Rendu du projet d’affichage SVG avec interaction réseau :

Ladislas WALCAK, Simon DRIEUX, Baptiste JOFFROY, Luka MERCIER

Licence 3 - Ingénierie Informatique - Université d’Orléans 2020

Introduction :

Ce projet a pour but le développement d'un projet en C++ qui réunit deux principales compétences vues en Licence cette année : Réseau et Programmation Orienté Objet.

Le but de cette application est de pouvoir modifier un fichier SVG, situé sur un serveur ayant un socket UDP ouvert. Ainsi, nous proposons une version client qui permet d'entrer et envoyer des valeurs au serveur qui se chargera alors de la modification du SVG et du rafraîchissement de l'image.

Structure du code :

Le projet a été structuré en plusieurs répertoires, ainsi qu'un fichier Makefile adapté contant plusieurs commandes utiles. Nous avons utilisé CMake pour le développement du projet.

La structure du projet est :

- Un dossier *include* contenant tous les fichiers en-têtes (.hpp)
- Un dossier *src* contenant tous les fichiers sources (.cpp)
- Un dossier *obj* créé lors de la compilation contenant tous les fichier objets (.o)
- Un dossier *bin* créé lors de la compilation contenant les fichiers exécutables
- Un dossier *Image Samples* contenant le fichier SVG utilisé par le serveur pour la modification.
- Un fichier *Makefile* contenant les commandes liées au projet

De plus, ce projet contient un nombre de fichiers supplémentaires lié au projet :

- Un fichier *LICENCE.md* contenant la licence open-source.
- Un fichier *README.md* contenant des instructions de lancements
- Un fichier *.gitignore* lié au répertoire Git du projet
- Un fichier *Sujet.pdf* contenant le sujet détaillé du projet

Le dossier *src* est constitué de 4 couples de fichiers en-têtes / sources, ainsi que de deux fichier source supplémentaire pour différencier le serveur et le client.

- Le fichier *main.cpp* est le fichier faisant la liaison entre tous les fichiers lié au serveur. Il créer une instance de *server* et lance l'écoute.
- Les fichiers *server* contiennent la classe *Server*. Cette classe est chargé de lancer un socket UDP écoutant sur un port donné. De plus, cette classe contient en attribut l'instance de *XMLController* afin de lui passer l'information lorsqu'une donnée est reçu.
- Les fichiers *XMLController* contiennent la classe *XMLController*, lié à l'utilisation de la bibliothèque *librsvg* qui permet la modification du XML d'un fichier SVG. De plus, il contient en attribut l'instance de *ServerUI* afin de mettre à jour l'affichage lorsque l'image est modifié.

- Les fichiers *serverUI* contiennent la classe *ServerUI*, lié à l'affichage de l'image avec la librairie *GTK2* ainsi que du rafraîchissement d'image à chaque fois que l'image est modifié.
- Les fichiers *client* contiennent un client UDP sans interface utilisateur, qui est préparé pour envoyer des requêtes à un port spécifié.
- Le fichier *clientUI.cpp* contiennent l'interface IHM étroitement lié à la classe *client* pour faciliter l'envoi de données au serveur (Pas finalisé et donc pas utilisé dans le projet final).

Fonctionnement

Le programme est divisé de manière à s'exécuter "fichier par fichier".

Dans un premier temps, le fichier main va instancier une classe Server qui s'occupera de générer un serveur UDP qui va écouter sur un port spécifique. Il va ensuite ajouter une nouvelle classe en attribut, XMLController pour faire transiter les changements de données. Cette classe XMLController va elle se charger de modifier l'image SVG grâce à l'utilisation de balise XML dites "driven". De plus, cette même classe va également ajouter à ses attributs une instance de la classe ServerUI. Cette instance va être chargé de créer l'affichage graphique de l'image, en créer une fenêtre gtk dans un thread à part.

Améliorations possibles

- Amélioration de l'interface client pour afficher le SVG et voir ses changements afin d'offrir une meilleur expérience utilisateur.
- Coté serveur, une possible gestion des clients avec blocage de réception de données par exemple.
- Meilleur gestion de la mémoire.

Difficultés rencontrées

- 1ere utilisation des libraires open-source non intégrées dans le langage C++, donc difficultées à exécuter les libraires, notamment avec le Makefile et le CMake.
- La documentation des libraires était pauvre, difficile à comprendre, pas assez d'explications sur le fonctionnement des fonctionnalités et des paramètres utilisés.