Université de Cergy-Pontoise

RAPPORT DE PROJET

pour le projet de Génie Logiciel
Licence d'Informatique deuxième année

sur le sujet

Géolocalisation Par Satellite

rédigé par

Amhiyen Mahfoud, Ottaviano Aurélien, Radolanirina Yaël

encadré par

Liu Tianxiao

Avril 2018

Table des matières

Table des figures

Liste des tableaux

Remerciements

Nous voudrions vivement remercier Monsieur Liu Tianxiao, de nous avoir suivi et de nous avoir apporté son expertise tout au long de ce projet. Nous souhaitons aussi gratifier Madame Dang Ngoc Tuyêt Trâm d'avoir eu l'idée originale pour le sujet, la communauté OpenStreetMap pour nous avoir fourni le fichier d'origine de la carte, et le peuple éthiopien pour avoir découvert le café qui fût une composante majeure dans le développement de ce programme. Enfin, nous sommes reconnaissant envers l'Université de Cergy-Pontoise pour son accueil et les enseignements qu'elle nous distribue.

1 Introduction

1.1 Choix du projet

Ceci est le rapport de projet effectué dans le cadre de l'UE Génie logiciel et Projet au sein de L'université de Cergy-Pontoise. Le principe du projet est très simple : réaliser un indicateur d'itinéraire multimodal exploitant un réseau de transport. Les moyens de transport considérés sont :

— les transports individuels : sans voie : la marche à pied ; avec voie : le vélo, la voiture
— les transports en commun : bus, métro, train, bateau, avion

Nous avons opté pour ce sujet car nous portons un intérêt et une curiosité particuliers aux notion de *Map Framing* et de *Path Finding*. De plus le sujet nous semblait assez vaste pour nous permettre de le modeler à notre image et d'obtenir un résultat très personnel.

1.2 Qu'est ce que Jalan

Du mot "Route" en malais, Jalan est un service capable de proposer un itinéraire optimisé en fonction du point de départ et d'arrivé saisi par l'utilisateur. Il est aussi intelligent car il intègre un système de suggestion d'itinéraires touristiques, économique ou rapide, en fonction de l'itinéraire de base cherché par l'utilisateur.

Fonctionnalités Le programme permet :

— D'extraire des informations à partir d'un fichier carte OpenStreetMap (OSM)
— D'afficher la carte de Singapour
— À l'utilisateur de saisir son point de départ et d'arrivé
— De calculer des itinéraires optimisés à partir de la saisie de l'utilisateur : l'itinéraire le plus court, le moins coûteux ou un itinéraire touristique
— D'afficher l'itinéraire de l'utilisateur sur la carte
— D'afficher le résumé de la recherche de l'utilisateur : son point de départ, son point d'arrivé, les horaires, les moyens de transports utilisés

Outils de développement Le projet a été créé grâce à :

— Eclipse
— GitHub
— Latex
— Swing

Il fallait concevoir et implémenter un plan de transport composé de stations, par lesquelles transitent les transports en commun et de lignes reliant ces stations. Cependant nous avons préféré prendre un format de fichier qui défini chaque élément par un nœud. Nous avons choisi de prendre la carte d'un vrai pays : celle de Singapour, dont les infrastructures et la situation géographique font d'elle un exemple simple et complet de transport multimodal.

Nous nous sommes ainsi investis complètement dans ce projet et, même si le rendu final n'a pu être à la hauteur de nos propres exigences, nous avons réussi à implémenter un l'indicateur d'itinéraire demandé.

Ce dossier technique a donc été réalisé dans le but de présenter notre projet, ses phases de conception, de développement et de correction. Dans une première partie nous allons donc vous présenter les spécifications techniques du projet, ensuite nous vous présenterons la réalisation conceptuelle, puis l'utilisation du programme avec un manuel utilisateur. Enfin nous traiterons de l'organisation et du déroulement du projet, et nous finirons par une conclusion.

Nous espérons que vous prendrez autant de plaisir à lire ce dossier que nous avons eu à réaliser notre projet.

2 Spécification

Nous avons présenté l'objectif du projet dans la section 1. Dans cette section, nous présentons la spécification de notre logiciel réalisé. Ceci correspond principalement au cahier des charges.

2.1 Cahier des charges

Le tableau ci-dessous regroupe les réponses aux questions à se poser pour bien appréhender le projet.

Qui ?	Le groupe GPS (Aurélien Mahfoud Yaël)
Quoi ?	Jalan : un indicateur d'itinéraire multimodal
Ou ?	Dans le Val-D'Oise à l'université de Cergy-Pontoise
Quand ?	Durant le second semestre de la deuxième année de licence MI
Comment ?	Avec notre cours de Génie Logiciel et Projet comme support, les connaissances acquises durant le premier semestre dans le module de POO
Combien ?	Beaucoup d'heures de programmation
Pourquoi ?	Réaliser un indicateur d'itinéraire multimodal basé sur un vrai pays

Table 1 – Récapitulatif du projet

Parmi les nombreuses applications de cartographie, très peu d'entre elles indiquent un itinéraire multimodal, elles se contentent que d'un seul et unique moyen de transport en plus de la marche à pieds. De plus, seules les applications les plus connues proposent un mode d'itinéraire touristique (le mode découvrir les environs sur Google Maps). Notre application a pour but de proposer à la fois un itinéraire multimodal et donner la possibilité à l'utilisateur de choisir entre plusieurs modes : l'itinéraire le plus court, le moins coûteux ou un itinéraire touristique.

Fonctionnalités envisagées L'utilisateur entre les coordonnées GPS de son point de départ et de sa destination, ou l'adresse de départ et d'arrivée. Il peut aussi choisir de partir maintenant ou plus tard, dans la journée par exemple. Après avoir reçu les premiers résultats de recherche, il a ensuite le choix d'affiner cette dernière en spécifiant si il désire seulement le trajet le moins coûteux, le plus court en temps, avec le moins de marche à pieds possible ou il peut aussi décider des moyens de transport qu'il préfère utiliser. Enfin, il peut demander a obtenir un «trajet touristique». Ce dernier donne tous les points d'intérêts culturels se trouvant sur le trajet. Ces contraintes ajoutées ne sont cependant pas combinable. Quand à l'interface graphique, elle intègre les fonctionnalités suivantes :

- Le déplacement de la carte avec la souris
- Le zoom avant et arrière gérer par des boutons (+) et (-) et éventuellement un curseur.
- L'affichage du point de départ et d'arrivée avec des icônes.
- La carte de Singapour à l'état initial (avant d'avoir effectuer une recherche d'itinéraire), la carte va s'actualiser pour prendre en compte le point de départ.
- Un sommaire de l'itinéraire avec les moyens de transports utilisés, chaque stations empruntées et les horaires, le point de départ et d'arrivée.

2.2 Environnement d'utilisation

Nous proposons un programme utilisable sur un ordinateur Linux, Windows ou Mac équipé d'une installation JAVA 1.8. De plus, un minimum de un GO de mémoire disque, et une souris sont à prévoir pour une utilisation performante.

3 Conception de la carte

Après avoir présenté le cahier des charges dans la section 2 nous allons expliquer comment la carte est créée et gérée.

La génération de la carte de Singapour se fait en quatre étapes.

Dans un premier temps, on récupère sur le site OpenStreetMap un fichier d'extension *.osm* que l'on va modifier en un fichier intermédiaire *.cosm*, comparable a un fichier *CSV*.

Ensuite avec ce fichier on génère un fichier *.jal*, qui est un fichier *XML* contenant les nœuds qui composent notre carte.

Après l'avoir créé on va l'épurer en supprimant :

— les nœuds qui n'ont pas d'autres attributs que leurs coordonnées
— les balises de relations simples
— toutes références aux créateurs des nœuds

On rajoute aux sous-nœuds les coordonnées des nœuds auxquels ils font référence. La carte est enfin générée.

Cependant on utilise *singapour.jal* pour créer trois fichiers :

— singapour.rel : contient les relations entre les nœuds.
— singapour.sug : contient les suggestions de recherche, c'est à dire les noms de chaque élément nommé.
— singapore.trm : contient chaque sous-nœuds et pour chacun d'entre eux le nombre de bâtiments touristiques à moins de 200m.

4 Réalisation

Après avoir présenté le cahier des charges dans la section 2, nous présenterons nos réalisations. Cela correspond principalement au fonctionnement de notre code.

4.1 Le Moteur

Le moteur se compose principalement des classes de traitement, c'est à dire la génération et la lecture des fichiers, ainsi que la recherche et la suggestion de trajet. Pour la génération des données, nous avons créé une classe OSMCompressor, chargée de passer le fichier OSM (XML) volumineux en un CSV, où l'on confond les anciennes balises-filles et attributs, pour un traitement unifié des données. Nous avons ensuite ajouté une classe JALBuilder qui génère à partir de ces données compressées un XML dont les balises correspondent aux besoins de notre projet (balises de longitudes/lattitudes, balises indiquant le caractère touristique d'un lieu...). JALBuilder est également chargé de créer les trois fichiers CSV évoqués dans la partie Conception de la carte.

Quant au traitement des données, nous avons créé une classe JALDocument permettant un parcours optimisé de notre fichier JAL, grâce à l'API Cursor de StAX. Cette classe peut extraire les données d'une balise, de toutes les balises d'un type, et peut les stocker sous forme de collection. Notre classe MapShapping nous permet de créer un fichier SVG contenant les informations graphiques de la carte au format XML, et directement traitable par l'IHM comme une image. Enfin, nous avons une classe CSVDocument nous permettant de lire les différents CSV dont nous nous servons.

Pour le Pathfinding, nous avons employé une variante de l'algorithme de Dijkstra, calculant à la fois le meilleur chemin à partir du départ et de l'arrivée. Le Dijkstra procède comme suit :

— On prend un noeud de départ, et pour chacun de ses voisins :
— On ne prend en compte que les noeuds non visités
— On évalue la distance entre le noeud et son voisin
— On prend le noeud voisin dont la distance est la plus petite
— On marque le Noeud comme visité.

— On répète l'opération en partant du noeud voisin, tant qu'on a pas atteint la destination

4.2 L'interface Graphique

La *GUI* se compose essentiellement de cinq classes. Quatre d'entre elles créent et gèrent chaque "zones" (*Component*) de l'Interface Homme Machine (IHM). La dernière forme l'IHM à proprement parlé, autrement dit elle crée la *frame* et répartie les différents *Component* à l'intérieur à l'aide de *JSplitPane*.

Nous allons donc décrire chacune des quatre classes à l'origine des "zones" de l'IHM.

SearchArea.java Comme l'indique son nom, cette classe crée la zone de recherche.

Elle se compose principalement de deux *JComboBox*, boites déroulante que nous avons rendu éditable grâce à la méthode *setEditable(true)*. On l'associe à un *JTextField*, et afin de récupérer les coordonnées saisies par l'utilisateur ou l'endroit sélectionné par l'utilisateur dans la *JComboBox*[1] on utilise les méthodes [...].*getSelectedItem().toString()*.

Enfin elle contient une classe nommée *splitSearchData()* qui revoie sous la forme d'un *String* les points de départ et d'arrivé à la classe *MapShaping.java* afin de les afficher graphiquement sur la carte.

Options.java Cette classe permet d'accéder à la recherche d'itinéraire en y ajoutant les informations nécessaires manquantes. Elle se compose sous forme d'une *Box* horizontale contenant six *JCheckBox* (une pour chaque moyen de transport gérer), d'un *SpinnerModel* indiquant la date et l'heure de départ, et d'un *JButton* permettant de lancer la recherche.

La sélection d'une des *JCheckBox* entraîne deux événements :

— Le nom du moyen de transport en question est ajouté à un *String* utilisé purement dans l'affichage du résumé de la recherche

— Le nom du moyen de transport est concaténé en minuscule au *String searchInfo* (contenant déjà la date) par la méthode *setSearchinfo()*

L'*ActionListener* nommé *searchListener* associé au *JButton* va donc effectuer ces deux actions. Mais il va aussi régénéré la carte. De plus il va envoyer à la classe *SuggestionSummary.java* les informations de départ, d'arrivée, de véhicules et de date spécifiées sous forme de texte grâce à la méthode [...]*setText()*. Pour finir il ajoute à *searchInfo* les informations manquantes (départ et arrivé).

SuggestionSummary.java Elle est la classe la moins complexe de l'IHM. Composée d'une *JComboBox suggestion* et d'un *JLabel summary*.

La boite *suggestion* contient trois *item* qui permette d'affiner la recherche afin de trouver le chemin le plus rapide (*Quickest*), le moins cher (*Cheapest*) ou le touristique (*Touristic*).

Le champs de texte *summary* quant à lui affiche les informations de recherche reçues de la classe *Options.java*.

Enfin *summary* et *suggestion* sont placés dans un *JSplitPane* vertical, qui est lui même mis dans un *JSplitPane* horizontal contenant aussi la carte en *JSVGCanvas*.

MapArea.java Composée d'un *JSVGCanvas*, nommé *image*, pour la carte en elle même et d'un *AffineTransform* pour les événements de cette dernière, cette classe s'occupe la carte.

Cette classe contient une méthode *DragAndZoom()* qui pour un *MouseAdapter* permet :

— À l'aide d'un *mouseWheelMoved(MouseWheelEvent e)*, et en récupérant la rotation de la molette, de zoomer sur la carte

— De déplacer la carte, en passant par *mouseDragged(MouseEvent e)*, en récupérant les coordonnées du point d'origine du clic de la souris et le delta de déplacement grâce *mouseMoved(MouseEvent e)*

5 Manuel Utilisateur

Dans la section 4 nous avons abordé les réussites de Jalan et nous allons, ici, expliquer comment utiliser l'application de façon optimale.

Lorsque l'utilisateur lance l'application, l'interface de Jalan apparaît. Tout au long de l'utilisation de l'application l'utilisateur peut zoomer et se déplacer sur la carte en utilisant, respectivement, la mollette et un clic de la souris.

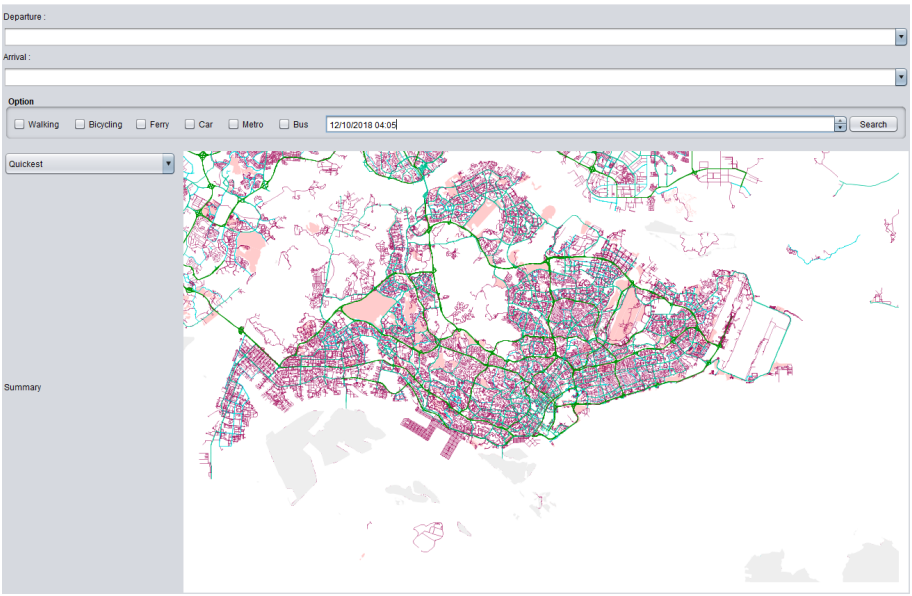

FIGURE 1 – Interface graphique de Jalan

5.1 Avant la recherche

Pour faire une recherche d'itinéraire il doit renter ses coordonnées (longitude et latitude), ou les noms, de son point départ et d'arrivé dans les deux champs prévus à cet effet. En tapant le début d'un lieu l'application lui fait des propositions qu'il peut sélectionner dans les boites déroulantes. Il combine ensuite ces informations aux moyens de transports qu'il souhaite utiliser. Pour finaliser sa recherche l'utilisateur dois préciser l'heure de départ (par défaut l'heure actuelle est sélectionnée) et appuyer sur le bouton "*Search*".

FIGURE 2 – Zone de recherche complétée de Jalan

5.2 Après la recherche

Une fois la recherche menée, l'itinéraire trouvé s'affiche graphiquement sur la carte. L'utilisateur peut lire le résumé de sa recherche sur l'encadré "*Summary*" à gauche de la carte. Celui-ci contient le point de départs et d'arrivé spécifiés, les véhicules utilisés, l'heure et la date de départ ainsi qu'une liste des rues par lesquelles passe le trajet. Enfin si il le désire, il peut se servir de la boite déroulante au-dessus du résumé pour affiner sa recherche, c'est à dire trouver le chemin le plus rapide, le moins cher ou un chemin touristique.

6 Déroulement du projet

Dans cette section, nous décrierons comment nous avons organisé notre travail.

6.1 Répartition des tâches

La répartition du travail au sein de l'équipe fut instinctive dès début.

Aurélien, le chef de projet, se chargea de la conception et implémentation du *pathFinding* ainsi que de la gestion de l'équipe.

Mahfoud, un des gestionnaire de projet, s'occupa de la création et de l'implémentation de la carte, de la gestion des événements liés à celle-ci (*zoom, drag'n'move, refresh*) et de l'écriture d'une partie du rapport.

Yaël, le dernier membre de l'équipe, se consacra à la création de l'IHM graphique, à la gestion des événements liés a celui-ci (*JComboBox, Jtextefield, ...*), à la rédaction du rapport et à la création du diaporama de présentation.

Chaque partie étant interconnectée, nous avons travaillé avec GitHub pour partager nos avancées avec les autres membres de l'équipe. Cela nous offrit aussi un droit de regard sur les travaux de chacun, ce qui facilita fortement la communication, l'entraide et la cohésion au sein de l'équipe.

6.2 Calendrier

Il nous était donné environ deux mois pour mener à bien ce projet. Nous avons donc du faire un calendrier que nous avons suivis du mieux que possible. *In fine*, notre travail se fit comme suit :

- du 25/01 au 05/01 : Documentation et conception des grandes lignes du programme (*UML*, problèmes envisagés, cahier des charges, etc.)
- du 06/02 au 13/02 : Traduction, *parse* du ficher OSM et création du fichier *.jal*
- du 14/02 au 19/02 : Création du squelette de l'IHM graphique (*Jalan.java* et les quatre classes graphiques)
- du 20/02 au 12/03 : Conception de la carte (*.svg*) et création des fichiers *.sug, .trm* et *.rel*
- du 13/03 au 23/03 : Gestion des événements de la carte et l'IHM en général
- du 24/03 au 05/04 : Implémentation du moteur et de l'IA du projet (recherche d'itinéraire et suggestion intelligente de trajet)
- du 06/04 au 09/04 : Finalisation du programme (corrections), création du rapport et du diaporama

Suite à l'hospitalisation de Yaël pendant trois jours, un délai supplémentaire d'une journée nous fut accordé à l'équipe afin de rendre le projet.

7 Conclusion

Pour finir, dans cette section, nous résumons la réalisation du projet et nous présentons également les extensions et améliorations possibles du projet.

7.1 Réussites

D'un point de vue général, ce projet n'est pas une réussite.

D'une part nous avons réussi à créer un indicateur multimodal fonctionnel. Ce dernier permet la sélection d'un point de départ et d'une destination sous la forme de coordonnées ou d'adresses textuelles. Il prend en charge plusieurs moyen de transport qui sont tout à fait combinables.

D'autre part, Jalan ne permet pas à l'utilisateur de spécifier sa recherche en fonction de son besoin. C'est à dire qu'il peut indiquer à l'application si il préfère obtenir le trajet le moins coûteux, le plus court en temps ou un «trajet touristique».

Enfin, comme nous le désirions, nous utilisons la carte d'une vraie ville comme support de notre programme. Carte que nous avons généré sous format *.svg*.

7.2 Améliorations

Malgré les réussites du projet, notre programme n'en est pas moins améliorable.

En premier lieu, nous pourrions améliorer la gestion du zoom. En effet nous pourrions faire en sorte que, plus le zoom est gros, plus il y a d'éléments affichés sur la carte. Cela permettrait de n'avoir que les grands axes au niveau de zoom minimum, et d'ajouter les plus petites rues en augmentant ce dernier.

Après, nous pourrions gérer la portabilité de Jalan. Cela reviendrait à rendre le projet totalement fonctionnel pour chaque ville du monde, en prenant en compte les différence de limitation de vitesse, de prix des transports en commun et des plages horaires de ceux-ci.

Enfin nous pourrions optimiser le code afin de réduire le temps d'affichage de l'IHM et de rendu de la carte.

7.3 Ouverture

En conclusion, après avoir passé autant de temps sur ce projet, nous nous sommes rendu compte des possibilités qui s'offraient à nous vis à vis de Jalan. Et nous nous sommes imaginé jusqu'où le développement de cette application pourrait nous mener. Alors nous pensons que, en plus d'être ce qu'il est aujourd'hui (c'est un dire un projet étudiant), Jalan pourrait devenir un application mobile. Un système de recherche d'itinéraire multimodal *on-line* permettant le téléchargement et l'utilisation *off-line* des cartes de toutes les villes du monde.

Glossaire

Vous trouverez ici une partie des définitions des concepts et éléments Java abordés tout au long de ce rapport.

- *Mapframing* : concept consistant à créer et gérer une carte.
- *Pathfinding* : concept informatique consistant à trouver comment se déplacer dans un environnement entre un point de départ et un point d'arrivée en prenant en compte différentes contraintes.
- *GUI (IHM)* : De *graphical user interface* (interface homme-machine en francais) est un moyen logiciel prévus pour la communication entre un être humain et une machine, dans lequel les objets à manipuler sont dessinés sous forme de pictogrammes à l'écran, de sorte que l'usager peut les utiliser en imitant leur manipulation physique.
- *Component* : Objet ayant une représentation graphique qui peut être affichée sur l'écran et qui peut interagir avec l'utilisateur.
- *Frame* : fenêtre de niveau maximal avec un titre et une bordure.
- *JSplitPane* : Contenant utilisé pour diviser deux (et seulement deux) *Component*s.
- *JComboBox* : *Component* qui contient un champ modifiable et une liste déroulante.
- *JTextField* : *Component* léger qui permet l'édition d'une seule ligne de texte.
- *String* : Chaîne de caractères.
- *JCheckBox* : Une implémentation d'une case à cocher - un élément qui peut être sélectionné ou désélectionné, et qui affiche son état à l'utilisateur.
- *JButton* : *Component* sous forme d'un bouton.
- *SpinnerModel* : Un modèle pour une séquence potentiellement illimitée de valeurs d'objet, dans le cas présent il y a un objet de type date.
- *ActionListener* : Interface pour la réception des événements d'action.
- *JSVGCanvas* : *Component* SVG *Swing* général.
- *MouseAdapter* : Classe adaptateur abstrait pour la réception d'événements de souris.
- *UML* : De *Unified Modeling Language*, est une modélisation graphique normalisée pour représenter la conception d'un système.
- IA : Pour Intelligence Artificielle, est l'ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine.

Références

[1] Oracle. *JComboBox (Java Platform SE 7)*, 2017 (accessed February 04, 2018).