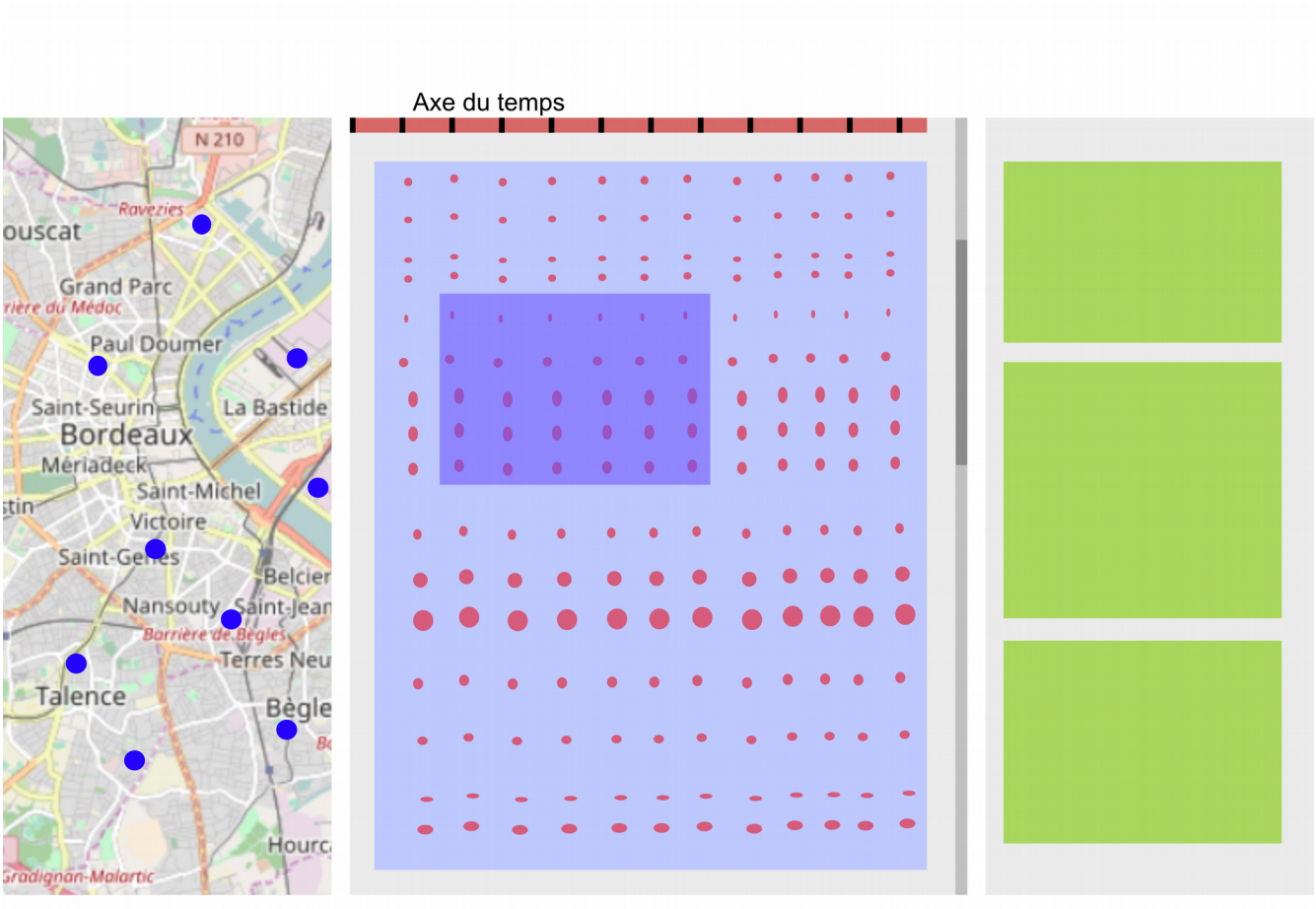
1. Présentation du projet

Le projet est réalisé par quatre étudiants en première année de master informatique dans le cadre d'une UE de projet de programmation. Ce projet vise a nous améliorer dans la création et la gestion d'un projet en équipe. Mais aussi a augmenter nos compétences en programmation et en documentation.

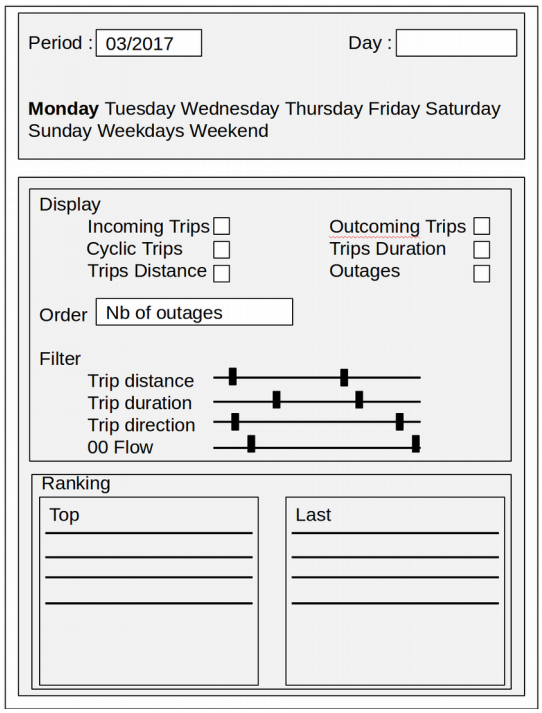
Le projet Visuall BSS (Bike Sharing System) consiste a développer une application qui permet de visualiser un système de vélo en libre service. Celle-ci doit s'appuyer sur un projet existant qui n'est pas accessible mais pour lequel il existe une vidéo (disponible ici) .



L’utilisation générale de cette application Visual BSS est de charger un fichier contenant tous les trajets et stations, puis de les visualiser sur la carte et sur la matrice. L’application permet aussi de trier les données grâce a diverses filtres mais aussi grâce a cette matrice. Nous allons maintenant détailler l’utilité de chaque fenêtre.

Cette application est divisée en trois fenêtres : (voir figure ci-dessus)

- La première permet de naviguer sur une carte afin de voir les stations et les trajets des vélos en libre service. Cette fenêtre est a gauche sur la maquette ci-dessus. Elle est composée de stations (ici les points bleus) et de trajets. Ses trajets sont représentées sous forme de courbe avec un dégradé.
- La seconde, quant a elle, est une matrice qui contient les trajets pour chaque station (en ordonnée) en fonction du temps (en abscisse), ce qui permet de sélectionner une plage horaire ou tous les trajets d'une station par exemple. Cette fenêtre est située au centre de la maquette. Les points rouges correspondent a tous les flux qui partent de cette stations. On peut aussi voir une zone de sélection en bleu qui sélectionne 5 stations dans une certaine plage horaire. Cette action va permettre de visualiser les trajets de ses stations sur la vue de gauche.
- La troisième fenêtre met a disposition plusieurs filtres. Ci-dessous, nous pouvons voir l’interface des filtres. La première partie permet de sélectionner un ou plusieurs jours, la deuxième permet de filtrer les trajets (« trips ») en fonctions de leur distance, leur durée et leur forme.



Nous allons maintenant décrire chaque fonctionnalité a implémenter / a faire durant le projet :

Gestion du fichier :

- Charger un fichier : les fichiers chargés sont les fichiers CSV (fichiers utilisés par exemple sur Microsoft Excel). Ses fichiers contiennent toutes les données des trajets et des stations.

tripduration	starttime	stoptime	start station id	start station name	start station latitude	start station longitude
634	2013-07-01 00:00:00	2013-07-01 00:10:34	164	E 47 St & 2 Ave	40.75323098	-73.97032517

end station id	end station name	end station latitude	end station longitude	bikeid	usertype	birth year	gender
504	1 Ave & E 15 St	40.73221853	-73.98165557	16950	Customer	\N	0

Voici le type de données contenues dans ce fichier. On peut y trouver toutes les informations utiles.

- Parser le fichier : Nous devons analyser le fichier pour récupérer les données utiles. Les données seront par la suite stockées afin de pouvoir les filtrer et les afficher.
- Ajouter un menu pour charger fichier depuis un panel : Ce menu pourra permettre d'ouvrir le fichier plus facilement, plus « user friendly », avec l'aide d'un menu classique (fichier → ouvrir → sélection du ou des fichiers etc) pluton que de devoir taper une commande sur le terminal ce qui pourrait gêner les utilisateurs.
- Si possible implémenter le parsing pour les fichiers Xml et Json (pour pouvoir gérer d'autres types de fichiers)

Matrice :

- Le rectangle de sélection renvoie les trajets qui se trouvent à l'intérieur :

- Afficher des ronds de différentes tailles sur la matrice en fonction de l'affluence
- Mettre à jour les trajets sélectionnes depuis la matrice vers la carte
- Paralléliser le parsing des trajets
- Ajouter une scrollbar vertical et horizontal
- Mettre une taille minimum pour la vue
- Afficher les directions pour chaque stations en fonction de l'heure
- Implémenter l'interaction du scroll down/up
- Implémenter l'animation

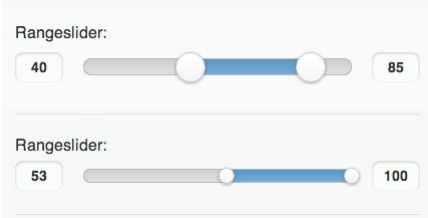
Carte :

- Mettre à jour la station sélectionnée depuis la carte vers la matrice : On peut cliquer sur les stations afin de la sélectionner dans la matrice
- Afficher toutes les stations dans la fenêtre de la carte
- Afficher les trajets avec OpenGL : Pour des soucis de performances, nous devons afficher les trajets avec OpenGL et non pas avec QT (qui les affiche plus lentement)

- Afficher les stations avec OpenGL : (pour des soucis de performances aussi)
- Afficher les trajets avec des courbes
- Afficher les trajets avec un degradé de couleurs
- Afficher la carte de la ville
- Implémenter une fonction pour afficher des cercles avec OpenGL

Filtre :

- Implémenter le filtrage des trajets
- Implémenter un slider à deux valeurs (permet de sélectionner une gamme de valeur et pas seulement une valeur)



Documentation :

- Écrire toute la documentation du code en anglais
- Générer la documentation avec Doxygen
- Faire des diagrammes (séquence, objets)

Robustesse :

- Gérer le cas où le fichier n'est pas valide : Si il n'y a pas de fichier ou un « mauvais »fichier, le logiciel doit être capable de ne pas planter et de prévenir l'utilisateur.
- Gérer le cas où la version d'OpenGL n'est pas valide : La version d'OpenGL doit être supérieur a la 3.3 ; si ce n'est pas le cas, on doit la aussi prévenir l'utilisateur pour qu'il puisse mettre a jour OpenGL
- Tester s'il n y a plus de place dans la RAM
- Tester avec un nombre très grand nombre de trajets et de station (stress test)
- Faire des tests unitaires (oui des tests unitaires) avec des granulés
- Tester le code avec Valgrind (fuite de memoire)

Ce projet est réalisé en C++ 14 avec OpenGL (version supérieur a 3.2). L'IDE utilisé est la dernière version de QT, qui est a l'heure actuelle la version 5.8 stable.

2. Étude de l'existant

Il existe déjà une application qui remplit les mêmes fonctions (développée par des chercheurs).
Cette application est visible sur ses deux vidéos :
- https://drive.google.com/file/d/0B3aeg8yMfRj0MWFmUHZ6ZlR4MzA/view
- https://drive.google.com/file/d/0B3aeg8yMfRj0R3VKQjdtX1htUUU/view

On peut constater que c'est une application web ouverte sur Google Chrome. Son URL (localhost :8080) nous permet de savoir qu'il s'agit d'une démonstration en local. Nous n'avons pas d'informations plus précises (nombres de données, code source etc). Malgres tout, on peut supposer que les performances ne sont pas optimales avec ce type de technologie pour le web (on peut constater quelques lenteurs dans ses vidéos).
Comme expliqué précédemment, notre projet avait pour but de reprendre les fonctionnalité de cette application. Malheureusement, nous n'avions pas toutes les données nécessaires donc nous avons du imaginer les fonctionnalités qui se cachaient derrière certain filtres.

3. Expression des besoins

4. Fonctionnalités implémentées

Lors de ce projet, nous avons implémenté plusieurs fonctionnalités. Afin de rendre le projet le plus complet possible, nous avons définis un ordre de priorité afin d'avoir les fonctionnalités les plus importantes dans un premier temps, puis les fonctionnalités plus « secondaire » après.

Ci-dessous, voici un tableau des fonctionnalités a implémenter, et ce qui a été fait durant le projet.

5. Présentation de l'architecture

i un t