Fear of the Map - Tribute
Rapport de Soutenance 2

Antoine *Dak* Parenton
Raphael *Sharr* Chardon
Nicolas *Dace* Brohee
Franck *kushou* Michea

Table des matières

Introduction

Ce document va vous présenter l'aboutissement de notre travail pour la deuxième et dernière soutenance.

Les protagonistes de ce groupe vont exposer a tour rôle un récapitulatif du travail effectué pour la première soutenance et ce qu'ils se sont évertué a faire pour la dernière, mais non la moindre, soutenance. Vous y découvrirez aussi une description poignante de nos sentiments tout autour du projet : nos joies et nos peines.

Le groupe

Lors de notre premier semestre nous avons eu a realiser, comme projet en informatique, un cartographe. Le principe est simple, Il faut generer a partir d'une carte en 2D son modele 3D correspondant.

Le tout doit etre accessible a l'utilisateur via une interface graphique. Cette interface permet a l'utilisateur de charger une carte 2D, d'effectuer differentes actions tel afficher la carte en niveaux de gris ou encore de rentrer l'altitude correspond au differentes couleurs de la carte.

Nous avons baptisé ce projet *'Fear of the Map'* car Iron Maiden fait également parti de la playlist de rush. Mais aussi en référence à une private joke associé à un des membres de notre du groupe.

Depuis la nuit fatidique où Dace a fourché sur le refrain mythique de *'Fear of the Dark'* pour formuler *'Fear of the Dak'*, pseudo de l'un de nos membres. Ce dernier en a fait sa signature à l'instar de la marque jaune de Blake et Mortimer.

Chapitre 1

Le Projet

1.1 La soutenance 1

1.1.1 Organisation interne

Pour la repartition des taches, nous avons suivie le decoupage du projet propose par le cahier des specifications qui nous a ete distribue au debut de l'annee.

Nous nous sommes donc repartis selon le plan suivant :

- Franck *'kushou'* Michea s'occupe du premier prétraitement de l'image.
- Nicolas *'Dace'* Brohée s'occupe, quant à lui, de la Triangulation lors du deuxième prétraitement de l'image.
- Raphaël *'Sharr'* Chardon est chargé, lui, du maillage de la map.
- Antoine *'Dak'* Parenton travaille sur la partie liée à l'interface graphique en GTK.

1.1.2 Objectifs

Pour cette première soutenance, nous nous étions fixés pour objectifs, en accord avec le cahier des charges :

- Disposer d'une ébauche de l'interface graphique
- Un premier prétraitement de l'image fini
- Un deuxième prétraitement utilisable en attendant de terminer la Triangulation de Delaunay
- Maillage de la map

1.2 La soutenance finale

1.2.1 Organisation Interne

Pour cette seconde partie du projet nous nous sommes repartis les taches restantes tel que le moteur 3D.

Ce qui nous donne :

- Franck *'kushou'* Michea s'occupe du moteur 3D du cartographe.
- Nicolas *'Dace'* Brohée s'occupe, quant à lui, de la Triangulation lors du deuxième prétraitement de l'image.
- Raphaël *'Sharr'* Chardon est chargé, de generer le .obj.
- Antoine *'Dak'* Parenton travaille sur la partie liée à l'interface graphique.

1.2.2 Objectifs

Au vue de la deuxieme soutenance nous avons prevue d'obtenir ce nous manquait, comme par exemple, une interface graphique complete

- Disposer de l'interface graphique complete
- Un moteur 3D
- Un deuxième prétraitement via la Triangulation de Delaunay
- Un fichier obj

Chapitre 2

Notre travail

2.1 Traitement de l'image

La mise en niveau de gris

Notre programme pocède deux niveaux de gris. Le premier est un niveau de gris standart, suit les recommandations de la C.I.E. [1]quant à la caractérisation de la luminance d'une couleur, recommandation 601. C'est formule corespond à la correction de la valeur gamma de l'image.

Soit $(r, g, b) \in [0; 255]^3$ et $g \in [0; 255]$, alors

$$g = 0.299 * r + 0.587 * g + 0.114 * b$$

Où g est la couleur grise corespondante.

La seconde méthode utilisé ne tient plus compte des couleurs, mais de leur nombre pour produire un dégradé proportionnel de gris. Cette méthode de mise en niveau de gris n'est actuellement pas utilisé dans le projet mais pourrait venir à être utilisé.

Détection des contours

Pour la détection des contours, j'ai décidé d'utiliser le filter de Canny [2]. L'auteur l'a conçu pour être optimal suivant trois critères clairement explicités :

- bonne détection : faible taux d'erreur dans la signalisation des contours
- bonne localisation : minimisation des distances entre les contours détectés et les contours réels
- clarté de la réponse : une seule réponse par contour et pas de faux positifs

L'algorithme intègre une réduction du bruit de l'image à la base, modifiant sa valeur suivant les valeurs de ses voisins. Pour ceci, on utilise une matrice de convolution [3], qui corespond à la somme de tous les voisins coéfiscienté grâce à la matrice si dessous, puis multiplié par un coefiscient inférieur à 1 (de l'ordre de $\frac{1}{130}$ minimum).

$$B = \frac{1}{coef} * \begin{pmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{pmatrix} * A$$

1. C.I.E. : Commission Internationale de l'Éclairage
2. http ://en.wikipedia.org/wiki/Canny_edge_detector
3. http ://manual.gimp.org/en/plug-in-convmatrix.html

Comme nous travaillons avec des cartes parfaites pour la première soutenance, cette option n'est actuellement pas utilisée, d'autant plus qu'elle rajoutait un certain nombre de surface lors de la detection des couleurs.

J'ai déjà en tête quelques idées pour règle ces erreurs dans le résultat, en cas d'utilisation de la réduction du bruit, nous n'aurons donc pas de problème de ce côté là dans le futur.

Au niveau de la détection des contours, on reste dans l'idée d'appliquer un filtre sur l'image, mais ici avec deux matrices. On procède en fait au calcul du gradient du point. Les deux matrices sont :

$$\begin{pmatrix} -1 & 0 & 1 \end{pmatrix} et \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

Dans le même esprit, on calcul les sommes, puis on calcul une valeur g égale à la somme des valeurs absolues des calculs sur les pixels voisin. On peut donc en déduir la presence d'un contour grace à ça.

Au niveau de la gestion des bords de l'image, j'ai décidé d'utiliser la technique du mirroir, qui consiste a prendre la couleur des pixels de l'autre côté du filtre.

2.1.1 Demande des hauteurs

Pour la demande des hauteurs, une l'idée la plus intéressante que nous avons eu est de mettre en valeur la zone pour laquelle nous demandons la hauteur. Ainsi, l'utilisateur a une idication visuelle de ce qu'il est en train de remplire. Pour ceci nous avions deux technique. La technique du remplissage par flood, qui corespond à la coloration d'une partie d'un graphe, pour faire un parallèle avec le cours actuel. Cette technique consiste à remplir la zone jusqu'à sa bordure.

Nous avons préféré une autre technique qui s'adaptait bien mieux avec notre algorithme de detection des couleur, qui est de récuperer la liste des pixel dans la zone et de les colorer sur une image en niveaux de gris avec les contours.

Je pense que cette technique est l'une des plus pratiques sur tous les projets.

2.2 Triangulation

La modélisation d'un objet 3D se fait à partir de la modélisation de cet objet en faces de polygones constitué de trois points aussi appelés triangles. La subtilité est de trouver un algorithme permettant de modéliser ces triangles à partir d'un semi de points.
La triangulation permet donc de déterminer ces faces à partir d'une liste de points. L'une des solutions a défaut d'être universelle est de tenir compte du fait que les points dans le cas qui nous intéresse sont les intersections du cadrillage. Soit une modélisation simple mais efficace, en partant du principe que chaque "carré" formé par le cadrillage contient deux triangles rectangles-isocèles. L'autre solution qui permet de traiter tout semi de points qu'il soit avec ou sans contrainte est la triangulation de Delaunay. C'est pour quoi nous nous sommes intéressés à ce dernier...

2.2.1 Delaunay

Principe de la triangulation incrémentale de Delaunay (TID)

Un espace ,dit de Delaunay, est un espace dont le cercle circonscrit a chaque triangle ne contient que les sommets de ce triangle. S'il y a conflit entre deux triangles, on bascule l'arrête commune aux deux triangles de façon a obtenir dont l'arrête commune est le complément aux sommets communs des deux premiers triangles. Explication en image :

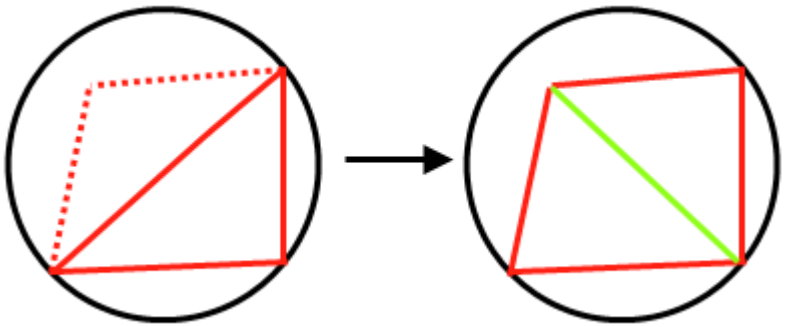

FIGURE 2.1 – Basculement (Flip) de deux triangles

Principe

Si on veut insérer un point dans un maillage, en partant du principe que ce maillage est déja triangulé (il respecte le critère de Delaunay), il faut donc :

1. Trouver le triangle contenant le point

2. Diviser ce triangle en trois nouveaux triangles à inserer dans le maillage et retirer l'ancien triangle du maillage

3. Vérifier que les triangles voisins aux nouveaux triangles respectent le critère de Delaunay par rapport au nouveau point inséré :

   – Si ce n'est pas le cas, le flip est appliqué aux deux triangles adjacents (le nouveau triangle et son voisin) et deux autres voisins sont testés récursivement

   – Sinon, rien n'est fait et le nouveau triangle est conservé

Ce crtière nous laisse une souplesse majeure dans l'implémentation : déterminer la fonction de recherche.

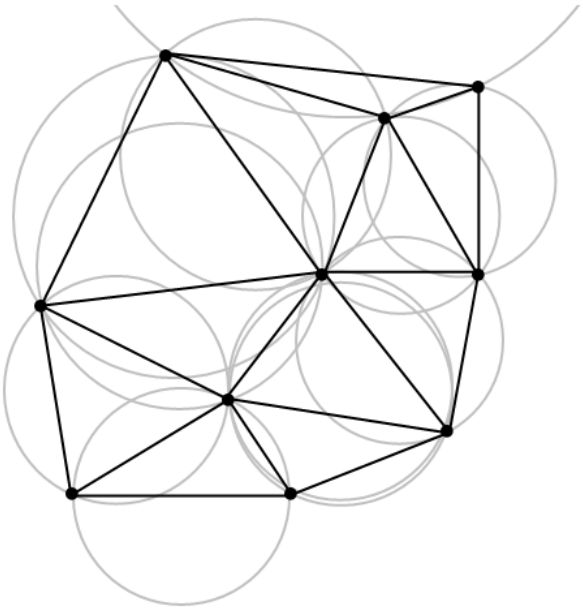

FIGURE 2.2 – Un exemple d'espace de Delaunay

Les structures de données utilisées

Etant donné qu'on s'intéressait de plus en plus à l'aspect "orienté objet" du langage CAML. j'ai décidé de changer les records que j'avais utilisé pour représenter les types lors de la première soutenance en objet. Je vous ferais grâce de l'énnumération des getters et setters lors ce descriptifs qui découlent de l'utilisation de ces structures.

L'objet `Point` possède assez simplement 3 attributs qui sont les 3 valeurs entières de ses coordonnées, sachant que l'algo n'exploite que l'abscisse et l'ordonnée des points.

L'objet `Straight` qui contient le coefficient directeur ,`gradient`, et l'ordonnée à l'origine ,`intercept`, qui sont tout les deux des valeurs flotantes est un outil qui me permet d'implémenter les droites afin de répondre à l'implémentation des fonctions de géométrie nécessaires. A noter que dans le cas d'une droite perpendiculaire à l'axe des abscisses (d'équation x=foo) `gradient` représente la valeur de x et `intercept` est un NaN.

L'objet principal de l'algo ,`Triangle`, possède 5 attributs vers des `Points` respectivement les 3 sommets du triangle, le centre de son cercle circonscrit et un point dont on est sur qu'il est dans le triangle (ici la somme des coordonnées des sommets divisée par 3) , 3 attributs vers des `Triangle Option` qui sont les voisins du triangle, du type `Option` car ils sont égaux a None si le triangle ne possède pas de voisin. Ainsi que le rayon du cercle circonscrit au triangle pour vérifier le critère de Delaunay et . Cet objet possède deux relations importantes : le triangle `tvi` est l'opposé de `si` et l'incrémentation de ces `i` se fait dans le sens horaire :

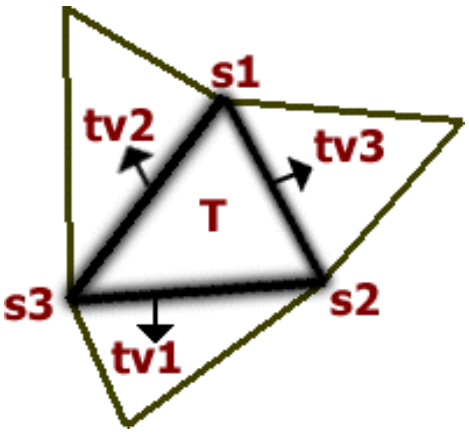

FIGURE 2.3 – structure Triangle

La recherche

Fort de mes connaissances en géométrie acquises au collège, je me suis lancé dans une première élaboration de l'algoritme me permettant de trouver le triangle contenant le point venant d'être inséré.

J'ai donc pensé a une fonction de recherche qui vérifie que le points n'appartient pas au triangle courrant et sinon parcourrerait récursivement le graphe créé par les triangles et leurs voisins en rappelant la fonction sur le triangle dont le point correspondant aux coordonées divisées par 3(appelons-le O) est le plus proche du points inséré.

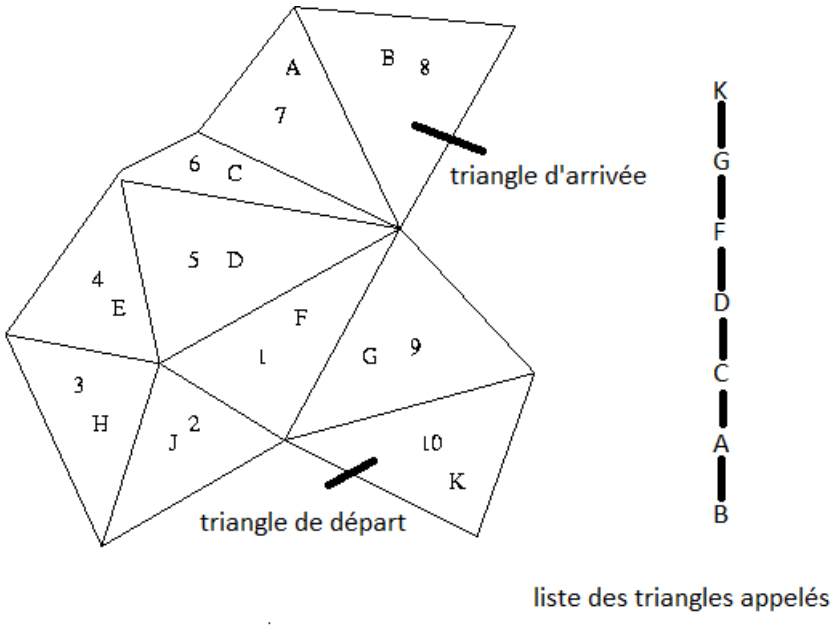

FIGURE 2.4 – Recherche Idyllique

Je ne prenais cependant pas en compte le cas suivant, où la distance entre O du triangle de départ et le points d'insertion est minimale mais le triangle ne contient pas pour autant le point :

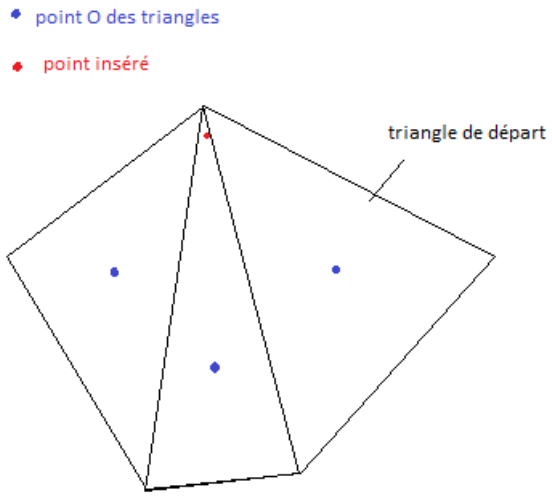

FIGURE 2.5 – Cas bien réel

Je me suis donc tourné vers une autre méthode. Cette dernière consiste a vérifier toujours si le point n'appartient pas au triangle courrant et sinon rappeler sur le triangle commun à l'arrête coupée par la droite entre le point O et le point inséré avec O et le points de part et d'autre de l'arrête.

Explication en image :

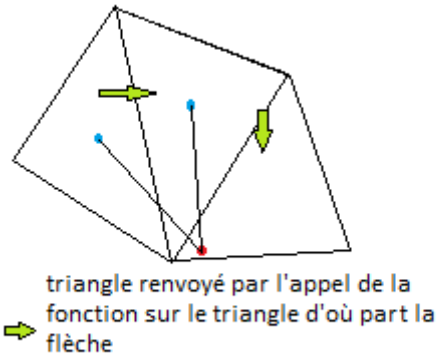

FIGURE 2.6 – Recherche adéquate

Complexité de l'algorithme

Dans la plupart des autres groupes la complexité de leur implémentation de Delaunay atteint du $O(n^3)$. Cependant ils stockent leurs triangles dans une liste de triangles et doivent donc l'enlever lors d'un flip ou d'une insertion dans ce triangle. Ils doivent parcourir la liste et vérifier si chaque triangle n'est pas celui contenant le point. Mon idée est d'éviter cela, en gagnant du temps sur la recherche du triangle par le parcours du graphe. Ce qui donne une bonne réduction de la complexité sur un semi, dont les points sont insérés aléatoirement, mais qui donne, ici, une réduction encore meilleure. Car le dernier point inséré (et donc le dernier triangle manipulé) est proche du prochain sauf en cas de passage d'une ligne du cadrillage à une autre. Ce qui se fait (hauteur de l'image/ le pas) fois, soit peu souvent comparé au nombre de points sur la majeure partie des images. De plus nul besoin de supprimer le triangle, le Garbage Collector s'en occupe pour vous !

Choix final

Malgré, le travail effectué sur cet algoritme nous avons tout de même décidé de prendre la triangulation de base, qui s'effectue en tenant compte du cadrillage. Car cette dernière a un meilleur rendu que Delaunay qui en plus de divers soucis de pointeurs à None a le défaut d'être également en $O(n^2)$

2.3 Moteur 3D

Pour le moteur graphique, j'ai utilisé OpenGL, avec les librairies freeglut et glMLite. Malgrès une documentation assez pauvre, j'ai réussi à produire un moteur graphique correct, en m'inspirant de codes trouvés sur le net.

Le moteur graphique permet d'afficher les cartes de deux façons différentes. La première, en faces pleines et la seconde en fil de fer. Disponibles en temps réel avec les touches *w* et *f*. Au demarrage, on se trouve au dessus de la carte. Cette vue permet de faire une annalogie avec la carte de base. Vous pouvez déplacer la vue avec la souris, en faisant des cliquer-deplacer.

Si des faces sont invisibles, vous pouvez essayer de regler le problème avec la touche *h*, mais ça ne marche pas totallement à tous les coups. Ceci vient d'un comportement bizarre d'OpenGL.

Hormis ce léger bug, le reste marche bien et répond convenablement au cahier des charges.

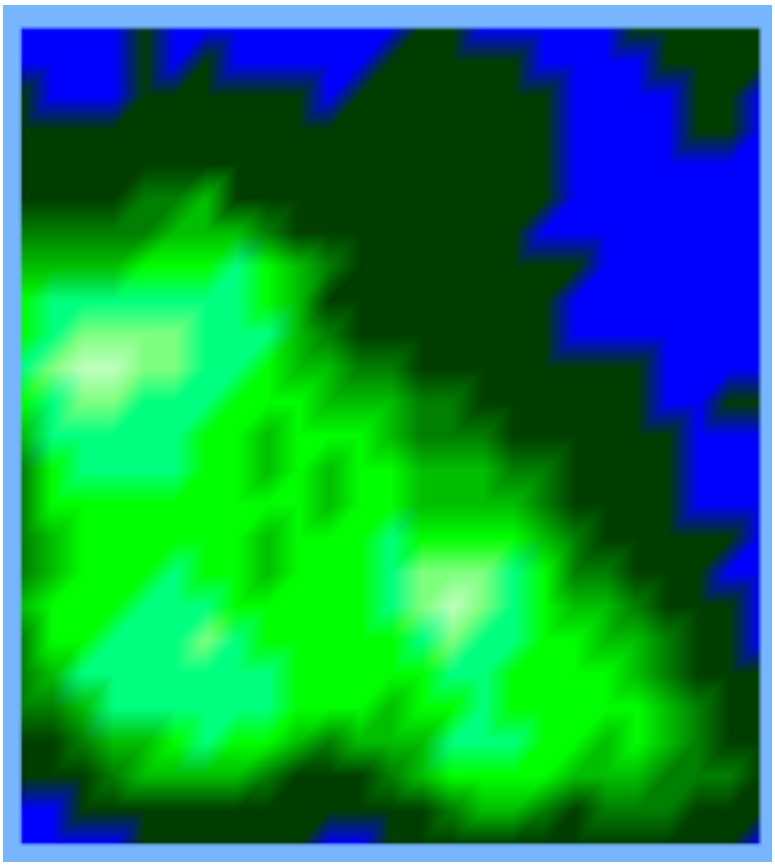

FIGURE 2.7 – 3D - Ouverture du programme : Au dessus de la carte.

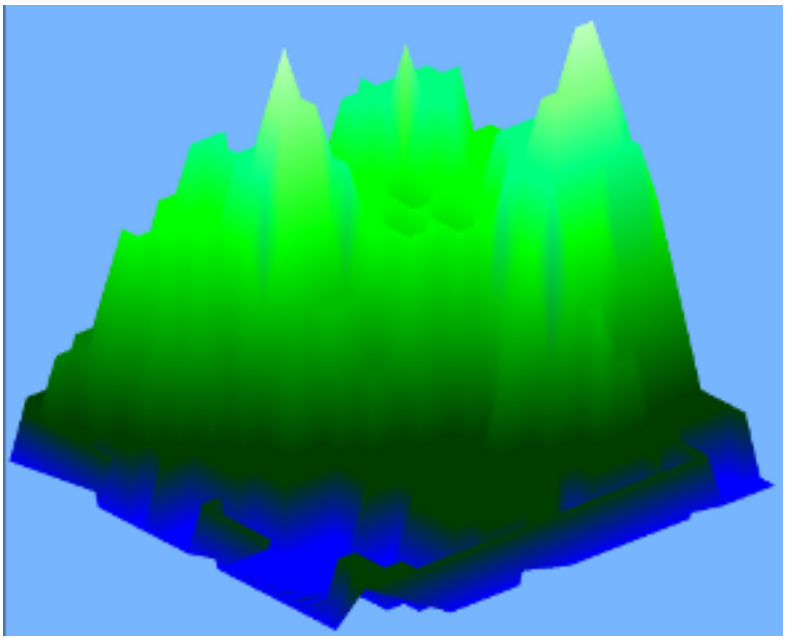

FIGURE 2.8 – 3D - Vu classique.

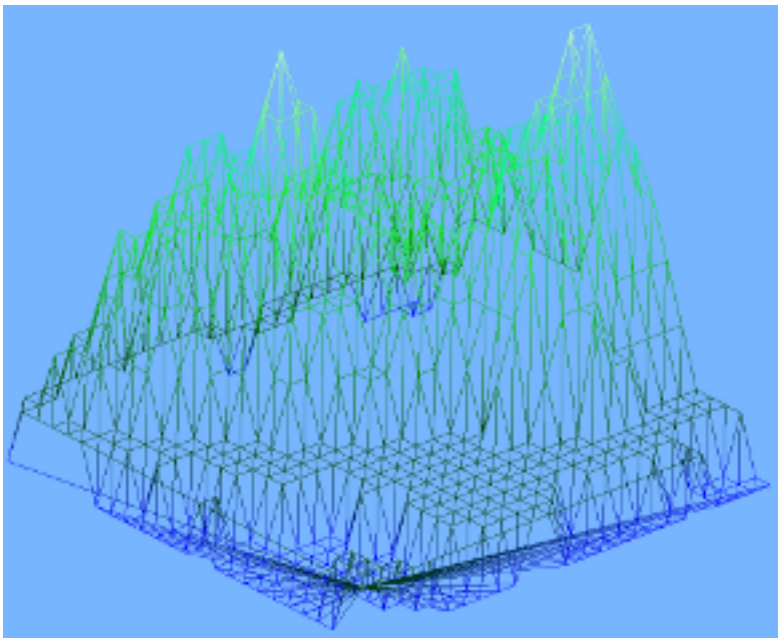

FIGURE 2.9 – 3D - Vu en fil de fer.

2.4 Generation et parsage du fichier objet

2.4.1 Un fichier obj c'est quoi ?

Lorsque que j'ai hérité de cette partie du projet j'ignore ce qu'était un fichier obj mais un rapide tour sur Wikipedia combla cette lacune.

> OBJ est un format de fichier contenant la description d'une géométrie 3D. Il a été défini par la société Wavefront Technologies dans le cadre du développement de son logiciel d'animation Advanced Visualizer. Ce format de fichier est ouvert et a été adopté par d'autres logiciels 3D (tels que Poser de e-frontier, Maya de Autodesk, Blender etc) pour des traitements d'import / export de données.
>
> Les formes géométriques peuvent être définies par des polygones ou des surfaces lisses telles que des surfaces rationnelles et non rationnelles. [4]

Le format obj est définie de manière suivante :

Une liste de points qui sont noté de manière suivante, v x y z. Par exemple :

v 1.0 2.0 3.0
v 4.0.5.0 6.0
v 7.0 8.0 9.0

Ensuite on définit les normales à chaque faces. une normale est un vecteur qui est orthogonal par rapport à un plan,le plan en question est définit par deux vecteur. Les normales nous servirons par la suite pour effectuer la réflection de la lumière sur les faces de nos objets Elle est calculée par la formule suivante : Soient deux vecteurs définit par u(u1,u2,u3) et v (v1,v2,v3), la normale est le produit vectoriel de c'est deux vecteurs.

Ce qui nous donne ici :

vn 7.0 6.0 -3.0

Et enfin on définit les objet que nous voulons créer via des triangles :

f 1/1 2/1 3/1

Il est définit par les sommets 1, 2 et 3 et par la normale 1

4. http ://fr.wikipedia.org/wiki/Objet_3D_(format_de_fichier)

Le rendu final aura cette allure la : Lorsque que j'ai hérité de cette partie du projet j'ignore ce qu'était un fichier obj mais un rapide tour sur Wikipedia combla cette lacune.

v 1.0 2.0 3.0
v 4.0.5.0 6.0
v 7.0 8.0 9.0

vn 7.0 6.0 -3.0

f 1/1 2/1 3/1

2.4.2 Le fichier dans le projet

Lorsque j'ai implémenté la première version de mon algorithme, la triangulation n'étais pas encore prête. J'ai donc été oblige de parcourir le fichier en créant des triangles via un pas arbitraire.

J'ai créé un type *triangles* et un type *point* et tout en parcourant la carte je créais des triangles lies par leurs sommets communs puis parcours le graphe ainsi obtenu, j'avais donc les points et les triangles correspondants sans avoir pour autant de doublons.

Malheureusement en utilisant cette méthode je n'avais pas accès au point étant sur les contours délimitants les changement d'altitude. Et la visualisation était assez moche a voir.

Pour la deuxième version j'ai change de méthode car j'avais accès a une liste de triangle que me renvoyait la triangulation. J'ai donc décidé de parcourir cette liste en écrivant les triangles via les points dans l'ordre de rencontre. Le problème c'est que cet algorithme n'était pas très optimise dans le sens ou j'écrivais les sommets de tous les triangles. Les triangles ayant des sommets communs, je me retrouvais avec un bon nombre de doublons dans mon fichier.

Dans la derniere version j'utilisais encore cette liste de triangle mais du manière plus effisciente. Lors de la génération du *.obj, je retiens à la volée les lignes de mes points, et les retrouve en O(1) grace à une table de hashage lorsque j'en ai besoin pour les triangles.

2.4.3 Parser un *.obj

Les fichiers *.obj ont une syntaxe compliqué, car très libre. De ce fait coder un parseur de *.obj qui pourrait prendre n'importe quel *.obj en entré s'avère très compliqué.

Pour ce projet, il a été décidé de ne parser que les *.obj que nous générons, ce qui nous permet d'avoir un parseur très simple à base de'utilisation du module *Scanf* d'ocaml.

Bien sûr, ce détails est également précisé dans le manuel d'utilisation fournit avec ce rapport.

2.5 Interface Graphique

L'interface graphique a pour but de guider l'utilisateur, lors des différentes étapes vues précédemment jusqu'au rendu 3D. Elle se veut intuitive, ergonomique et agréable pour l'utilisateur.

L'objectif principal de cette soutenance était de réunir toutes les parties réalisées par mes collègues dans mon interface, et de faire en sorte que tout aille bien !

Chose qui ne fût pas facile, puisque nous avons dû beaucoup nous concerter pour réunir les parties une à une, sans perdre trop de temps à chaque fois.

2.5.1 Reprise de la première soutenance

Lors de la première soutenance, j'avais proposé une interface graphique minimale, développée avec lablGtk à la dure. Elle était composée d'une fênetre et de plus boutons me permettant un premier affichage de l'image choisie par l'utilisateur.

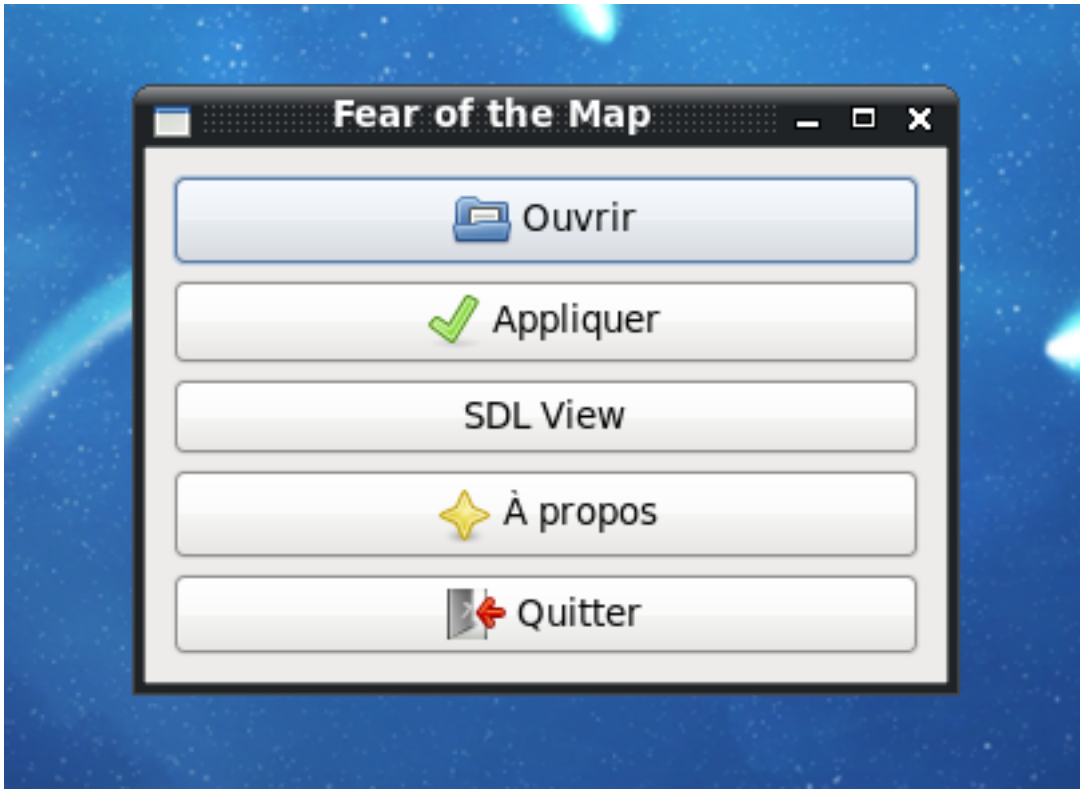

FIGURE 2.10 – Interface graphique - Première Soutenance

Malheureusement l'affichage de l'image via le module GMisc était statique et ne me permettait pas de changer l'image à afficher. C'est pourquoi j'avais opté pour une solution de secours, utilisant la librairie graphique SDL. Mais il était clair que pour la suite du projet cette méthode m'empêcherait de réaliser l'interface graphique digne du groupe *Tribute*.

C'est pourquoi, lors de la première soutenance avec Krisboul, j'avais évoqué la possibilité de me tourner vers Glade ou Gazpacho afin de réaliser l'interface plus simplement pour éviter de me retrouver bloqué à cause du peu de documentation concernant lablGtk.
En effet, lors des précédents rushs j'avais passé beaucoup de temps à lire le peu de documentation, et à essayer de trouver les outils dont j'avais besoin. Du temps qui me semblait perdu puisque le rendu était minimale.

2.5.2 Décision prise

Tout de suite après la première soutenance, je me suis donc penché sur Glade (plus documenté que Gazpacho) et j'ai donc découvert à quel point Glade était simple d'utilisation, concernant la création visuelle de l'interface.

J'ai donc créé l'interface que je voulais obtenir, avec Glade, ce qui m'a permis de trouver et comprendre les outils qui me seraient nécessaires pour la réaliser en codant directement avec lablGtk.

En effet, ma décision était prise ! Le temps passé à lire la documentation, à tester, et à chercher n'était pas du temps de perdu, mais bien au contraire un gain de temps énorme pour la suite du projet, puisqu'avec les bases acquises et l'aperçu de ce que je souhaitais obtenir, la suite fût aisée.

2.5.3 Interface visuelle réalisée

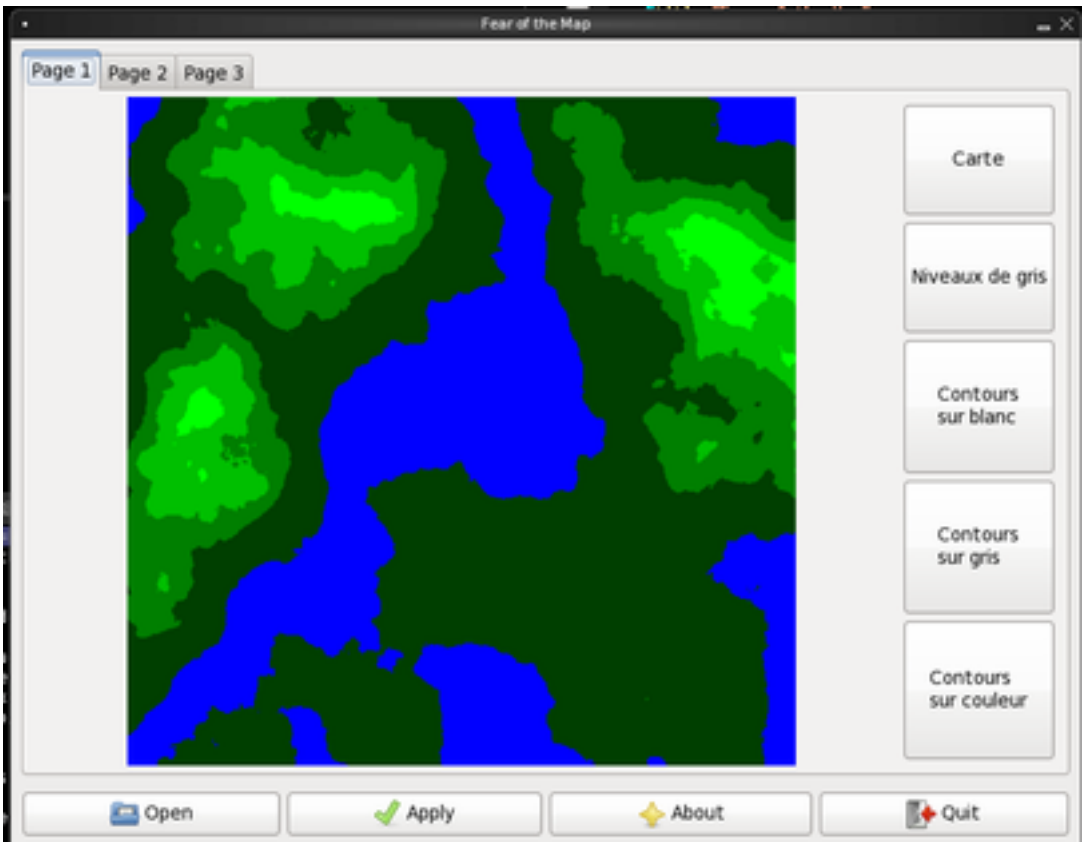

FIGURE 2.11 – Interface graphique

Lors de la réalisation de l'interface visuelle, j'ai divisé l'écran en 3 parties : emacs, l'affichage de ce que j'avais compilé, et l'affichage que je voulais obtenir.
J'ai donc tout repris à zéro, et j'ai avancé pas à pas à un rythme très soutenu pour enfin arriver au résultat tant attendu.
L'interface se divise donc en trois onglets une barre de boutons située en bas.
Tout d'abord, l'utilisateur arrive sur le premier onglet, où il est invité à charger une image via le bouton correspondant. A partir de là, il lui suffit de valider son choix en cliquant sur le bouton 'Appliquer'. Sur la gauche de l'onglet, l'utilisateur peut voir divers boutons qui serviront à l'affichage de la map sous divers formats (image en gris, avec contours, etc ...).
A partir du moment où l'image est chargée, l'utilisateur peut alors passer sur le deuxième onglet qui lui permet d'enregistrer les hauteurs correspondantes aux différentes zones, puis d'enregistrer son choix pour enfin arriver sur le troisième onglet et sur l'affichage 3D.

De plus, l'utilisateur pourra trouver des informations concernant le projet Fear of the Map, ainsi que sur nous, membres du groupe *Tribute* : par exemple, nos emails, notre site internet, ou encore la license du projet.

2.5.4 Implémentation des autres parties

Le travail le plus long fût d'intégrer les parties de mes collègues dans l'interface, et ce ne fût pas facile, car nous avons rencontré plusieurs problèmes plus ou moins durs à gérer.

Tout d'abord afin de nous démarquer des autres projets de cartographie, nous avions pensé à divers affichages de la carte choisie par l'utilisateur comme par exemple l'affichage en gris, ou l'affichage des contours avec ou sans couleurs. Pour cela, nous avons décidé d'intégrer les fonctions de création des fichiers *.grey.bmp, .edges.bmp, .iwedges.bmp, etc ..* au bouton d'appliquer afin de les afficher dans le premier onglet.

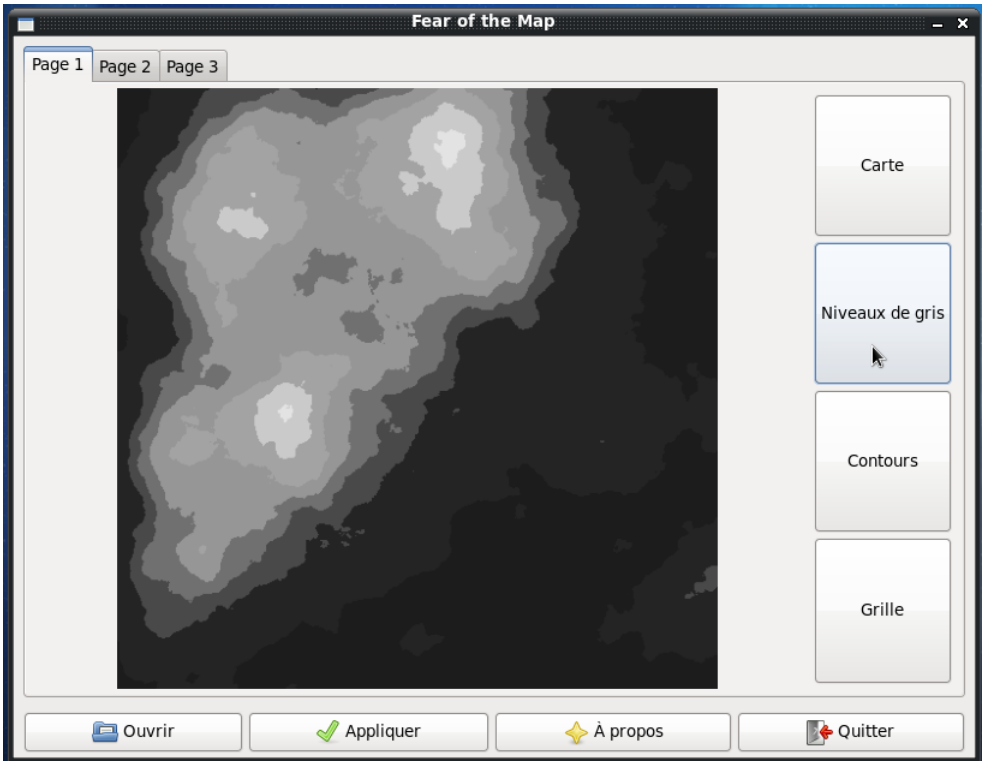

FIGURE 2.12 – Interface graphique

Notre deuxième idée était d'implémenter un système de choix de hauteurs par rapport à la zone colorée, tel que nous l'avions réalisé pour la première soutenance en console. Nous avons donc créé les images correspondantes aux zones pour les afficher une à une dans le deuxième onglet, permettant ainsi à l'utilisateur de bien voir sur quelle zone il souhaite appliquer la hauteur.

Notre idée a été jalousement copié par d'autres groupes, mais nous sommes fiers d'avoir sû nous démarquer de la plupart des autres groupes.

Une fois toutes les hauteurs enregistrées ainsi que le pas, l'utilisateur peut enregistrer, et appliquer ses choix.

Cela va alors lancé la procédure de création du .obj correspondant à la carte choisie par l'utilisateur, pour enfin l'amener à l'affichage 3D.

2.6 Site web

L'adresse du site est : http ://www.fearofthedak.com/fotm

Le site a un design simple dont les couleurs dominantes sont le rouge et le noir. J'ai réalisé un site statique en HTML agreemente de CSS.

Le site se décompose en 4 parties :

- L'accueil. Dans cette section on y retrouver les *news* sur l'avancement du projet. Ainsi que deux versions de *Fear Of The Dark* une par *Iron Maiden* et une autre par *Van Canto*, groupe de métal à capela : ils n'ont pour tout instruments qu'un batterie, le reste étant fait a la bouche.
- La présentation du groupe, ou on peut trouver de magnifiques photos de tout les membres du groupe. Ainsi qu'un résumé succin de leur vie.
- La section telechargement, ici on y trouve les sources de notre projet ainsi que nos rapports de soutenances.
- Et enfin la section contact ou vous pouvez prendre contact avec nous en cas de bug ou tout simplement pour féliciter les auteurs.

Conclusion

Au final ce projet, a priori facile, a ete ,au vue du manque de documentation, en particulier a niveau de GTK, plus difficile que prevue.

Mais malgré tout nous avons reussi a surmonte ces difficultees pour atteindre nos objectifs. Le projet est complet, et répond correctement au cahier des charges qi nous a été fournit en début d'année.

De plus malgre le peu de souvenirs du Caml vue en Sup, ce projet nous a permit de faire de gros progres dans ce language.