ENSG Géomatique

ÉCOLE NATIONALE
DES SCIENCES
GÉOGRAPHIQUES

Master : Technologies des Systèmes d'Information

Projet sur le thème : « Modélisation de l'écoulement de l'eau sur un modèle numérique de terrain (MNT). »

Réalisé par : Imane BIH, Clémentine CHASLES, Alban KRAUS

2015 /2016

Sommaire

Introduction

Ce rapport tend à présenter en une dizaine de pages le projet de géomatique réalisé par Imane Bih, Alban Kraus et Clémentine Chasles, élèves de TSI, promotion 2015/2016.

Le sujet du projet sera exposé dans une première partie, qui présentera aussi les contraintes de réalisation imposées, ainsi que les données fournies en entrée et celles produites en sortie.

La programmation de l'outil permettant de calculer ce modèle sera synthétisée et commentée en anglais dans une seconde partie, puisqu'elle porte directement sur le code (et qu'il est de bon ton de commenter son code en anglais pour respecter les bonnes pratiques de développement !).

Enfin, dans une troisième et dernière partie sera expliquée la façon dont a été géré le projet, c'est-à-dire en appliquant au mieux la méthode Agile Scrum.

I. Présentation du projet

1. Le sujet

1.1 La compréhension du sujet

Notre projet avait pour thème : « L'écoulement de l'eau sur un modèle numérique de terrain (MNT) ». À partir du sujet, on a pu déduire que ce projet nécessitait en entrée un modèle numérique de terrain (MNT) fourni par le commanditaire, des connaissances en langage de programmation C++, un ensemble de fonctions pour calculer et afficher le MNT en 3D. Notre projet réalise le calcul du chemin de plus grande pente suivi par une goutte d'eau tombant sur le modèle numérique de terrain. Ce projet pourra être utilisé par exemple pour la prévention des divers risques hydrogéologiques.

1.2. Les fonctionnalités correspondantes

Les fonctionnalités de notre projet consistent à afficher le MNT, à calculer et tracer le chemin d'une ou plusieurs gouttes d'eau, afin de déterminer le point d'arrivée de chacune des gouttes sur le MNT.

2. Les contraintes de réalisation

Au début de notre projet on a trouvé certaines contraintes concernant l'affichage du modèle numérique de terrain (MNT) sur Qt, qui a été imposé par le commanditaire, ainsi que le langage de programmation C++, donc pour faciliter la tâche, on a choisi de travailler avec une librairie basée sur Qt qui simplifie l'affichage du MNT en 3D.
Le choix de la version Qt5 au lieu de Qt4 était pour s'entrainer avec les pratiques les plus modernes.
On a pensé aussi à faire de la triangulation avec une bibliothèque CGAL mais c'était très lourd pour nous puisqu'elle contenait plus de fonctionnalités par rapport à ce dont on avait besoin. Finalement, on a décidé de coder nous-même la triangulation puisqu'on a besoin de connaitre comment se calcule la triangulation au niveau des points.

Proposition de clémentine :
L'utilisation de Qt a été imposée par le commanditaire, ainsi que le langage de programmation C++. Pour faciliter la tâche, nous avons choisi de travailler avec une librairie basée sur Qt qui simplifie *l'affichage du MNT en 3D : libqglviewer.*
Le choix de la version Qt5 au lieu de Qt4 était pour s'entrainer avec des pratiques plus modernes.
Nous avions aussi pensé à faire de la triangulation avec une bibliothèque CGAL mais elle était très lourde par rapport à ce dont nous avions besoin. Finalement, nous avons décidé de coder nous-même la triangulation.

3. Les données

3.1. Les données d'entrée

La donnée en entrée est un nuage de points fourni sous la forme d'un fichier en format texte portant l'extension .xyz. Le fichier de coordonnées, d'une taille de 215,0 Mo, contient à peu près 7 millions de points. Les coordonnées (X, Y, Z) sont triées par Y décroissant puis par X croissant. Les coordonnées sont séparées par un espace. Le Z définit l'altitude du point. Pour mieux préciser, il y a un pas de 25 m entre les points. Les points ont comme projection Lambert II étendu, qui couvre toute la France métropolitaine.

Pour lancer le calcul, l'utilisateur choisit un point de départ.

3.2. Les données en sortie

Les résultats obtenus par ce projet sont un affichage en 3D du MNT, avec une triangulation et un dégradé de couleur, et le tracé des chemins jusqu'à leurs points d'arrivée. L'affichage du chemin est personnalisable en couleur et épaisseur via un menu, et exportable dans un fichier image. Ensuite on exporte les coordonnées des chemins dans un fichier texte.

II. Partie technique

1. Overview

The project is written in standard object-oriented C++ with a few c++11 extensions. It uses the Qt framework, version 5; as this is a student's project, a modern version of the framework was chosen to anticipate the future needs. The framework has been extended by a library, libQGLViewer, which handles 3D-rendering in OpenGL and provides a camera control. This library is linked to Qt5 in its most recent version.

The project aims at computing the path of a water drop from a user-defined starting point on a digital terrain model (DTM). Its main features therefore are:

- Display a digital terrain model in 3 dimensions
- Let the user select a starting point
- Compute and display the flow path from that point.

The project is made of 8 classes; three of them handle the data and data-related operations, and the five other handle specific parts of the interface. The overall design is a bad Model-View-Controller, where each datatype implements the operations made on it.

2. Classes

2.1. Comments on the GUI

The graphical user interface uses two main windows: a QMainWindow that have all the buttons and menus, and a viewer that can only display the DTM. The former is called

MainWindow and inherits QMainWindow, and the latter is called glDisplay and inherits QGLViewer. There are additional windows, namely: a progress bar, a context menu for the history dock, a path customization window and its color chooser dialog, and several file dialogs. The last two are created and destroyed in the same function, so they don't need to be recorded as a pointer. Generally speaking, the MainWindow keeps pointers to all the windows used by the program. Some of them are instantiated in the constructor and thus are kept with a * const. They use signals and slots extensively.

During computations, the GUI can be *locked*, which means that some components are grayed out.

To select a point, the user has two methods. By clicking the Selection Mode button, they can afterwards click on the viewer without changing the point of view. If the clicked pixels displays a part of the DTM (QGLViewer and OpenGL handle the conversion 2D-click to 3D-point), the coordinates boxes are updated, and otherwise an error message is displayed. The user can also directly enter coordinates in the boxes. In any method, the computation begins when the user presses the Computation button. It first tries to find the nearest DTM vertex. If no DTM vertex is found, that means the coordinates given are invalid, and an error message is returned to the user. Otherwise, these coordinates are passed to a newly-constructed FlowPath object, which is added to the list.

2.2. Comments on the viewer

The viewer is responsible of drawing everything: the DTM and the flow paths. It accesses them through the MainWindow, by keeping pointers to MainWindow's pointers. If the DTM or the FlowPath list changes, they are updated in MainWindow, and then there is no need to spread the change to the viewer.

The QGLViewer redraws the display only upon a user interaction that activates the window. To allow the immediate rendering after a DTM or FlowPath change, MainWindow provides signals that are connected to glDisplay slots. In the case of a DTM change, the scene bounding box needs to be updated, so the `init` function must be called. But if the viewer is hidden, the `init` function cannot do anything, so a boolean `m_initialized` is used to record whether the `init` function is to be called before the next drawing or not.

2.3. Comments on the drawables

Every object that is to be drawn inherits the Drawable class that provides a consistent interface for the viewer. This class keeps track of the data vertices, and builds the arrays for use with openGL (see below).

In the program, two way of using OpenGL is supported: the legacy way using `glVertex` and `glColor`, which is called for us MODE_LEGACY, or more efficient ways by sending an array with all vertices in the right order for drawing and a color array in the same order (MODE_VERTEX_ARRAY), or sending unaltered vertices and color arrays and an array which only contains in the right order the indices of the vertices to be drawn (MODE_VERTEX_INDICES). The latter is the most lightweight way.

Using any method, the vertices need to be processed in the same order: to call `glVertex`, to append them to the vertices array, or to append their indices to the indices array. The Drawable class provides for its children two build functions (a const for legacy openGL and a non-const to update the arrays) that process a given index according to the drawing mode. The two child classes needs to call these function in the right order. It is relatively simple for FlowPaths, but as the DTM is a grid, it must be drawn line-by-line.

For the DTM, the building process is divided in four steps. A function is called at the very beginning and the very end. Then, all vertices are processed one by one (with a `build_line` function), with a halt at each end of line to draw back to the beginning of the following line. The backwards movement follows the vertices, otherwise there are bad lines drawn under the DTM.

These functions are templates, because their action depends on the primitive used to draw the DTM (points, lines that make triangles, lines that make squares, filled shapes, and other more or less supported openGL primitives).

III. La gestion de projet

1. Le profil du groupe

1.1. Présentation des membres du groupe

Le groupe du projet MEGAFI se compose de trois personnes aux profils très différents :

- ✓ Imane, ingénieur en Géoinformation, avec des compétences en réseau et télécoms ainsi que dans les métiers du net ;
- ✓ Alban, étudiant ingénieur de l'ENSG féru d'informatique ;
- ✓ Clémentine, géographe-géomarketeuse en reprise d'études après six ans en entreprise.

1.2. L'organisation entre les membres

Naturellement, Alban s'est imposé comme chef de projet : son niveau avancé en programmation et son expérience en réalisation de projets informatiques ont justifié cette position.
Ces mêmes qualités associées à sa patience et sa pédagogie ont permis de guider le groupe sereinement vers l'achèvement du projet.

La compréhension du sujet par la reformulation des besoins en fonctionnalités s'est faite conjointement entre les membres du groupe et le commanditaire à chaque étape du projet.
Chacun s'est montré force de proposition (et d'acceptation), autant dans le choix de ces fonctionnalités que dans leur mise en œuvre. Les développements ont été attribués en concertation collective, en fonction des compétences de chacun et des volontés d'apprentissage. L'œil avisé d'Alban a été d'une aide précieuse à Imane et Clémentine pour les faire appréhender en toute autonomie les nouveaux langages et outils de programmation.

2. La gestion de projet selon la méthode Agile SCRUM

2.1. Le fonctionnement général

L'équipe s'est efforcée de respecter une gestion de projet proche de la méthode Agile SCRUM.

En début de projet, elle a réfléchi au sujet et a listé les fonctionnalités (*features*) qu'elle souhaitait développer.
Ensuite, elle a détaillé chaque feature en user stories, a attribué à chaque user story une priorité, un nombre de points selon la suite de Fibonacci, pour finalement en faire un backlog de référence.
Au total, 232 points ont été comptabilisés pour 30 user stories. Des user stories "bonus" ont aussi été proposées au développement en fonction de l'avancement du projet.
Au regard de ces éléments et des 20 jours consacrés au projet, l'équipe a défini 3 sprints de respectivement 5, 8 et 7 jours chacun, le premier sprint devant servir de repère pour ajuster ces choix et éventuellement reprendre le nombre de points prévus pour chaque user story.

Le premier sprint a mis en place le mode de fonctionnement de l'équipe sur la totalité du projet.

Au début de chaque sprint, le backlog était repris, les user stories se voyaient attribuer un temps de développement estimé en fonction de leurs points, du nombre de membres de l'équipe et du nombre de jours dans le sprint.

Chaque jour ou presque, en début d'activité, l'équipe tenait un daily stand-up où chacun décrivait les tâches accomplies, le temps passé, les problèmes rencontrés. En fonction, chaque user story était proclamée EC (En Cours), OK, KO (abandonnée) ou Reportée. Si d'autres tâches que celles liées aux user stories avaient été faites (debug, recherche documentaire plus approfondie, nouvelles fonctionnalités nécessaires, etc.), elles étaient simplement ajoutées au backlog du sprint en cours. En cela, ce dernier en est vite venu à différer du backlog de référence.
Chaque soir, les membres de l'équipe poussaient sur le dépôt Git de l'école leurs *commits* en fonction de leurs réalisations du jour. Le lendemain matin, chacun pouvait donc faire un *pull* pour se mettre à niveau dans le code.

À la fin de chaque sprint, un sprint review était organisé avec l'équipe et le commanditaire pour faire un point sur le sprint passé, une démonstration des fonctionnalités achevées, l'avancée du projet et les user stories restantes à développer.

Pour débuter le prochain sprint, le backlog était repris à partir des fonctionnalités en attente de développement.

2.2. Le détail des sprints

Le premier sprint a globalement été consacré à :

- prendre en main l'environnement de travail via les mises à jour logicielles liées à Qt et OpenGL, ainsi qu'à l'ouverture d'un dépôt Git sur le serveur de l'école et Github sur internet ;
- le dessin de la fenêtre principale de l'outil, avec une première barre de menu et les premières connexions de fermeture ;
- la création d'une boîte de dialogue permettant de sélectionner le fichier de coordonnées du MNT, de l'ouvrir et d'en lire le contenu ;
- se documenter en amont sur comment afficher le MNT ainsi lu à l'écran grâce aux possibilités d'OpenGL.

Le travail d'équipe de ce premier sprint fut clairement commun : Alban le Product Owner a passé le plus clair de son temps à guider Imane et Clémentine dans leurs premiers développements, si simples fussent-ils. Mais c'est ainsi que ces deux dernières ont pu réussir à créer les fonctions se rapportant à l'ouverture et à la lecture du fichier.

Au deuxième sprint, le sujet a été attaqué en plein cœur :

- Alban a utilisé et installé la librairie libqglviewer pour afficher le MNT à l'écran selon les primitives TRIANGLE et QUAD, dans un premier temps ;
- Imane s'en est brillamment sortie en codant l'algorithme du parcours récursif sur les voisins permettant de calculer le chemin d'écoulement d'une goutte ;
- Clémentine a satisfait ses élans créatifs en agrémentant l'interface graphique de nouveaux éléments et connexions d'affichage du MNT, et du calcul du chemin d'écoulement, notamment grâce à la récupération des coordonnées du MNT sous un clic souris.

Le sprint final a permis d'achever un outil fonctionnel mais n'intégrant cependant pas toutes les features prévues au départ :

- l'équipe a passé beaucoup de temps à corriger des bugs soulevés par la revue de code (code review) des uns par les autres ;
- Alban en a profité pour refactoriser le code existant en l'améliorant scrupuleusement ;
- avec Imane, ils ont apporté leur pierre à l'édifice en permettant l'affichage d'un MNT et d'un chemin d'écoulement tout en délicatesse et en couleur ;
- Clémentine quant à elle a rendu la vie du futur utilisateur de l'outil bien plus agréable, en créant une barre de progression l'invitant à patienter ou encore en affichant les étapes du déroulement des opérations dans un log ;
- c'est aussi à Alban que l'on doit la possibilité d'afficher l'historique des chemins calculés dans un caisson prévu à cet effet, à Clémentine d'exporter les coordonnées des chemins d'écoulement dans un fichier, et à Imane d'exporter une capture d'écran et de visualiser une icône du projet MEGAFI digne de ce nom !

3. Bilan

À la fin du dernier sprint, MEGAFI est un outil simple mais fonctionnel, et qui a a priori atteint son objectif : proposer un modèle d'écoulement d'une goutte d'eau sur un MNT.
Certes sur les 232 points de fonctionnalités prévues, 84 ont été ignorés, sans compter ceux des fonctionnalités « bonus ». Parmi eux, l'équipe regrette la possibilité d'afficher un chemin d'écoulement animé et progressif, le sous-échantillonnage du MNT pour un affichage plus rapide ou encore le calcul des bassins-versants.

Le bilan de cette gestion de projet reste globalement positif. L'équipe a su avancer de concert dans une communication saine et productive. Alban a joué le rôle d'un parfait Product Owner mettant Imane et Clémentine en confiance dans leurs initiatives de développement.

- l'équipe s'est tenue à ses quasi-daily stand-up et sprints reviews de façon régulière, même en voyant ces derniers différer du backlog de référence ;
- le code fait par chacun a été expliqué aux autres et commenté un maximum, et ce en anglais afin de s'entraîner à produire un code international. Chaque membre était au fait de ce que faisait l'outil, comment, dans quelle classe se trouvait telle méthode, pourquoi. Même si le contenu de certaines classes n'était pas toujours très bien compris dans le détail par Imane et Clémentine (car complexe !), l'idée principale et l'organisation avec les autres classes étaient assimilées et ne les empêchaient ainsi aucunement d'avancer ;
- le code reviewing et les tests manuels ont permis de détecter un certain nombre de bugs à corriger ;
- le système de versioning Git a facilité l'accès au code et son partage ;
- le travail de chaque membre de l'équipe a eu une réelle utilité.

Quelques points négatifs sont également à relever dans ce bilan :

- peu de tests ont été mis en place. Un seul test unitaire sur la lecture du fichier de coordonnées a été rédigé ;
- il n'y a pas eu d'analyse statique et dynamique de code autre qu'humaine. Les outils d'aide au développement spécifiques comme CPP Check ou Visual Leak Detector n'ont pas été utilisés ;
- si la méthode SCRUM a été globalement et régulièrement appliquée, la mise à jour des points et des temps estimés en fonction de l’apparition de nouvelles fonctionnalités au cours des sprints n'a pas été faite, ce qui rend difficile l'analyse de l'existant avec le prévisionnel.

Conclusion

Le modèle d'écoulement de l'eau sur un modèle numérique de terrain a été achevé dans les temps impartis au projet de géomatique.

Il a atteint le but prioritaire que l'équipe s'était fixé au départ, celui de pouvoir calculer le chemin d'écoulement d'une goutte d'eau lâchée à la surface du MNT, et d'afficher son tracé sur ce dernier.

La limitation de temps au regard des différences de niveaux en informatique de chacun a freiné le développement de fonctionnalités supplémentaires qui se seraient avérées intéressantes pour une utilisation plus avancée de ce modèle.

Pouvoir par exemple afficher le remplissage du MNT selon les chemins d'écoulement successifs calculés pour ainsi rejoindre le modèle d'inondation fait par l'équipe en ayant la charge, aurait été une piste riche pertinente à exploiter.