ÉCOLE NATIONALE
DES SCIENCES
GÉOGRAPHIQUES

Ecole Nationale des Sciences Géographiques
IT 2 – Projet de webmapping

VISUALISATION DE L'INTERVISBILITE SUR UN MNT :

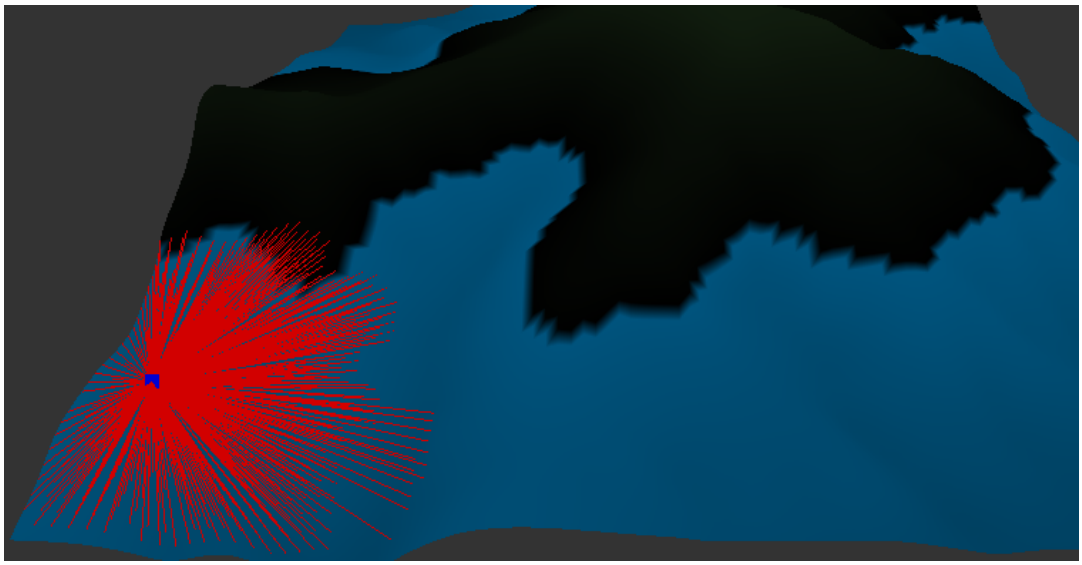


Table des matières

TABLE DES MATIERES.....2

INTRODUCTION.....3

POSITION DU PROBLEME.....4

CHOIX TECHNIQUES.....5

GESTION DE PROJET.....8

CONCLUSION.....10

Introduction

Dans le domaine de la géomatique, la visualisation de données à trois dimensions sur ordinateur est devenue incontournable et par conséquent les élèves du master Technologie des Système d'Information de l'Ecole nationale des sciences géographiques doivent apprendre à manipuler au moins une bibliothèque graphique. C'est donc dans le cadre de ce master, que s'inscrit ce projet géomatique visant à utiliser des librairies comme OpenGL, ou des frameworks tels que Qt.
Ce projet, réalisé par Philémon Pensier et Maxwell Houtonhadja, a pour but de visualiser en trois dimensions un modèle numérique de terrain, ainsi que de représenter les éventuelles intervisibilités, lorsque l'on décide de se placer à un endroit précis du MNT.

I. Position du problème

1) Le sujet

Ce projet avait pour thème: "L'intervisibilité sur un MNT (Modèle numérique de terrain)". La notion d'intervisibilité peut être définie, soit comme l'ensemble des points visibles depuis un point de vue, soit comme des points de vue, d'où est perceptible une cible précise. Un utilisateur qui décide de se positionner à un endroit précis du MNT doit donc avoir la possibilité de voir ce qui est visible à côté de lui depuis sa position. Ce projet propose ainsi une méthode originale de visualisation du champ d'intervisibilité dans le contexte d'un modèle numérique de terrain maillé (MNT raster).
Les modèles numériques de terrain (MNT) sont très largement répandus pour visualiser le relief d'une zone en trois dimensions. Ils permettent de nombreuses applications pratiques, dans des domaines aussi divers que les sciences géographiques, la réalité virtuelle ou encore la construction. Aussi, la visualisation d'un champ d'intervisibilités possède plusieurs applications dans des domaines distincts, tels que, les applications militaires ou encore les télécommunications. C'est par ailleurs avec l'intention de l'appliquer au premier domaine que l'application a été conçue. Ainsi, l'utilisateur doit pouvoir choisir une distance limite à partir de laquelle, il ne doit pas chercher les éventuelles intervisibilités. Deux cas d'utilisation ont ainsi été développés au cours de ce projet:
-Rechercher l'inter visibilité entre deux points définis par l'utilisateur à partir de leurs coordonnés planimétriques et altimétriques.
-A partir d'un point créé par l'utilisateur, grâce aux coordonnées de ce dernier, définir l'ensemble des points visibles.
Dans un souci de simplification, les interférences atmosphériques ou encore la courbure terrestre ne seront pas prises en compte. De plus, comme nous ne disposons pas d'un MNE (Modèle Numérique d'Elévation), les obstacles topographiques seront donc les seuls obstacles possibles à la visibilité.

2) Les Contraintes de réalisation

Pour réaliser ce projet, nous étions dans l'obligation de programmer en C++, mais aussi d'utiliser le framework Qt, pour développer l'IHM (Interface Homme Machine) de notre application. En outre, afin de réprésenter le MNT, nous avons également dû utiliser OpenGL. Par ailleurs, c'est dans ce contexte et afin de gagner du temps, que nous avons également choisi d'utiliser la librairie libqglviewer, qui permet une gestion facile de la caméra via la souris ou le clavier.

3) Données en entrée

La principale donnée pour ce projet, fournie par notre commanditaire, est un fichier texte au format xyz contenant un semis de point 3D correspondant à un MNT. A l'intérieur du fichier, chaque ligne correspond aux composantes X, Y et Z d'un point du MNT. Les coordonnées des points sont exprimées dans le système de projection Lambert II étendu, qui couvre toute la France métropolitaine. Les coordonnées (X, Y, Z) sont par ailleurs triées par Y décroissant puis par X croissant, avec un pas en X et en Y de 25 mètres. Le MNT couvre une zone rectangulaire de 100 kilomètres de longueur, dans le massif des Ecrins, dans les Alpes francos-italiennes; il s'agit donc d'un fichier assez volumineux (plus de sept millions de points), d'où l'intérêt pour l'utilisateur de définir une zone de recherche pour le calcul des intervisibilités.

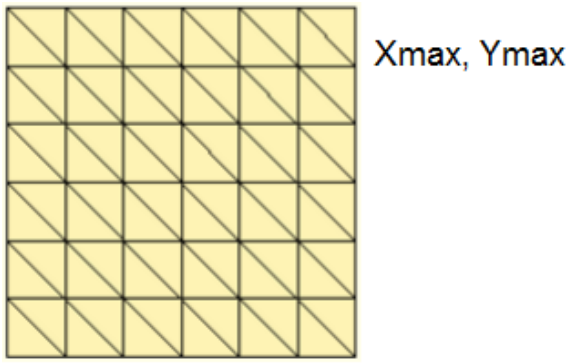
II. Choix techniques

1) Affichage du MNT

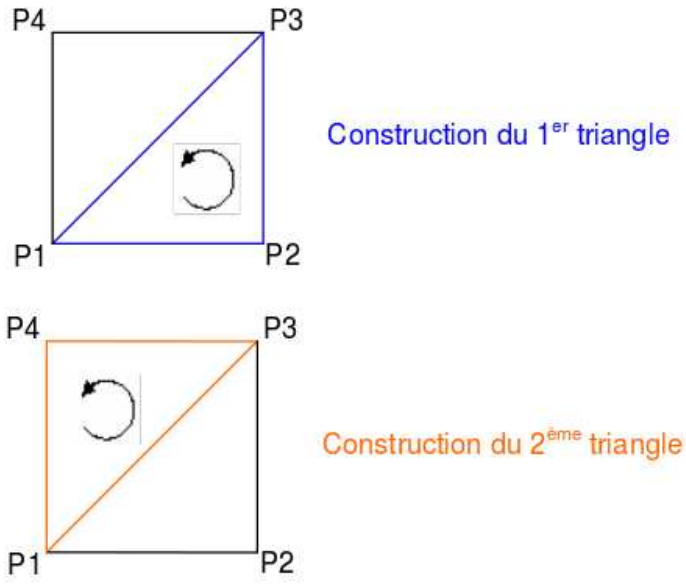
Dans un souci de clarté et de modularité, notre code a été initialement volontairement découpé en 3 grosses parties, la partie calculs (correspondant principalement aux classes MNT, droite et plan), la partie visualisation (la classe SimpleViewer) et enfin la partie IHM (la classe MainWindow).

Une fois que l'utilisateur a choisi un MNT dans la fenêtre de dialogue qui lui est proposée lors de l'affichage, est abordée l'étape de la lecture. Le fichier en entrée est alors lu séquentiellement et les points sont alors stockés dans un QVector de QVector3D, puis transmis par référence au viewer et à l'objet MNT. C'est à ce moment-là, que le pas en X et en Y du MNT, le nombre de points par lignes, ou encore les bornes du MNT(valeurs minimales et maximales des coordonnées), sont calculés, via les méthodes *computeLineLength* et *computeExtreme*. Les bornes du MNT serviront plus tard à recentrer les points en X=0 et en Y=0 et donc de permettre un choix plus simples des coordonnées pour l'utilisateur final. Le pas s'obtient quant à lui en faisant simplement une différence entre deux points successifs en abscisse et en ordonnées, ce qui impose de connaître le nombre de points par lignes. Ce dernier est calculé en repérant simplement l'arrêt de la croissance des abscisses.
Faute de temps nécessaire, nous n'avons malheureusement pas eu le temps de réaliser l'ouverture (parfois longue) du MNT en parallèle, afin de ne pas figer l'interface du programme et de tenir l'utilisateur au courant de l'avancée de l'ouverture.

Comme nous disposions d'un maillage de points régulier, nous n'avons pas eu besoin de triangulation ou d'utiliser une bibliothèque complexe pour cela. Nous avons simplement choisi de dessiner deux triangles par maille, selon la figure suivante:



Concernant l'affichage qui se passe au niveau du viewer (classe simpleViewer), nous aurions pu décider de dessiner les 14 millions de triangles un par un. Cependant, cette pratique n'est pas du tout optimale, puisque les échanges lors du transfert de mémoire entre le processeur et la carte graphique ne sont pas du tout optimisés. C'est pourquoi, nous avons préféré envoyé un tableau de tous les points en une seule fois à la carte graphique, via la fonction *glDrawArrays*. Cette fonction implique cependant de lui envoyer un tableau trié, de l'ensemble des points, dans l'ordre avec lequel on construit les triangles successifs de notre MNT. Les points ont donc été ajoutés via la procédure suivante:



Par ailleurs, on peut ensuite ajouter des tableaux de couleur voire de texture, de la même manière. En effet, comme nous ne disposions pas d'orthoimages de la zone géographique considérée, nous avons décidé d'utiliser un tableau de couleurs, avec trois zones de couleurs (bleu azur

pour la vallée, vert pour la zone intermédiaire et enfin blanc pour la zone de plus haute altitude) et un dégradé de couleurs pour chaque zone d'altitude. Par ailleurs, il s'est avéré pertinent d'utiliser non pas des lignes pour dessiner notre MNT, mais des triangles. Cela nous a, en effet, permis de déterminer visuellement quand une droite intersectait oui ou non le MNT.

2) Calcul de l'intervisibilité

Cette phase de notre projet était la plus importante puisque de son résultat, dépendait la réussite de notre projet. Lors de cette étape, nous avons changé régulièrement de méthode, avant de tomber sur celle qui nous a semblé optimale. Comme, nous disposons des coordonnées planimétriques entre lesquelles, nous devons réaliser le calcul d'intervisibilité, nous n'avons alors, qu'une (petite) partie du MNT à traiter, c'est à dire le rectangle limité par les abscisses et les ordonnées des points sur lesquels, on calcule l'intervisibilité. Cette partie du code, correspond par ailleurs à la méthode *calculateAxis*. Une fois, que l'on a identifié la zone à traiter et que l'on s'est assuré que les deux points considérés ne sont pas sous terre (via la méthode *computeSousTerre*), se pose alors la question de quelle méthode utiliser pour calculer l'intervisibilté. La méthode naturelle consisterait à calculer l'intersection entre la droite formée par les deux points considérés et chaque triangle de la zone du MNT à tester, cependant cette méthode peut s'avérer coûteuse en temps, puisque l'on va tester beaucoup de triangles qui ne seront pas sur le passage de la droite. C'est pour cela, que nous préférons tester en premier lieu les droites horizontales de la zone, en calculant l'intersection de la droite formée par les deux points considérés et les plans horizontaux, puis en comparant l'altitude de cette intersection avec l'altitude du point correspondant sur la droite horizontale. Si l'altitude du point correspondant est supérieure à celle de l'intersection entre la droite et le plan horizontal, il n'y a alors pas d'intervisibilité entre les deux points.
On réalise par la suite le même calcul pour les droites verticales et obliques.
La méthode proposée est équivalente avec celle couramment utilisée pour trouver les triangles concernés par le passage de la droite. Elle présente alors l'avantage d'être analytique, mais surtout d'être de complexité linéaire, ce qui important dans le cadre d'une généralisation du calcul sur plusieurs points.

3) Optimisation du calcul

Même si la méthode de calcul de l'intervisibilité peut sembler optimale. Le fait de généraliser le calcul sur plusieurs centaines de points peut vite devenir long. C'est pourquoi, dans le cas où l'utilisateur choisit de rentrer

un seul point dans l'interface de saisie, on peut paralléliser le calcul de l'intervisibilité, non pas en séparant en plusieurs parties égales notre tableau de points, comme on pourrait le penser intuitivement. Cela ne serait, en effet pas optimal, puisque plusieurs droites identiques seraient alors traitées par des parties différentes du MNT; c'est pourquoi, nous avons décidé de créer 3 threads; le premier, traitant les droites verticales, le second, les droites horizontales et le troisième, les droites obliques. La communication entre les différents threads est alors gérée par le booléen *resultatGlobal*, protégé par une variable de type *QMutex*, qui empêche les différents threads d'agir sur la même variable simultanément. Cependant, même si le calcul de l'intervisibilté fonctionne en multi-thread dans le cas de deux points, nous n'avons pas eu le temps de l'étendre pour le cas de plusieurs centaines de points, à cause de notre inexpérience dans le domaine, ainsi que du long travail bibliographique nécessaire pour cela. Il aurait fallu pour cela, de façon à optimiser au maximum, créer un thread par point, dans la limite de la table de processus.

III. Gestion de projet

1) Fonctionnement en méthode agile

L'équipe de projet s'est efforcée de respecter une gestion de projet selon la méthode SCRUM, dont le cours avait été dispensé quelques jours avant le début du projet. En début de projet, nous avons donc été amenés à définir les différentes fonctionnalités de l'application développée, puis à détailler chaque fonctionnalité en user story, dont la priorité aura été attribuée avec les coefficients de la suite de Fibonacci. Ainsi, compte tenu du temps imparti pour mener à bien ce projet, nous avons identifié 4 sprints, dont les détails sont les suivants:

Sprint1 :

- créer une interface graphique en incluant la zone de dessin ainsi que quelques boutons.
- ouvrir le fichier de coordonnées du MNT au moyen d'une boîte de dialogue et le lire.
- Se documenter sur l'utilisation d'OpenGL pour réaliser un premier affichage graphique dans la zone de dessin.

Sprint 2 :

- Proposer une méthode calcul de l'intervisibilité pour deux points.
- afficher de manière optimiser l'ensemble du MNT.
- ajouter des textures au MNT.

Sprint 3 :

- Coder la méthode d'intervisibilité pour deux points.
- Généraliser la méthode sur plusieurs points.

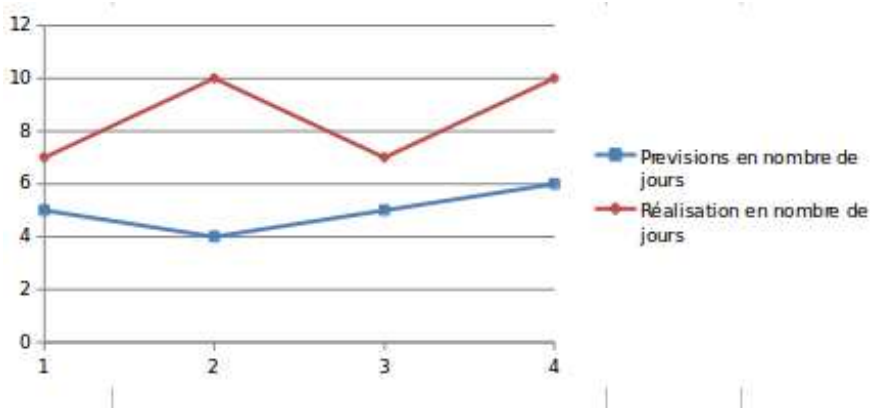
Sprint 4 :

- Optimiser le calcul pour généraliser le calcul à plusieurs centaines de points.
- Ajouter d'autres primitives géométriques pour le calcul de l'intervisibilité (paraboles).

Du fait de notre présence simultanée dans la même salle de projet, il n'a pas été difficile d'organiser tous les jours des réunions (daily stand-up), afin de faire remonter les problèmes éventuels rencontrés par les différents membres de l'équipe.
De plus, au cours de ce projet, nous avons également utilisé le logiciel de versionning Git, afin que chaque membre du groupe puisse avoir accès aux dernières modifications apportées par chacun.

2) Bilan

Le bilan de la gestion de cette gestion de projet est globalement positif puisque, les membres de l'équipe ont travaillé de concert sur ce projet et que le travail de chaque membre de l'équipe a eu une réelle utilité. On peut cependant noter, que d'après le graphique suivant, les user story ont été sous-estimées en temps, puisque le nombre de jours a toujours été dépassé. En particulier, pour le deuxième sprint, où un des deux membres du groupe était absent pour raisons médicales :



Conclusion

Etant donné que la majeure partie des objectifs a été atteinte dans le temps imparti au projet géomatique, le bilan est globalement positif. Cependant, compte tenu de la limitation en temps, l'implémentation des dernières fonctionnalités n'a pas pu être menée à son terme. Pouvoir généraliser le calcul de l'intervisibilité pour plus de de points aurait été intéressant dans le cadre de ce projet.