08/06/2014

Rapport de projet

Sujet : *Application de recherche du plus court chemin à l'échelle d'une région*

Réalisé par : Max Etter / Olivier Schweitzer

Responsable : M. You Li

Table des matières

I. Cahier des charges

1. Cadre

Ce projet a été réalisé dans le cadre de l'Unité de Valeur LO43 (« Bases fondamentales de la programmation objet ») enseignée à l'Université de Technologie de Belfort Montbéliard (UTBM).

Il a été réalisé par ETTER Max et SCHWEITZER Olivier, étudiants à l'UTBM en Génie Informatique, respectivement en 1er et 2nd semestres du cycle ingénieur.

Il est effectué durant le semestre de printemps 2014, sur le site de Belfort, et est encadré par le responsable de l'UV M. Jean-Charles Créput, dirigé et évalué par M. You LI.

2. Objectifs

L'objectif de ce projet est de réaliser une application de calcul du plus court chemin entre deux points saisis sur une carte de la région de Belfort et d'en fournir un itinéraire.

L'application devra donc permettre de :

- Visualiser la carte et le réseau routier en renvoyant à l'utilisateur les informations géographiques appropriées (échelle/précision, information sur la zone, coordonnées des points sélectionnés, nom de rues, etc.).
- Zoomer avant/arrière, taille réelle, vue globale.
- Editer et visualiser le système d'unités et la précision. On choisira l'unité Km avec une précision au mètre près, et non pas l'unité « pixel » initiale.
- Sauvegarder/lire le fichier de rues et routes dans ce nouveau système d'unités.
- Calculer le plus court chemin entre deux points saisis et renvoyer les informations appropriées.

Afin d'y parvenir les éléments suivants sont fournis :

- Un algorithme de calcul de plus court chemin (Dijkstra) écrit en Java ainsi qu'un exemple d'utilisation de celui-ci.

- Une carte de la région autour de Belfort (region_belfort_routes_fleuves_habitats.gif) dont l'étendue est donnée par les 2 points de coordonnées géodésiques Lambert II (897990, 2324046) et (971518, 272510). Cette carte défini une image de fond de taille 9807 ´ 6867 qui correspond à une surface d'environ 73 km ´ 51 km, la précision est de 7.5 m par pixel. L'étendue exacte en m est donnée par les deux points Lambert II.

- Les données du réseau routier correspondant à cette région (region_belfort_streets.xml), se présentant sous forme d'un graphe sur lequel doit être appliqué l'algorithme de plus court chemin. Les coordonnées des points dans ce fichier sont données dans l'unité « pixel » de la carte de la région.

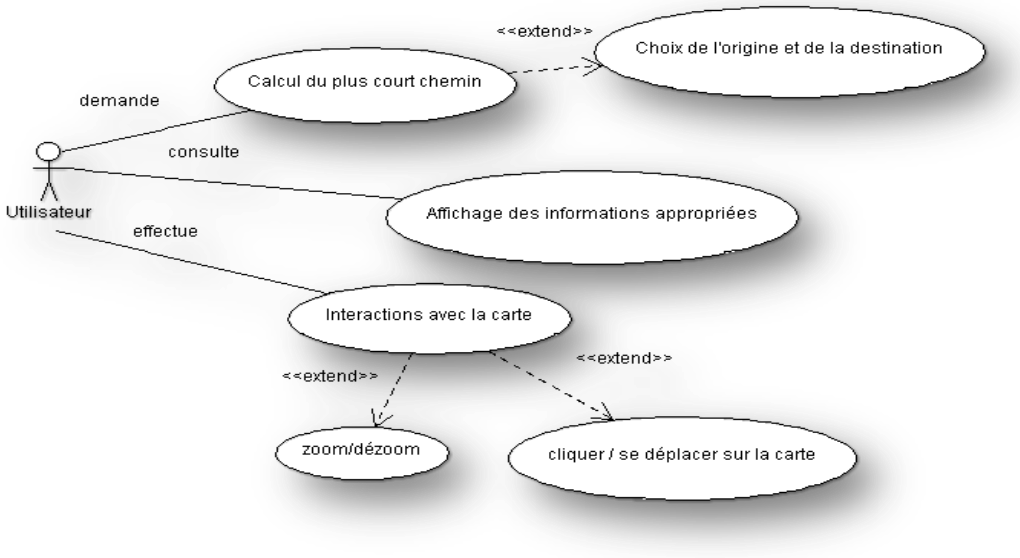
II. Dossier de spécifications

1. Diagramme des cas d'utilisation

Les trois principales fonctionnalités à mettre en œuvre dans cette application sont :

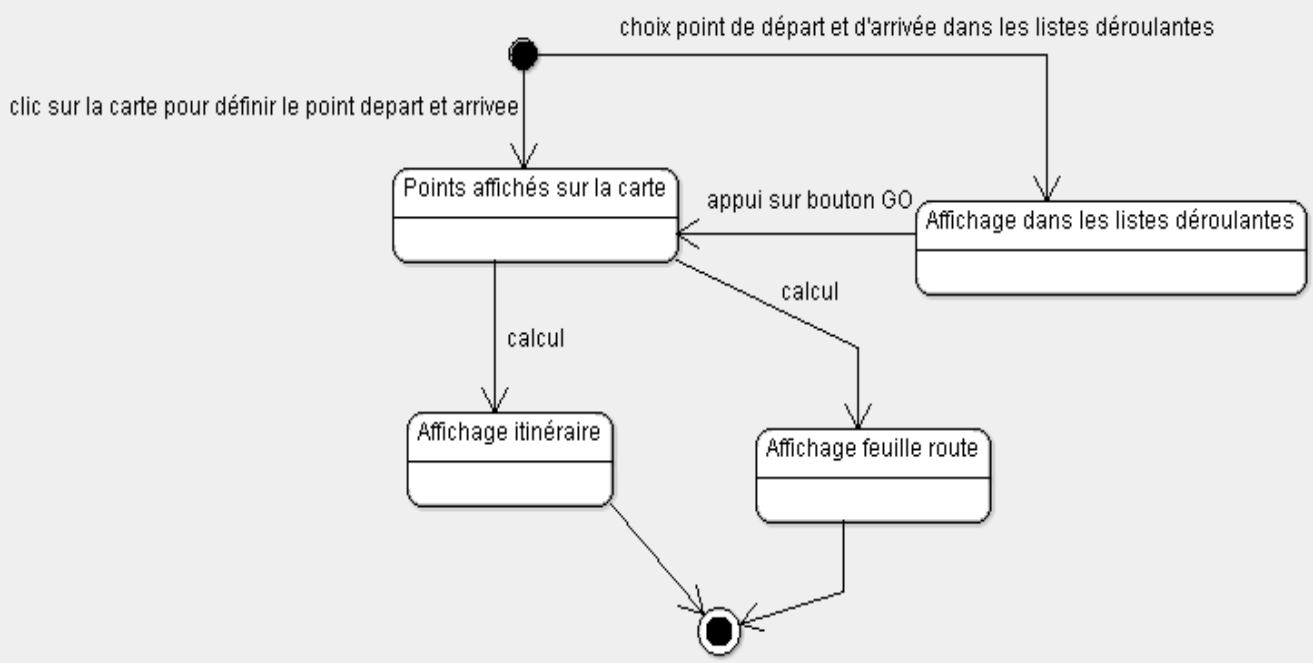
- Le calcul du plus court chemin entre deux points de la carte.
- La possibilité de zoomer et se déplacer sur la carte.
- L'affichage des informations pertinentes concernant l'itinéraire et la feuille de route.

Le diagramme de cas d'utilisation suivant représente ces fonctionnalités :



2. Diagramme états-transitions

Afin de mieux comprendre l'aspect dynamique de la principale fonctionnalité de l'application, à savoir le calcul et l'affichage de l'itinéraire, nous avons réalisé un diagramme états-transitions :



III. Dossier de conception

1. Interface Homme Machine

Sur cette interface on peut distinguer trois zones principales :

- La zone de la carte
- La zone des informations
- La zone de contrôle

Voir Annexe I.

a) La carte

La zone de la carte est la principale zone de l'application, elle est constituée de plusieurs éléments.

Elle contient l'image représentant la carte sur laquelle sera affiché l'itinéraire entre deux points préalablement sélectionnés.

On peut apercevoir dans le coin gauche l'échelle de la carte qui change lorsque l'on zoome ou dézoome.

Lorsque l'on se déplace sur la carte avec la souris, une info-bulle apparaît nous indiquant l'identifiant du point que l'on survole, ses coordonnées ainsi que la route sur laquelle se situe ce point.

Plusieurs fonctionnalités sont disponibles grâce à la souris. Tout d'abord le clic gauche sur un point de la carte permet de définir ce point comme étant le point de départ de l'itinéraire. De la même manière, le clic droit permet de définir le point d'arrivée. En laissant le clic gauche enfoncé, cela permet de se déplacer sur la carte en bougeant la souris. Pour finir la molette permet de zoomer et de dézoomer.

b) Le panneau des informations

La zone de droite est la zone réservée aux informations de la carte et de l'itinéraire.

Dans cette zone on trouve tout d'abord des informations liées à la carte avec :

- Le système d'unité utilisé

- L'identifiant du point courant avec ses coordonnées
- Le pourcentage du zoom

Ensuite on trouve dans cette des informations liées à l'itinéraire c'est-à-dire la feuille de route. Cette feuille indique quelles routes empruntées et sur quelle distance. A chaque changement d'itinéraire cette feuille est mise à jour.

c) Le panneau de contrôle

La partie supérieure de l'interface propose deux zones de contrôle.

La première zone, celle de gauche, permet à l'utilisateur de sélectionner son itinéraire via trois listes déroulantes. Ainsi, il pourra choisir pour le départ et l'arrivée, une ville, une rue et/ou un point.

La seconde zone est quant à elle dédiée à la gestion du zoom. L'utilisateur pour incrémenter ou décrémenter la valeur du zoom grâce aux boutons, déplacer le curseur pour zoomer ou dézoomer ou encore réinitialiser le zoom.

2. Conception de l'application

a) Architecture

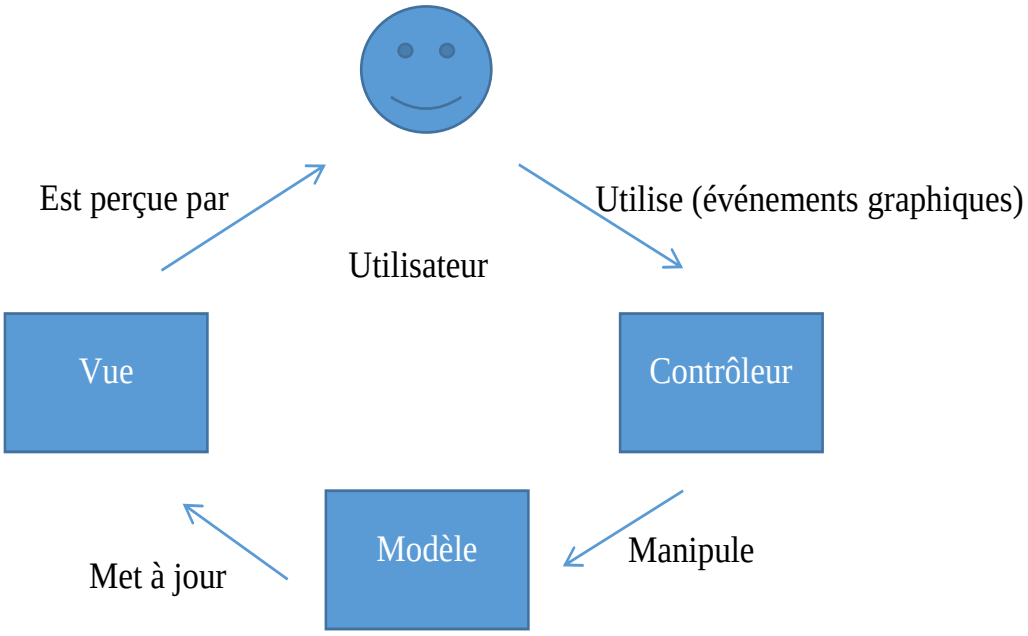
Pour réaliser ce projet, nous avons décidé d'utiliser le patron MVC (Modèle Vue Contrôleur) car c'est une solution très utilisée par les professionnels et que ce projet représente pour nous l'occasion d'en apprendre plus sur cette approche de conception.

MVC est donc une architecture logicielle qui permet de séparer une application en trois éléments :

- Le Modèle qui contient la logique de l'application
- La Vue qui affiche à l'utilisateur des informations sur le modèle
- Le Contrôleur qui agit sur demande de l'utilisateur et effectue les actions nécessaires sur le modèle.

Cette architecture est simple à utiliser en Java car les packages nous permettent de facilement segmenter notre projet.

Voici un schéma qui résume cette architecture telle que nous l'avons utilisée pour notre projet :



b) Détails du modèle de classe

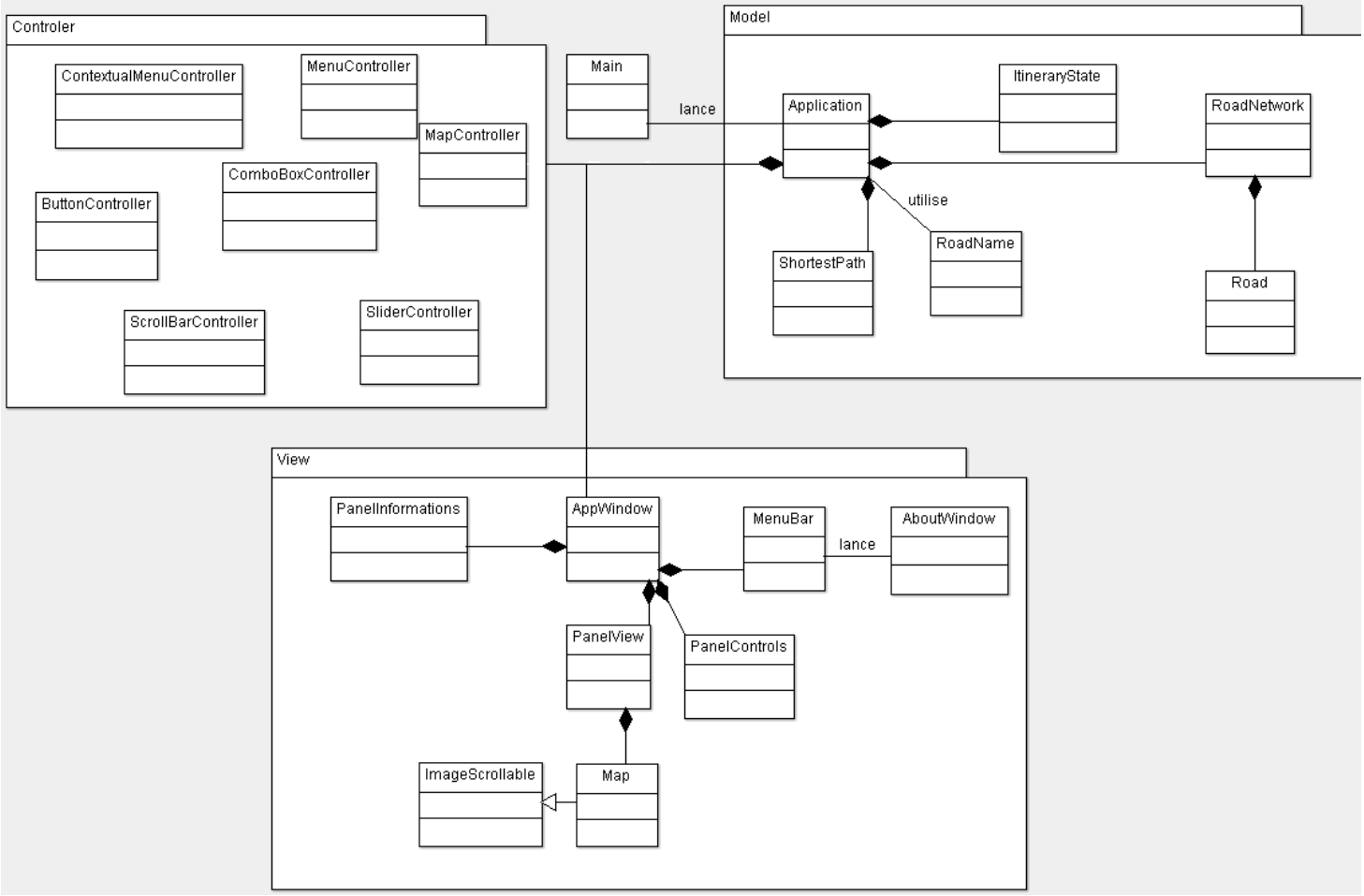


Figure 1 : Diagramme de classe de l'application

La classe Main.java n'appartient à aucun package car elle ne sert qu'à lancer l'application.

- Modèle (package « model ») :
  - *Application* : il s'agit de la classe principale de l'application. Elle gère les différents éléments de l'application, crée la fenêtre principale et ajoute tous les écouteurs (Contrôleur).
  - *ItineraryState* : classe qui permet de stocker le plus court chemin courant.

- *Road* : gère tout ce qui concerne les routes (les points et la direction).

- *RoadName* : classe qui permet de formater les noms des routes (autoroutes, nationales, départementales, européennes).

- *RoadNetwork* : classe qui s'occupe du fichier XML afin d'en tirer les informations utiles concernant le réseau routier lié à la carte.

- *ShortestPath* : Adaptation de la classe *Dijkstra* fournie qui permet de calculer le plus court chemin via des Nœuds et des Arcs (classe *Node* et *Edge*).

- Vue (package « view ») :

  - *AppWindow* : classe qui hérite de la classe *JFrame*. C'est la fenêtre principale de l'application qui contient les trois panneaux (View, Information et Control).

  - *AboutWindow* : classe qui hérite également de la classe *JFrame*. C'est la fenêtre qui correspond à l'item About de la barre de menu. Elle contient des informations sur l'application (qui a réalisé le projet et à quelle date…).

  - *MenuBar* : classe qui hérite de la classe de *JMenuBar*. C'est le menu de l'application qui permet d'exporter la feuille de route, de quitter l'application ou encore de consulter l'aide.

  - *PanelInformations* : classe qui hérite de *JPanel*. Cette classe représente le panneau des informations. Elle est composée d'une zone d'informations contenant en autre la feuille de route.

  - *PanelView* : classe qui hérite de *JScrollPane*. Cette classe représente le panneau vue qui contient la carte. Ce panneau fournit des barres de défilement permettant de parcourir la carte.

  - *PanelControls* : classe qui hérite de *JPanel*. Cette classe contient le panneau de contrôle. Elle contient les listes déroulantes qui permettent de choisir l'itinéraire ainsi que les contrôles concernant le zoom.

- *Map* : classe qui hérite de *ImageScrollable*. Cette classe représente la carte sur laquelle on place les points de l'itinéraire, on affiche cet itinéraire et on offre la possibilité de zoomer.

- *ImageScrollable* : classe qui hérite de la classe *JLabel* et qui implémente *Scrollable*. Permet de se déplacer sur une image et permet d'afficher la carte via un *JLabel*.

- Contrôleur (package « controller ») :

  - ContextualMenuController : implémente l'interface ActionListener. Cet écouteur est lié au menu contextuel qui apparaît lorsque l'on survole un point de la carte.

  - MenuController : implémente l'interface ActionListener. Cet écouteur est lié à la barre de menu de l'application.

  - MapController : implémente les interfaces MouseInputListener et MouseWheelListener qui gèrent les événements liés à la souris. Cette classe est liée à la carte et gère les clics souris et la molette.

  - ComboBoxController : implémente l'interface ActionListener. Cet écouteur rempli les listes déroulantes et les filtres selon les choix effectués.

  - ButtonController : implémente l'interface ActionListener. Cette classe gère les événements liés aux boutons à savoir le bouton de calcul d'itinéraire et les boutons de zoom et de dézoom.

  - SliderController : implémente l'interface ChangeListener. Cet écouteur gère le curseur du zoom.

  - ScrollBarController : implémente l'interface MouseListener. Elle gère le déplacement sur la carte à partir des barres de défilement.

3. Justifications choix de conception

a) Fonctionnalités

La principale fonctionnalité de l'application étant le calcul d'un itinéraire, ce fut la première étape de notre réflexion. Nous nous sommes inspirés des sites de calcul d'itinéraire tel que Google Map. C'est pourquoi nous avons mis en place des listes déroulantes permettant de choisir parmi toutes les villes, les rues et les points. Le lancement s'effectuant en cliquant sur le bouton GO. Pour plus d'ergonomie, nous avons ajouté la possibilité de choisir les points de départ et de d'arrivée en cliquant directement sur la carte. Une fois les deux points choisi, l'itinéraire se calcule automatiquement et s'affiche sur la carte.

La deuxième fonctionnalité majeure est la possibilité de zoomer et de dézoomer sur la carte. Pour cela, nous avons d'abord créé les boutons d'incrémentation et de décrémentation du zoom situé dans le panneau de contrôle. Puis afin d'accélérer le processus de zoom, nous avons implémenté le slider qui permet de zoomer par palier. Finalement, nous avons mis en place la possibilité de zoomer directement sur la carte via la molette de la souris.

Concernant le panneau d'informations, il est représenté par une JList qui est facilement gérable, l'ajout et la suppression des éléments se faisant facilement avec SWING.

b) Carte

Nous avons implémenté la classe Map (celle de la carte) en la faisant hériter de la classe ImageScrollable facilitant ainsi le déplacement sur la carte. ImageScrollable héritant elle-même de la classe JLabel qui permet d'afficher facilement une image à l'intérieur et qui permet de redéfinir la fonction *paintComponent()*. Ainsi on peut aisément afficher l'itinéraire sur la carte.

Afin de gérer le zoom de l'image, nous avons utilisé un objet de type AffineTransform qui est mis à jour à chaque modification du zoom.

IV. Bilan

1. Résultats

L'application semble performante dans l'ensemble, les principales fonctionnalités ont été implémentées. Les méthodes proposées par SWING permettent de réaliser facilement ces fonctionnalités sans les rendre trop lourdes.

2. Critiques

L'application est relativement lourde en raison de la taille de l'image. Cela pourrait se résoudre en chargeant uniquement des parties de la carte, mais cela demanderait plus de temps.

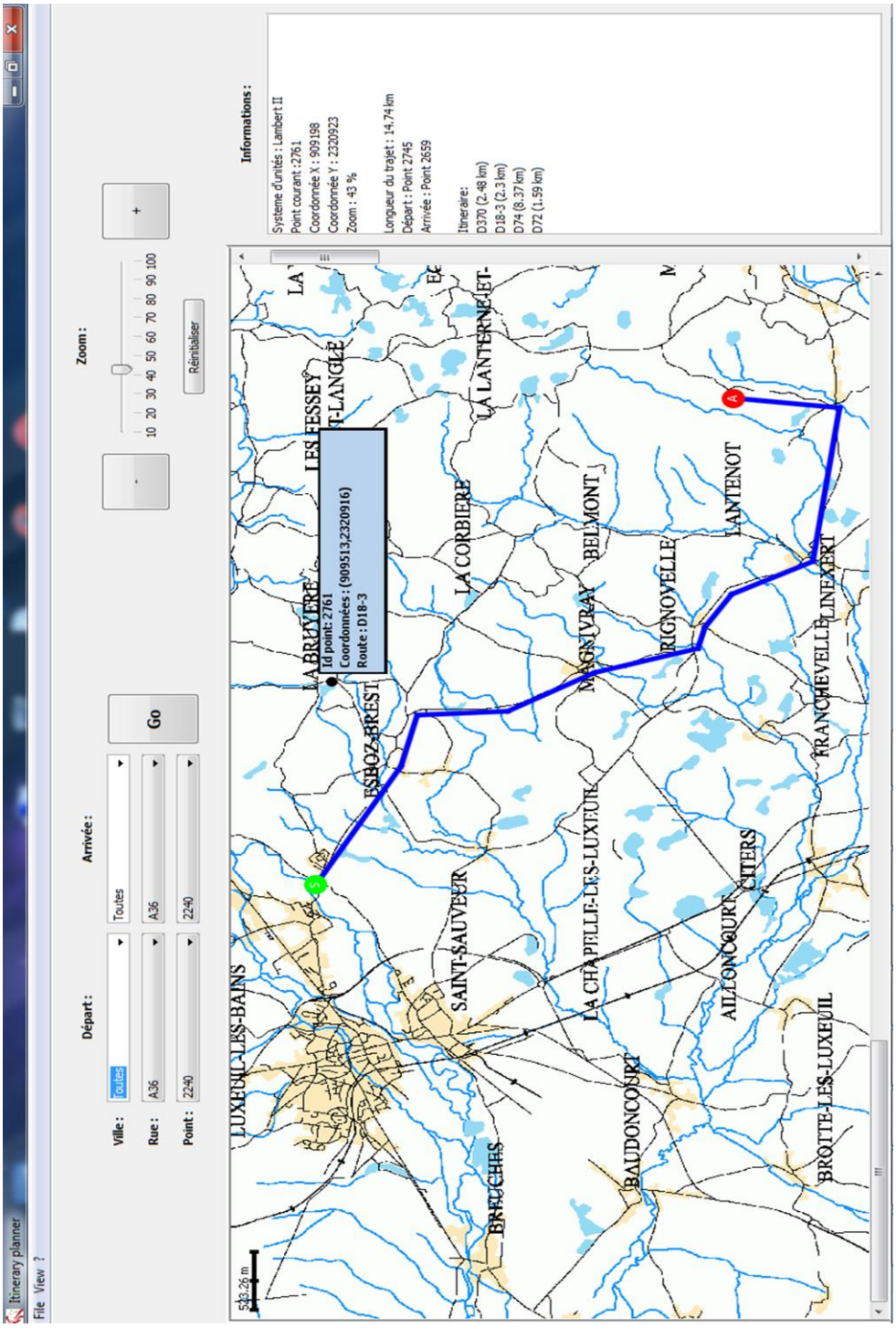
La qualité de l'image rend les zooms très élevés peu lisible, il aurait fallu mettre en place une carte en image vectorielle afin de ne pas perdre d'information en zoomant.

3. Améliorations

Toutes les fonctionnalités n'ont pas été implémentées et d'autres idées pourront à l'avenir être mis en œuvre tel que :

- La possibilité de créer un itinéraire à plusieurs points
- Editer et visualiser le système d'unités et la précision.
- Sauvegarder/lire le fichier de rues et routes dans ce nouveau système d'unités.
- Zone de recherche des noms des villes, des rues.
- Ajouter ou supprimer des routes.
- Extraire une sous zone définies par ses coordonnées Lambert II.

Annexe I


Itinerary planner
File View ?
Départ :
Arrivée :
Ville : Toutes
Rue : A36
Point : 2240
Go
Zoom :
Réinitialiser
Informations :
Systeme d'unités : Lambert II
Point courant : 2761
Coordonnée X : 909198
Coordonnée Y : 2320923
Zoom : 43 %
Longueur du trajet : 14.74 km
Départ : Point 2745
Arrivée : Point 2659
Itinéraire:
D370 (2.48 km)
D18-3 (2.3 km)
D74 (8.37 km)
D72 (1.59 km)
Id point: 2761
Coordonnées : (909513,2320916)
Route : D18-3
LUXEUIL-LES-BAINS
SAINT-SAUVEUR
BREUCHES
LA CHAPELLE-LES-LUXEUIL
BAUDONCOURT
AILLONCOURT
CITERS
BROTTE-LES-LUXEUIL
FRANCHEVELLE
LINEXERT
RIGNOVELLE
MAGNY-VRAY
BELMONT
LA CORBIERE
LA LANTERNE-ET-
LANTENOT
LES FESSEY
T-LANGLE
LA BRUYERE
ESBOZ-BREST