1) Remerciements

Nous tenons a remercier notre client … pour le projet qu'il nous a confié.

Nous tenons aussi a remercier notre tuteur de stage … pour le temps qu'il nous a consacré durant toute la période de ce travail.

2) Glossaire

BSS : Bike Sharing System

C++ : language de programmation

Checkbox : Permet a l'utlisateur de cocher ou decocher une case en cliquant dessus

CSV : Comma-separated values, est un format informatique représentant des données contenues dans un tableau en les séparant par une virgule.

Json : Format de données structurées

OpenGL : librairie de calcul d'image 2D ou 3D

Parsing : analyser un fichier pour récupérer ses données

QtCreator : Environnement de developpement multiplateforme.

Rangeslider : Permet a l'utilisateur de selectionner une gamme (entre une valeur minimum et maximum)

Scrollbar : Composant d'interface graphique, barre de défilement, ascenseur.

URL : localisateur uniforme de ressource, adresse pour acceder aux ressources sur Internet

VBS : Vélos en libre service

XML : Format de données structurées

3) Introduction

Les VBS sont maintenant très utilisé dans les grandes villes en France et dans le monde. Nous en avons un exemple dans la ville de Bordeaux avec les « VCub », pour lesquelles il existe de nombreuses stations.

Ce système permet de naviguer d'un point A a un point B a l'aide d'un vélo mis a disposition (gratuitement ou non).

Très populaire, des logiciels ont été mis au point pour observer le flux de ses vélos, c'est dans ce registre que s'inscrit notre application.

4) 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.
Ce projet a été réalisé en C++* 14 avec OpenGl* 3.3 sur l'IDE Qtcreator*.

a) Objectif :

Le projet Visual BSS* consiste a développer une application qui permet de visualiser un système de vélo en libre service. L'utilisateur pourra observer les flux entre chaque station contenu dans un fichier sur une carte et pourra appliqué divers filtres pour préciser sa recherche.
Celle-ci doit s'appuyer sur un projet existant qui n'est pas accessible mais pour lequel il existe une vidéo (disponible ici) .

b) Application

L'application va être divisé en 3 fenêtres qui vont interagir entre elles (voir fig 1.). Nous allons détailler le fonctionnement de chacune des fenêtres ci-dessous, nous allons aussi parler de la gestion de fichier qui joue un rôle important dans notre projet :
- La première fenêtre permet de naviguer sur une carte. Sur celle-ci, on peut voir et interagir avec les stations. Entre chaque station, on peut apercevoir des courbes (en dégradé de couleur) qui correspondent aux trajets. Ses courbes sont dessinées du cyan (station d'origine) vers le bleu (station de destination) dans le sens horaire, tandis que les arrivées sont dessinés du rouge (stations d'origine) vers le jaune (destination) dans le sens anti-horaire. Les stations et les trajets vont être affiché avec OpenGL* afin d'optimiser l'affichage.
- La seconde fenêtre représente une matrice (voir fig. 2) . Elle permet d'accéder aux flux pour chaque station (en ordonnée) en fonction du temps (en abscisse). Les points rouges correspondent aux flux pour chacune des stations, ses points sont donc plus ou moins gros en fonction de l'affluence. Cette matrice permet aussi de sélectionner les points (grâce a un zone de sélection) afin de mettre a jour les données disponible sur la carte. On peut naviguer sur celle-ci avec l'aide d'un scrollbar* vertical et horizontal.

- La troisième fenêtre met a disposition des filtres (voir fig. 3). Dans la première partie, on peut sélectionner une journée ou une date. Dans la deuxième partie, on va pouvoir accéder a plusieurs checkbox * expliqué ci-dessous :

→ Incoming Trips : Trajets entrants

→ Cyclic Trips : Trajets cycliques

→ Trips Distance : Taille du trajet

→ Outcoming Trips : Trajets sortants

→ Trips Duration : durée du trajet

→ Outages :

→ Order :

L'utilisateur peut aussi utiliser divers «range slider » :

→ Trip distance : Régler la distance minimun et maximum des trajets a afficher.

→ Trip duration : Régler le temps minimum et maximum des trajets a afficher.

→ Trip direction :

→ 00 Flow :

- Pour récupérer les données nécessaire au bon fonctionnement de l'application, nous récupérons les trajets a partir d'un fichier (.csv*). L'utilisateur pourra, a partir d'une ligne de commande ou de façon plus traditionnelle (fichier → ouvrir → sélection du fichier), charger le fichier sur l'application. Ce fichier contient plusieurs champs (voir fig. 4) :

→ tripduration : le temps que dure le trajet.

→ Starttime : heure de départ

→ stoptime : heure d'arrivée

→ start station id : identifiant de la station de départ

→ name : nom de la station de départ

→ start station latidure : Latitude de la station de départ

→ start station longitude : Longitude de la station de départ

→ end station id : identifiant de la station d'arrivée

→ end station name : nom de la station d'arrivée

→ end station latitude : Latitude de la station d'arrivée

→ end station longitude : Longitude de la station d'arrivée

→ bikeid : id du trajet

→ usertype :

→ birth year :

→ gender :

5) Étude de l'existant

Ce projet a pour but de s'inspirer d'une application déjà existante. Elle est visible sur ses eux 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.

6) Fonctionnalités implémentées

Lors de ce projet, nous avons du définir un ordre de priorité afin de pouvoir livrer une application convenable et facilement améliorable si le projet n'arrive pas a sa fin.

Nous avons donc mis au point un ordre de priorités pour définir les fonctionnalités qui nous paraissait primordiale pour le bon fonctionnement de l'application et celle qui sont plus secondaire.

Notre but a été d'avoir un fonctionnement globale de chaque partie de l'application afin de pouvoir itérer a partir de cette base. Consulter le diagramme de Gantt (fig 4.).

Le premier point important était le chargement et le parsing* du fichier. Nous avons choisi, dans un premier temps, de parser les fichiers csv* (Le parsing des fichiers XML* et Json* fonctionnent de la même facon). Ensuite, nous nous sommes accès sur l'implémentation de la vue Carte et Matrice, qui ont toutes les deux, dans un premier temps, implémenté sans l'aide d'Opengl.

Nous avons, pour finir, crée la vue Filtre sur laquelle on a implémenté toute la logique, puis nous avons modifié la vue carte et matrice afin qu'elles utilisent OpenGL, ce qui permet d'avoir de meilleures performances (les donnés chargent plus vite).

Voici un tableau récapitulatif de ce que nous avons pus faire pendant notre projet :

7) Conclusion

Ce projet de groupe nous a permis de comprendre l'importance de l'organisation au sein d'un projet informatique.

De plus, Nous avons appris a respecter les conventions de codage qui permettent d'avoir un code lisible.

Nous avons aussi appris a respecter les attentes du client et des autres personnes étant lié au projet et faire preuve d'initiatives.