AVIGNON UNIVERSITÉ

CERI Map

Group 5
Yingqi LUO
Sara OUMOHAND

20 mars 2022

L3 informatique
Ingénierie Logiciel

Responsables
Prénom5 Nom5
Prénom6 Nom6
Prénom7 Nom7

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

1 Introduction

Dans le cadre de l'UE projet programmation, on a dévelopé le logiciel CERI Map est un logiciel en Java,le but de ce projet est de produire un logiciel reproduisant une partie des fonctionnalités proposées par des services de cartographie, Il se basera sur les données OpenStreetMap via l'Api overpass.

Ce projet implémente un certain nombre de fonctionnalités tel que rechercher d'un endroit, en entrant l'adresse ou les coordonnées GPS ou bien d'un iténeraire , affichage d'une carte ...

2 Données OpenStreetMap

OpenStreetMap, est une base de données géographique, mondiale, libre, gratuite et collaborative. Les données de cette base sont donc libres d'accès sous condition d'accepter la licence ODbL. Les données cartographiques dans openStreetMap sont représentées de trois façons différentes[1]:

2.1 Le noeud (node) :

Représente un point spécifique sur la surface de la Terre, identifié par la latitude et la longitude. Certains nœuds servent à décrire la forme et l'emplacement de certains objets sur la carte, tels que : des bâtiments, des routes, certains équipements et des lieux publics. Il existe des nœuds servant à déffinir l'emplacement des lieux important. Représente un point spécifique sur la surface de la Terre, identifié par la latitude et la longitude. Certains nœuds servent à décrire la forme et l'emplacement de certains objets sur la carte, tels que : des bâtiments, des routes, certains équipements et des lieux publics. Il existe des nœuds servant à déffinir l'emplacement des lieux important.

2.2 La ligne (way) :

Le way est formé de 2 à 2000 points (nœuds), il peut représenter les trois types d'éléments graphiques suivants (open polyline, closed polyline, area). Pour les ways avec plus de 2000 nœuds, il peut être géré par division.

2.2.1 Open polyline

Termine un segment de ligne non fermé. Souvent utilisé pour représenter des routes, des rivières, des voies ferrées, etc.

2.2.2 Closed polyline

Terminer les lignes connectées. Par exemple, il peut représenter un véritable métro Circle Line.

2.2.3 Area

zone fermée. Habituellement, landuse=* est utilisé pour désigner des zones, etc.

2.3 La relation (relation) :

Une relation peut être composée d'une série de nœuds, de ways ou d'autres relations, et les relations mutuelles sont définies par des rôles. Un élément peut être utilisé plusieurs fois dans des relations, et une relation peut contenir d'autres relations.

2.4 Le tag :

Les tags ne sont pas les éléments de base de la carte, mais chaque élément enregistre des informations de données via des tags. Les données sont enregistrées par «key» et «value». Par exemple, les routes résidentielles peuvent être définies avec Highway(k)=residential(v) ; pendant ce temps, des informations supplémentaires peuvent être ajoutées à l'aide d'espaces de noms supplémentaires, tels que :

maxspeed=100 - La vitesse limite est de 100 km/h

Pour des catégories de tag spécifiques, veuillez vous référer à :
https://wiki.openstreetmap.org/wiki/FR

3 Api Overpass

API Overpass permet d'interroger la base de données OSM.

Pour récupérer les données OpenStreetMap, il faut envoyer une requête URL à un serveur distant qui renvoi un fichier xml en réponse. Pour lancer cette requête, nous avons besoin des éléments suivants : Serveur[option1] [option2][...] ; (requête) ;

3.1 La fonction getAdresseCity(String city) :

Premièrement, avec la fonction getAdresseCity(String city) on a commencé par récupérer les informations dont les coordonnées géographiques d'une ville a l'aide de la requête :

```
n = new URL"https://nominatim.openstreetmap.org//search?q="+Nom-ville+"&format=json&limit=1";
```

3.2 La fonction getData(String data) :

Ensuite, à l'aide de la fonction getData() qui prend en entrée le résultat de la première requête on récupère la box des coordonnées géographiques :
nwr (raccourci pour "nodes, way or relations")

```
            String donnees = gp +","+ pp+","+ pg+","+gg;
        return "nwr("+pp +","+ gg+","+ pg+","+gp+");out;";
```

3.3 La fonction getNodesViaOverpass(String query) :

Enfin, avec la fonction getNodesViaOverpass(String query) on interroge l'api overpass et on récupère tout les objets nodes, way et relation :

```
String hostname = "http://overpass-api.de/api/interpreter?";
            URL osm = new URL(hostname);
            HttpURLConnection connection = (HttpURLConnection) osm.openConnection();
            connection.setDoInput(true);
            connection.setDoOutput(true);
            connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");


            DataOutputStream printout = new DataOutputStream(connection.getOutputStream());
            printout.writeBytes("data=" + URLEncoder.encode(query, "utf-8"));
            printout.flush();
            printout.close();

            InputStream response = connection.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(response);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
```

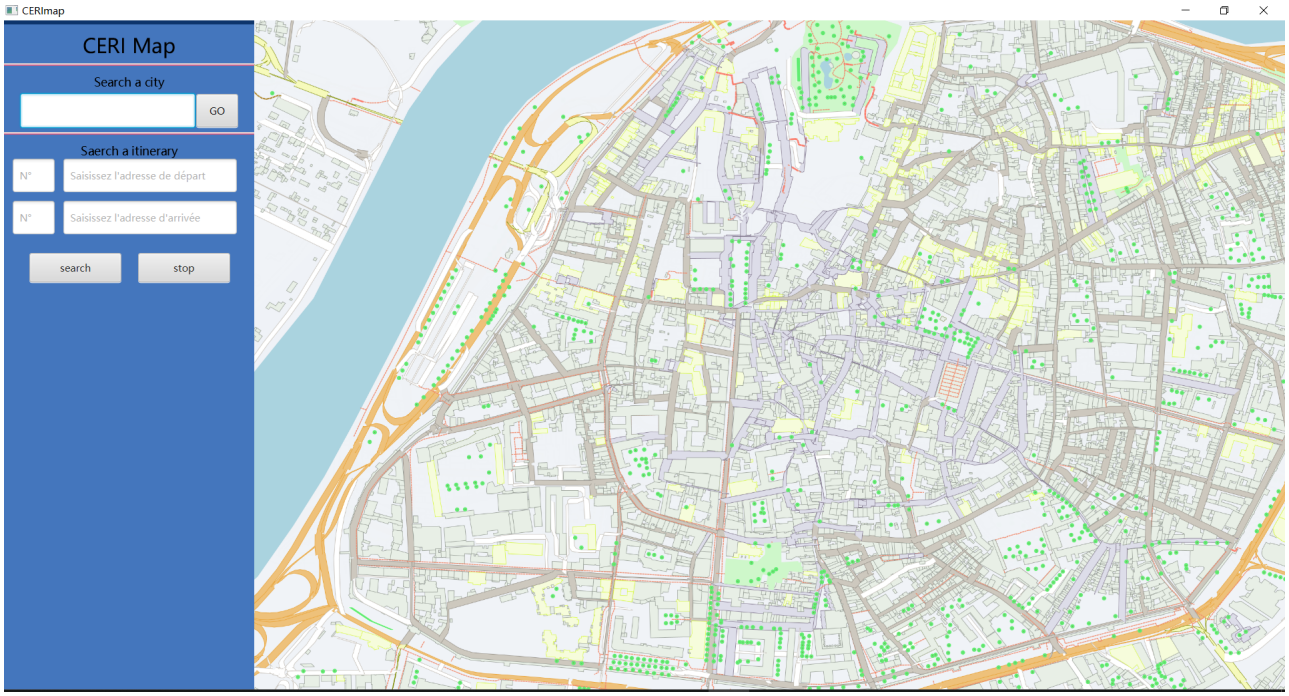
Et on écrit les résultats dans un fichier :

```
// ecrire le résultat dans le fichier
          String inputLine;
             String donnees="";
             StringBuffer content = new StringBuffer();
                   while ((inputLine =  bufferedReader.readLine()) != null) {
                     content.append(inputLine);
                     donnees += inputLine + "\n";
                   }

             bufferedReader.close();
             Files.write(f.toPath(), donnees.getBytes());
```

4 Interface graphique

Notre interface graphique est développé avec JavaFx.



5 Description des travaux

5.1 Description du rendu

5.1.1 Détails relatifs au groupe

Nous sommes une équipe projet composée de deux étudiante d'informatique de licence 3 dont les membres sont : Yingqi LUO et Sara Oumohand. Nous nous sommes rencontrés tous les deux à ce projet et ne nous connaissions pas avant. Nous tous commit les codes sur le projet « CERI UAPV - projet de prog. G05 » sur GitLab.

5.1.2 les branches Git à consulter

Nous avons les branhces suivant : .main par defaut ;
.Yingqi est créee depuis la branche .main (ne utilise plus) ;
.AffichageMap est créee depuis .Yingqi - les codes de ce branche fonctionnent principalement sur l'affichage de la carte ;
.InterfaceGraphique est créee depuis .AffichageMap - les codes rajouté de ce branche travailler principalement sur l'affichage et le fonctionnement de l'interface graphique (faire fonctionner de la menu de recherche et récupérer les infos entrées) ;
.plusCourtChemin est créee depuis .InterfaceGraphique - les codes rajouté de ce branche travailler principalement sur le calcul le plus court chemin du point du départ au point d'arrivée (en utilisant l'aligorithme A*) ;
.api est créee depuis .main - les codes viennent de la branche master, servent à télécharger le fichier .xml par api du OSM ;
.master - une branche indépendant de tous les autres branches, créee par Sara Oumohand.

Les code des branches sont tous pull sur la branche .AffichageMap actuellement. (sauf la branche .api(parce qu'ils n'ont pas encore utilisé dans le logiciel)

5.2 Liste des tâches individuel

Tableau des tâches individuel pour S1

ID	Fonctionnalité	Responsable
1.2.1	Saisir un nom de ville √	Yingqi LUO
1.2.2	Rechercher les coordonnées d'une ville avec Overpass API √	Sara OUMOHAND
1.2.3	Créer un cadre autour d'une ville en récupérant les données d'OMS √	Yingqi LUO
1.2.4	Afficher la carte de la ville sélectionnée √	Yingqi LUO
1.3.1	Saisir les informations zone de départ (numéro, rue, etc.) √	Yingqi LUO
1.3.2	Saisir les informations zone d'arrivée (numéro, rue, etc.) √	Yingqi LUO
1.3.3	Complétion automatique du nom de la rue (zone départ et arrivée)	Yingqi LUO
1.3.4	Lancer la recherche d'itinéraire √	Yingqi LUO
1.4.1	Arrêter un itinéraire en cours √	Yingqi LUO
2.1.1	Afficher le contour des objets (voies, bâtiments, parcs, zones aquatiques) √	Yingqi LUO
2.1.2	Coloriser les objets différemment en fonction des voies, bâtiments, parcs, et zones aquatiques) √	Yingqi LUO
2.1.3	Changer la taille des noms des objets en fonction de la taille de l'objet (proportionnellement)	Yingqi LUO
2.2.1	Zoomer avec la molette de la souris	Sara OUMOHAND
2.2.2	Zoomer avec des boutons présents sur l'interface	Sara OUMOHAND
2.3.1	Afficher les points de départ et d'arrivée √	Yingqi LUO
2.3.2	Modifier les points de départ et d'arrivée via une popup activée au clic de la souris.	Yingqi LUO
2.3.3	Visualiser les informations des points de départ et d'arrivée en les survolant	Yingqi LUO
3.1.1	Spécifier le sens de circulation de l'itinéraire. √	Yingqi LUO
4.1.1	Charger un modèle pré-entraîné via speech brain	
4.1.2	Vérifier si un enregistrement appartient à l'un des membres du groupe	
6.1.1	Affichage Responsive √	Yingqi LUO
6.1.2	Empêcher les erreurs sur entrées textuelles	
6.1.3	Contrôles standards d'une fenêtre : minimiser, maximiser, réduire, fermer, déplacer √	Yingqi LUO

5.3 Planification des tâche

5.3.1 Yingqi LUO

J'ai commencé par rechercher des données libre d'accès du OpenStreetMap, puis j'ai récupéré les donnée sur le fichier .xml que j'ai téléchargé en ligne dont j'ai besoin pour dessiner après (export sur le site de OpenStreetMap), puis j'ai créee une classe **node** et une class **way** pour façonner toutes les données nécessaires dans mon graphe.

Après avoir réussit à dessiner tous les **node**s et les **way**s sur le canevas, j'ai passé un peu de temps à essayer de rendre le canevas auto-réglable en fonction de la taille de la fenêtre, j'ai donc ajouté la classe ResizableCanvas (chaque ajustement de fenêtre obtiendra automatiquement la taille de la fenêtre , à ajuster le canevas et redessiner la carte).

Immédiatement après avoir créé et coloré tous les objets demandés pour l'instant (celles qui correspondent à tous les objets qui seront rendus sur la carte : bâtiments, rivières, parcs, arbres, etc.) puis je les ai colorés, j'ai une classe dédiée à la définition des couleurs d'objets individuels - **Colors**.

Ensuite, j'ai ajouté une barre d'action à mon interface graphique pour entrer et obtenir le nom de la ville, le lieu de départ et le lieu d'arrivée via sa méthode correspondante. Et obtenez son ID correspondant (type : long) à partir du fichier .xml via la classe **Itinerary**.

Le type de données obtenu à partir de la zone de saisie de texte de la barre d'opération de l'interface utilisateur graphique se compose du nom de la route ou du numéro de la maison sur la route et du nom de la route . Grâce à la classe **Itinerary**, le fichier .xml sera filtré. Si les données sont composées `du nom de la route`, la valeur correspondant à k="name" contenue dans "**highway**" dans la balise **tag** est directement recherchée et comparée, et le chemin correspondant est obtenu après confirmant l'objet. L'identifiant du **node** en position médiane ; si le type de données est composé `du numéro de la maison sur la route et du nom de la route`, obtenez d'abord l'identifiant du **node** ou du **way** correspondant, puis recherchez le même nom ("v" de "k="name"") dans les **way**s, il y a la liste des **node**s, en calculant la distance entre chaque nœud et le sien, obtient le nœud le plus proche et en récupérant son id.

Après cela, afin de calculer le plus court chemin, j'ai ajouté la classe **NodePath** pour établir une chaîne de **node** plus tard pour obtenir le **node père** du chaque node; et la classe **ShortestPath** pour calculer le chemin le plus court via l'algorithme A*.

5.3.2 Sara Oumohand

Pour la récupération des données via l'api, j'ai commencé par chercher la requête, j'ai d'abord trouvé une requête avec les coordonnées géographiques d'une ville et j'ai commencé les tests via mon navigateur.
Ensuite, j'ai commencer à écrire le code pour interroger l'api avec java.
Enfin, j'ai cherché la requête qui me permet de récupérer les cordonnées géographiques de la ville à partir son nom.

6 Scénarios fonctionnels.

- Chrgement de la ville (.gif)
- Recherche d'itinéraire (.gif)

7 Conclusion.

7.1 Yingqi LUO

7.1.1 Les difficultés rencontrées

- Compéhension des données du OpenStreetMap ;
- Compréhension d'tulisation du JavaFx ;
- Compréhension l'algorithme A* ;
- Affichage de la mer ;
- OPtimisation de l'interface avec la carte ;
  - Vu que j'ai ajouté la barre de recherche d'itinéraire, ca me prend du temp de les optimiser.
- Quantité de travails ;
- Difficulté de communication entre l'équipement ;
- Difficulté de travailler toute seule.

7.1.2 Les incompréhension

Affichage de l'océan !

7.1.3 Les problèmes de groupe

- Nous n'avons pas assez de gens dans le groupe ;
- Difficulté de communication.

7.2 Oumohand Sara

7.2.1 Les difficultés rencontrées

- La compréhension de l'API Overpass : c'est la première fois que je travaille avec une api donc c'était un peu difficile au début de comprendre son mode de fonctionnent.

- Trouver la requête pour récupérer tout les données d'une ville :
  en effet c'est langage compliqué ça ma demander beaucoup de temps pour trouver la bonne requête qui me permet de récupérer le fichier xml.

- Le gros problème c'est que les ressources de l'api son limitées :
  en effet quand on exécute exactement la même requête (à la même adresse IP), ce qui engendre des comportements problématiques du coup le code ne marche pas tout le temps, et pour cela j'ai trouvé une solution qui le vpn ça nous permet de récupérer les données sans problème.

- Et comme préciser précédemment l'une des plus grande difficulté c'est le groupe on a pas assez de personnel de plus vu qu'on est dans le même groupe pour les autres tp on a des difficulté de communication.