Visualisation d’un MNT et calcul de lignes de niveaux

**I. Organisation du projet**

Nous avions pour objectif la création d’un MNT visualisable en 2D ou 3D. De plus, sur ce MNT nous devions pouvoir afficher des courbes de niveaux à des altitudes choisies.

Dès la première séance nous avons créé un projet Github afin de pouvoir tous deux travailler en parallèle sur le code, retrouver les anciennes versions du code mais surtout assurer des sauvegardes régulières du programme. Les premières séances ont été consacrées à la compréhension du sujet et à son analyse. Le sujet nous est rapidement apparu complexe. En effet, les méthodes à mettre en place n’étaient pas définies explicitement. Il nous a fallu faire beaucoup d’hypothèses quant aux attentes vis-à-vis de ce projet.

Dans le but de nous organiser efficacement, nous avons décidé de faire un diagramme de GANTT. Cependant, par manque d’expérience, nous ne pouvions pas quantifier exactement la durée des différentes étapes de programmation. Le GANTT s’est donc montré rapidement obsolète.

Nous avons, autant que possible, rédigé les rapports en parallèle de la programmation. De cette façon, nous étions sûr de ne pas prendre de retard dans la rédaction de ces rapports, et cela nous assurait également de ne rien oublier d’important pour le bilan du projet.

Enfin, lors de la programmation, nous avons défini des méthodes *main* dans chaque Classe afin de débuguer nos fonctionnalités au fur et à mesure en réalisant des tests unitaires. Ces *main* sont toujours présents à titre informatif même s’ils ne présentent plus forcément d’utilité.

**II. Réussites et améliorations**

Nous sommes partie avec l’hypothèse qu’un MNT est un nuage de points de 3 coordonnées (XYZ) et que les points ne sont pas forcément séparés par un pas régulier. Nous avions à cœur de créer un outil qui pourra être utilisé dans divers scénarios, et plus particulièrement si le pas du MNT est irrégulier. Nous avions aussi pensé à représenter le MNT par une matrice dont l’indice de la ligne et de la colonne correspondrait aux coordonnées XY et la valeur de la matrice pour cet indice correspondrait à l’altitude Z. Cependant cette hypothèse nous paraissait moins représentative de la réalité, c’est pourquoi nous choisîmes la première solution.

Nous devions pouvoir visualiser le MNT en 2D et en 3D. L’affichage graphique en Java était une nouveauté pour nous deux. Nous avons d’abord mis beaucoup de temps à comprendre le fonctionnement de *java2d* et des classes qui lui sont associées comme *JPanel.* Nous avons finalement trouvé comment afficher une fenêtre vide, une forme géométrique primitive et enfin un nuage de point simple. Pour ce faire, nous avons pris comme base un code déjà fonctionnel, tiré d’articles universitaires et de tutoriels (OpenClassrooms, univ-lemans…), que nous avons ensuite modifié pour s’adapter à nos besoins.

L’affichage en 3D nous a posé bien plus de difficulté. Nous avons tenté d’utiliser la bibliothèque *JOGL* de Java mais n’avons pas pu aboutir à des résultats suffisamment satisfaisants pour l’affichage du MNT. Nous avons seulement pu créer, en suivant différents tutoriels, une forme géométrique 3D basique (cf le package *Affichage3D).* Nous avions aussi hésité avec d’autre librairie Java pour la 3D (Java3D, …). Mais ces autres librairies étaient plus complexes à prendre en main et nous n’avons pas eu le temps de suffisamment nous y attacher.

Un autre élément qui nous a posé quelques problèmes fut le chargement puis l’affichage d’un MNT déjà existant. Nous avions à notre disposition la BD altimétrique IGN au format *GRID ASCII.* Il fallut donc comprendre le format du fichier asc puis le convertir en une liste de points 3D. On retrouvera cette partie du code dans la classe « PointImport »

**III. Partie technique**

1) Description (non exhaustive) des packages et fonctionnalités

­Le programme est décomposé en 4 packages. Le package principal, *ProjetMNT* comprend les points, les courbes, et le MNT dans son ensemble.

Le deuxième package *Affichage2D* permet la visualisation des points et des courbes du MNT dans un plan.

Le troisième package quant à lui, se nomme *Affichage3D* mais reste incomplet (cf : Réussites et amélioration) car nous n’avons pas eu le temps de développer les fonctionnalités en 3D.

Enfin un 4eme package *Execution* a été créé très récemment (il ne figurait pas sur le diagramme de classe du rapport d’analyse). Il permet de lancer le programme selon les besoins utilisateurs.

La réalisation du MNT repose sur certaines fonctionnalités principales que nous allons décrire brièvement :

* *pointAltitude*

D’abord, nous avons cherché à créer des « points intermédiaires » à des altitudes précises. En effet un des premiers problèmes auquel nous avons été confronté est que nous voulions des points à des altitudes précises afin d’en tracer la courbe de niveau. Cependant Il arrivait que nous ayons des points à des altitudes pas strictement égales à celle de la courbe de niveau souhaitée. Nous avons eu l’idée de créer de nouveaux points correspondant à cette altitude. Par exemple, à partir de deux points aux altitudes respectives 90m et 110m, nous pouvions créer un point à 100m d’altitude (en considérant que l’altitude varie linéairement entre ces deux points) et situé entre ces deux points. Nous pourrons ensuite y faire passer une courbe de niveau la plus précise possible.

* *planAltimetrique*

Initialement, nous avions distingué 2 types de courbes de niveaux (en nous basant sur des cartes IGN de base). Les courbes dites « ouvertes », c’est à dire qui sortent du cadre du MNT, représentées par une ligne ouverte. Et les courbes « fermées » représentées par une forme circulaire sur les cartes. Mais faute de temps, nous n’avons finalement pas pu poursuivre cette idée, nous avons donc travaillé qu’avec un seul type de courbe : les courbes ouvertes. Cela ne gêne pas pour autant lors de l’affichage.

|  |
| --- |
| * *planAltimetrique*   Cette méthode permet de différencier plusieurs courbes de niveaux de même altitude à partir d'un nuage de point. Dans un premier temps, il faut trier la liste donnée en argument de telle sorte que les points de la liste correspondent à la succession des points du dessin. Ensuite, il s’agit de calculer les distances entre les points respectifs de cette liste et comparer ces distances. Si une distance est jugée trop élevé (selon des critères statistiques). La liste triée sera séparée en deux. On appliquera alors une nouvelle fois cet algorithme jusqu’à avoir un nombre n de listes où tous les points seront suffisamment proches pour considérer qu’ils forment une unique courbe de niveau. Nous avons en fait une méthode récursive. |
|  |

|  |
| --- |
|  |

* *MTNImport*

Cette fonction a été conçue en se basant sur le format GRID ASCII. La méthode permet d’ouvrir le fichier asc , de le parcourir à l’aide d’un buffer et d’ajouter les points en coordonnées XYZ dans une liste (List<Point>) qui servira plus tard à l’affichage. Nous avons pris comme test unitaire, le MNT de la Réunion (car étant le moins volumineux) fourni par la BD ALTI® de l’IGN.

2) Exécution du programme

Pour exécuter le programme, il faut se rendre dans le package *Execution* qui ne contient qu’une seule classe, la classe *Main.* Une fois compilée, il suffit de suivre les instructions données par la console en fonction des besoins utilisateurs.

Des conseils pour éviter les bugs et les temps de calcul trop long seront donnés par la console.

**III. Bilan**

Nous aurions naturellement aimé avoir plus de temps afin d’affiner nos résultats, mais aussi pour développer de nouvelles fonctionnalités répondant aux problèmes posés. Cependant nos fonctionnalités permettent d’ores et déjà la création, le chargement et la visualisation d’un MNT de base, ainsi que l’affichage des courbes de niveau qui lui correspondent.

Ce projet nous aura permis de revoir les bases de modélisation UML tout en apportant un contexte concret à celle-ci. L’utilisation de *Modélio* et *Star UML* fut une nouveauté pour nous deux.

Pour ce qui est du développement informatique, nous avons appris qu’il est très compliqué de prévoir le temps que prendra le codage d’une fonction. En effet de nombreux bugs imprévus apparurent inopinément, ce qui fut très chronophage. Nous avons aussi découvert l’importance de la documentation car en cherchant des algorithmes intéressants pour l’affichage 2D sur internet, nous nous sommes heurtés à des programmes peu/pas documentés et donc incompréhensibles pour un utilisateur novice.

Enfin en termes de gestion de projet nous avons pu avoir un aperçu de ce qu’est un travail de groupe. Nous avons appris à nous organiser et à travailler en parallèle sur un même programme.