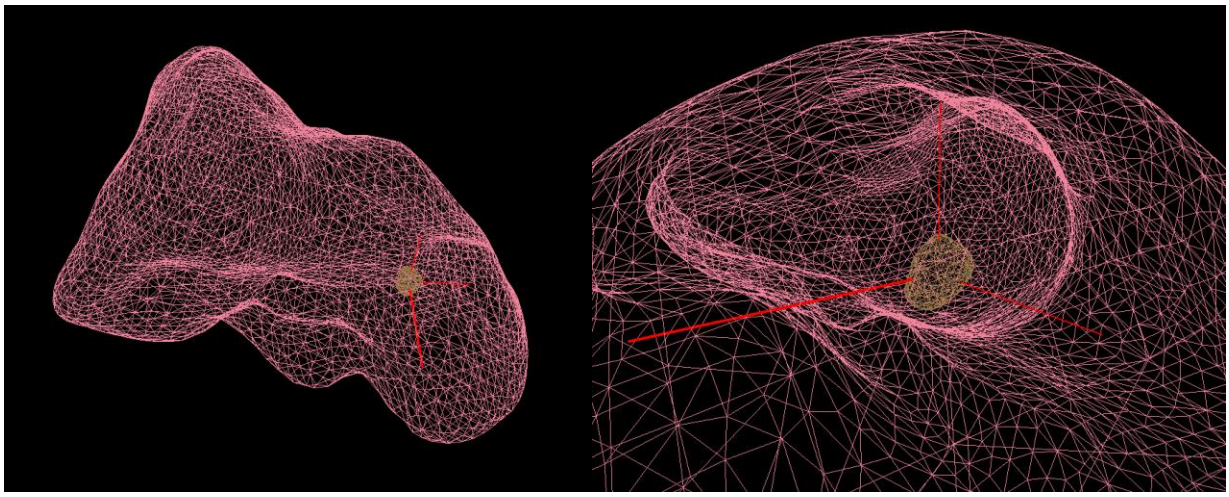


Figure 13 : Affichage des distances avec des lignes

- Affichage des distances avec des cylindres:

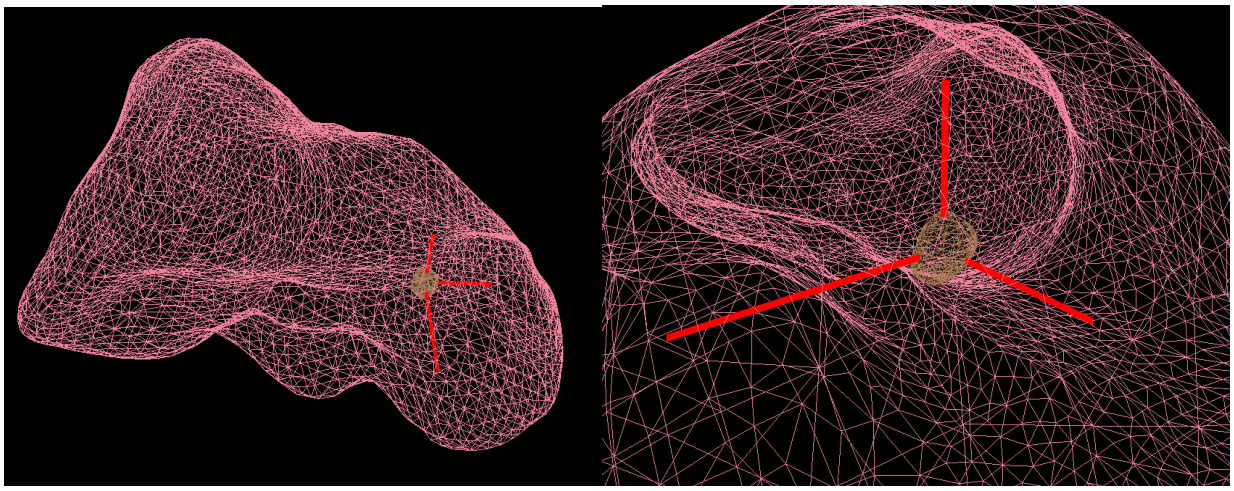


Figure 14 : Affichage des distances avec des cylindres

- Affichage des distances avec des cônes:

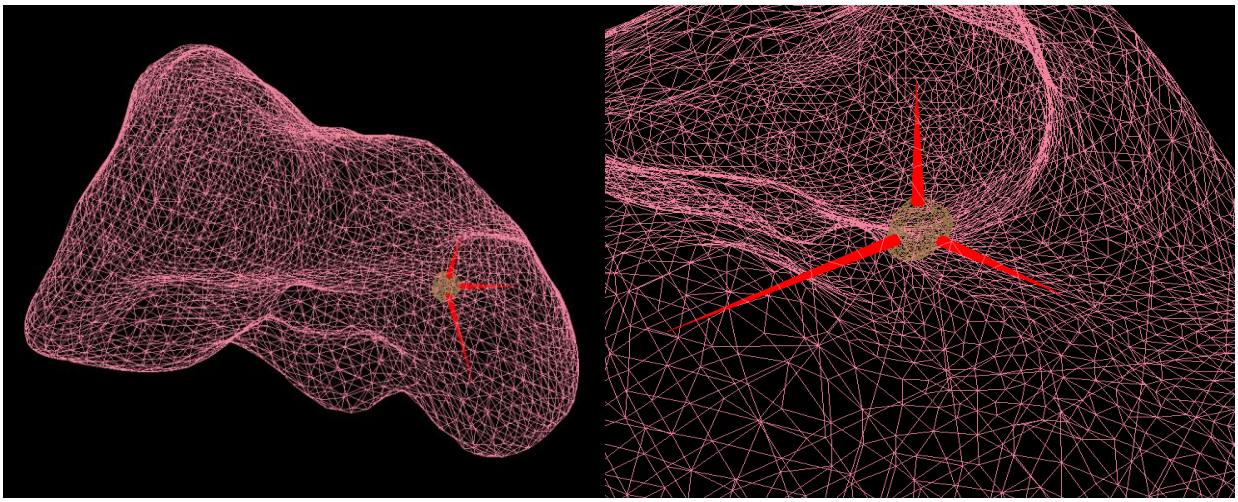


Figure 15 : Affichage des distances avec des cônes

Propositions non implémentées:

- Réaliser un affichage de la distance avec des chiffres, dans un coin de la fenêtre du rendu.
- Réaliser un affichage avec une forme (parmi celles précédemment présentées, ou d'autres) qui adapte ses dimensions en fonction de la distance.
- Réaliser ces affichages en simulant des mouvements de la tumeur par entrée d'inputs de l'utilisateur.

IV. Bilan

A- Bilan par rapport aux objectifs de départ

Les objectifs de départ étaient de proposer tout d'abord une application permettant de représenter des structures en 3 dimensions à partir d'un ensemble de coordonnées (vertices). Ensuite, il fallait trouver une méthode de calcul de la distance séparant la surface du foie et celle de la tumeur. Pour finir, il fallait proposer différentes méthodes pour l'affichage de ces distances suivant donc les trois axes : x, y et z.

Nous avons rempli ces objectifs, cependant, nous aurions aimé pouvoir implémenter plus de solutions pour la représentation des distances entre la tumeur et le foie. De même, nous aurions voulu simuler différentes positions de la tumeur en faisant des translations afin d'éprouver la fonction de calcul de distance.

B- Bilan personnel

Le projet a été très enrichissant du point de vue technique car nous avons réussi à maitriser les bases d'OpenGL et voir quelques notions plus avancées. Le choix du language C++ pour le développement de l'application nous aura aussi permis de nous remémorer et d'appliquer les cours de programmation objet que nous avons pu suivre durant notre cursus à Polytech.

De même, la gestion de projet ainsi que le travail d'équipe sont des compétences que nous avons pu développer durant le déroulement de ce PFE, et qui nous seront utiles dans nos futures vies professionnelles. Nous avons eu le plaisir de travailler avec des membres de l'équipe de recherche Shacra de l'Inria, et donc de découvrir le sujet et les problématiques de leurs travaux plus en profondeur.

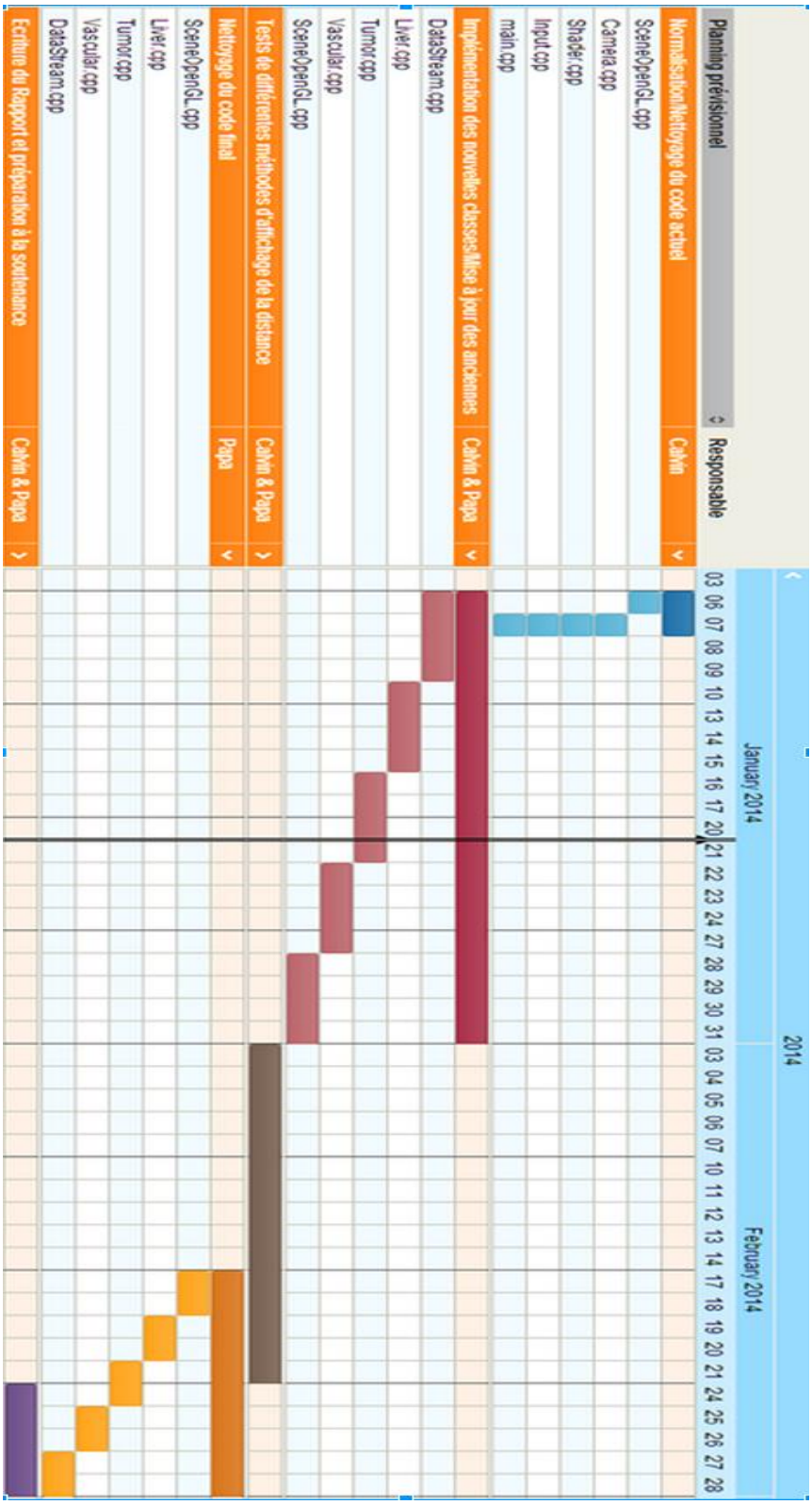
Conclusion

Ce projet nous a permis d'apporter notre contribution dans un domaine concret et important qu'est la médecine. De même, il nous a permis d'appréhender les techniques de modélisation utilisées de nos jours avec OpenGL.

Du point de vue réalisation, notre application affiche la tumeur et le foie en 3 dimensions et permet aussi de visualiser les distances, suivant les trois axes (x, y et z), entre la tumeur et la surface du foie. Différentes solutions de représentation de ces distances ont été proposées et laissées à l'appréciation des chirurgiens, à qui appartiendra la décision finale sur l'utilité et la lisibilité de chacune d'entre-elles, ainsi décideront-ils d'en garder une ou plusieurs, d'en améliorer une autre, ou peut-être que nos propositions ne leur conviendront pas, mais elles auront au moins permis de donner des pistes à l'équipe de recherche Shacra pour la suite de ses travaux.

Annexes

Annexe 1: Planning prévisionnel au second semestre



Annexe 2: Lien des tutoriaux pour l'apprentissage d'OpenGL:

http://fr.openclassrooms.com/informatique/cours/developpez-vos-applications-3d-avec-opengl-3-3

Annexe 2: Lien de la théorie mathématique sur le tracé rayon/triangle

http://www.scratchapixel.com/lessons/3d-basic-lessons/lesson-9-ray-triangle-intersection/