Bontempi Hugo
Gauer Louis-Marie

18/05/2018

Document de synthèse :

Notre projet était de créer une application Java permettant la lecture d'un fichier MNT. A partir de ce fichier, notre application devait calculer des courbes de niveau avec différents pas en vu d'un affichage graphique du MNT ainsi que des courbes de niveau calculées.

I) Organisation & problèmes rencontrés

Nous avons consacré les premières séances du projet à l'analyse du projet. Le but était de trouver un agencement logique entre les différentes classes. Nous avons donc commencé par rechercher quelles classes nous seraient utiles ainsi que leurs attributs et méthodes. Une fois ce travail d'étude et de réflexion effectuée, nous avons pu poursuivre avec l'élaboration du diagramme de classes ainsi que des autres diagrammes nécessaires au codage de notre application. Le premier problème que nous avons rencontré a été la mise en place du diagramme de classe. Il nous a fallu trouver un moyen efficace de stocker les données au travers des classes pour poursuivre le travail d'analyse des autres diagrammes. La suite et la construction des autres diagrammes ont été facilitées une fois le diagramme de classes fini (même si nous avons dû le remodifier par la suite).

Une fois les diagrammes construits nous avons recherché et imaginé les méthodes et algorithmes permettant de calculer les courbes de niveau à partir des différents points du MNT et des altitudes.

Il y avait deux méthodes principales :

- La première méthode consiste à parcourir le tableau de point altimétrique du MNT. Pour chaque point, on créer un triangle avec deux points voisins du point observé. L'intersection de ce triangle avec un plan horizontal pour une altitude constante donne un bout de la courbe de niveau. Ainsi en affichant tous les segments calculés on affiche les courbes de niveaux fermées et entières.
- La seconde méthode consiste aussi à parcourir le tableau de point altimétrique du MNT. A chaque point observé on regarde si son altitude et l'altitude de ses voisins (un par un) encadrent l'altitude de la courbe de niveau considérée. Ainsi, on obtient un encadrement de la courbe de niveau d'altitude considérée et on peut donc calculer des points de la courbe de niveau. Cela permet de calculer une courbe de niveau fermée de proche en proche.

Nous avons d'abord opté pour la première méthode qui semblait plus simple à implémenter. Nous nous sommes donc séparés la charge de travail. Le premier a commencé à créer le package de gestion de MNT, avec le calcul de MNT le centre de notre projet. La première chose à faire était de créer une classe permettant l'import de données dans l'application selon un format de fichier. Nous avons choisi de créer en premier un import possible des fichiers « .asc » qui étaient faciles à trouver sur le site de l'IGN. Durant ce temps, le second a commencé à créer un package Geometric implémentant des objets et outils de géométrie. Le but de ce package était de pouvoir créer et surtout calculer des intersections mathématiques. La classe devait permettre de créer des objets : Point, Droite, Plan puis Segment (héritant de Droite mais avec un début et une fin) et Triangle (héritant de Plan mais avec lui aussi une limite).

Pour implémenter l'intersection nous avons calculé mathématiquement l'intersection de deux plans dont les équations cartésiennes sont connues. Il est apparu qu'il y avait trop de conditions sur les

coefficients des plans pour pouvoir coder simplement les méthodes. De plus, nous n'avons pas trouvé la solution pour limiter des objets de la classe Plan et Droite simplement afin de créer les classes Triangle et Segment. La classe Geometric n'étant pas le centre de notre projet, nous avons donc décidé d'arrêter le développement de ces outils et à la place d'en rechercher un déjà implémenté.

Au final, et à la place d'utiliser des outils que nous n'avions pas implémentés, nous avons décidé de changer de méthode de génération des courbes de niveau. Nous nous sommes donc de nouveau intéressés à la seconde méthode.

Nous avons choisi de diviser la charge de travail en une partie génération des courbes de niveau et une partie visualisation du MNT et affichage graphique, la génération du MNT étant déjà fonctionnel selon deux méthodes (l'import d'un fichier « .asc » et la génération d'un MNT aléatoire avec en paramètre l'altitude minimale et l'altitude maximale, ainsi que les dimensions du MNT).

La partie génération des courbes de niveau a apporté beaucoup de problèmes successifs. Le premier était de pouvoir afficher et trouver des points présents sur les courbes du MNT. Grâce à l'écriture d'une méthode recherchant les voisins qui permettent d'encadrer une altitude à partir d'un point voisin nous avons pu résoudre ce problème. Le problème suivant a été de réussir à fermer les courbes de niveau. En effet, en parcourant le tableau de données altimétriques on ne peut pas créer des courbes de niveau fermées car on ne sait pas si deux points calculés successivement appartiennent à la même courbe de niveau ou non. Il a donc fallu imaginer un algorithme permettant de suivre une courbe de niveau et ainsi de pouvoir la fermer. L'élaboration de cet algorithme a été compliqué et a pris du temps. Mais nous avons fini par réussir à le créer et à l'améliorer en évitant de parcourir des cases plusieurs fois pour accélérer l'algorithme. Cette méthode a été longue à implémenter et à rendre effective, il y avait des problèmes dans les indices entre les deux axes du tableau de données de MNT.

De l'autre côté, la partie visualisation du MNT et affichage des courbes de niveau a elle aussi apporté plusieurs problèmes. Le premier problème auquel il a fallu faire face était que l'on ne connaissait aucun outil d'affichage graphique sur Java. Il a donc fallu découvrir tout ces outils et se les approprier ce qui a pris beaucoup de temps. De même la découverte des outils de création d'une interface avec des boutons a posé les mêmes problèmes.

II) Fonctionnement :

1) Description des différentes classes :

Nous avons décidé d'organiser nos classes en trois packages pour faciliter la clarté et l'accès aux parties du code qui nous intéressent.

- **Affichage** :

Affichage est le package qui contient toutes les classes nécessaires à l'affichage des résultats graphiques ou non. Il contient trois classes.

- *Main* :

Main est la classe qui gère tout le programme. Elle appelle les constructeurs, créé les objets, lance les affichages graphiques, génère les boîtes de dialogue… C'est elle qu'il faut exécuter pour lancer l'application.

- *Panneau* :

Panneau est la classe qui affiche les MNT ainsi que les courbes de niveau associées à ceux-ci. Son constructeur permet d'initialiser les grilles en fonction des paramètres entrés par l'utilisateur.

- **Generation** :

Generation est le deuxième package. Il possède l'ensemble des classes et des méthodes permettant de générer l'ensemble des courbes de niveau. Les données des courbes de niveau sont stockées dans les instances de ses classes.

- *Relief* :

La classe *Relief* est la classe qui représente l'objet contenant l'ensemble des données des courbes de niveaux. Son attribut table possède l'ensemble des plans altimétriques pour les différentes altitudes de génération des courbes de niveau, il s'agit d'une ArrayListe<*PlanAlti*>.

- *PlanAlti* :

PlanAlti est la classe juste en dessous de *relief*. Elle représente l'ensemble des courbes de niveau fermées du *MNT* à une altitude fixe. Elle possède un attribut courbes qui contient l'ensemble des *CourbeNiveau* fermées de l'altitude du *PlanAlti*.

- *CourbeNiveau* :

CourbeNiveau est la classe qui représente une courbe de niveau fermée à une altitude déterminée. Elle est stockée dans la liste courbes, attribut de *PlanAlti*.

- *Coord* :

Coord est une classe contenant deux attributs : un attribut x et un attribut y. Ces deux attributs forment un objet Coord , la coordonnée d'un point. C'est le plus petit objet composant les objets Relief.

- **Importation** :

La Classe **importation** est le premier package créé. Il contient l'ensemble des classes permettant l'importation des données à partir d'un MNT ou la création d'un MNT de façon aléatoire.

- *Alea* :

La classe *Alea* est la classe qui gère la génération d'un *MNT* aléatoire. Un de ses constructeur prend en paramètres : l'altitude minimale, l'altitude maximale et la taille du *MNT* à générer.

- *ASC* :

La classe *ASC* permet d'importer des fichiers de type « .asc » en lisant les données et en les structurant dans un objet *MNT* (*ASC* hérite de *MNT*), ainsi que de généraliser les méthodes par la suite.

Un objet de type *ASC* peut être créé à l'aide d'un constructeur qui prend en paramètre le chemin d'accès au fichier de données.

- *Import* :

La classe *Import* est la classe qui devait gérait l'ouverture d'un fichier. Par manque de temps et comme nous n'avons fait qu'une classe d'ouverture de fichier. La lecture de fichier se fait donc dans la classe *ASC*.

- *MNT* :

La classe *MNT* est l'aboutissement de **Importation**. Elle permet de structurer les données comme nous l'avions décidé au début de projet et ainsi de généraliser les méthodes appliquées aux objets de type *MNT*.

2) Lancement et exécution du programme :

Pour lancer l'application, il faut importer le projet dans le workspace et ouvrir la classe Panneau, puis lancer le script.

Ou bien simplement lancer le fichier Launch.bat.

Une fois l'application lancée, un message va apparaître et demander la méthode d'importation des données (Aléatoire ou Fichier .asc). Une fois celle-ci renseignée l'application va demander successivement dans des boîtes de dialogue les paramètres nécessaires à la création des différents objets. Enfin, la dernière fenêtre de dialogue demande si l'utilisateur souhaite afficher le MNT ou bien les courbes de niveau.

III) Bilan du projet :

Durant tout le projet, apprendre et découvrir plus en profondeur Java et ses outils nous a pris beaucoup de temps. Nous aurions aimé développer un peu plus notre application. Voilà les différents outils et développements que nous aurions souhaité faire :

- **Plus de classes d'importation :**

Lors de l'analyse du sujet nous avons décidé de commencer par implémenter la méthode d'ouverture des fichiers « .asc » qui étaient facilement accessible. Mais nous avons trouvé d'autres extensions de fichiers MNT et avons décidé d'en implémenter plusieurs. Au final seul l'ouverture des fichiers « .asc » a été codée.

- **Affichage graphique plus développé :**

La découverte des différentes bibliothèques d'affichage nous a demandé beaucoup d'attention. De plus, l'implémentation de la génération des courbes de niveau ayant pris beaucoup de temps, les possibilités de tester l'application avec les résultats de la génération ont ralenti le développement de l'affichage graphique.

- **Affichage de plusieurs MNT côte à côte :**

Nous avions stocké dans un attribut de MNT le point ou le MNT était géoréférencé. Le but était à la fin de pouvoir afficher plusieurs MNT côte à côte en connaissant leur position dans une projection. Ayant eu des difficultés avec l'affichage graphique, nous avons très rapidement abandonné cette option mais avons laissé les attributs dans le cas d'un développement futur.

- **Superposition des courbes de niveau et du MNT :**

Nous avons réussi à afficher les deux objets séparément mais lorsque nous superposions les deux objets, les courbes de niveau et le MNT n'étaient pas affichés avec la même échelle ce qui empêchait de visualiser correctement les deux objets, raison pour laquelle nous avons séparé l'affichage des deux objets avec une boîte de dialogue.

IV) Conclusion :

Pour finir, ce projet nous a faire prendre conscience de l'importance de l'organisation. La répartition du travail nous a permis d'avancer plus vite, chacun dans les parties que l'on préférait et dans lesquelles on se sentait le plus à l'aise.

Néanmoins, nous avons eu quelques difficultés à trouver un outil simple pour coder en même temps sur le projet. Au début du projet, ce problème ne se posait pas car nous travaillions sur deux packages différents. Mais lorsque nous avons commencé à travailler sur les mêmes packages nous avons dû trouver un moyen facile pour pouvoir concaténer nos travaux. Nous n'avons pas utilisé GitHub car nous préférions avancer dans le projet plutôt qu'apprendre à utiliser un nouveau logiciel. La première méthode que nous avons utilisée consistait en la récupération des classes sur lesquelles l'autre avait travaillées suivi du remplacement de celles que l'on avait dans notre workspace. Ensuite, pour être plus efficace nous avons commencé à travailler sur Dropbox avec en arrière-plan un programme Python qui faisait une sauvegarde toutes les 5 minutes pour éviter les pertes de parties de notre projet en cas de problème.

Au final, nous avons réussi à rendre un fichier.jar avec une interface. L'utilisateur n'a donc pas besoin de parcourir le code pour lancer l'application.