Using Qt to build a first GUI

IHM – Laboratoire 1

# Introduction

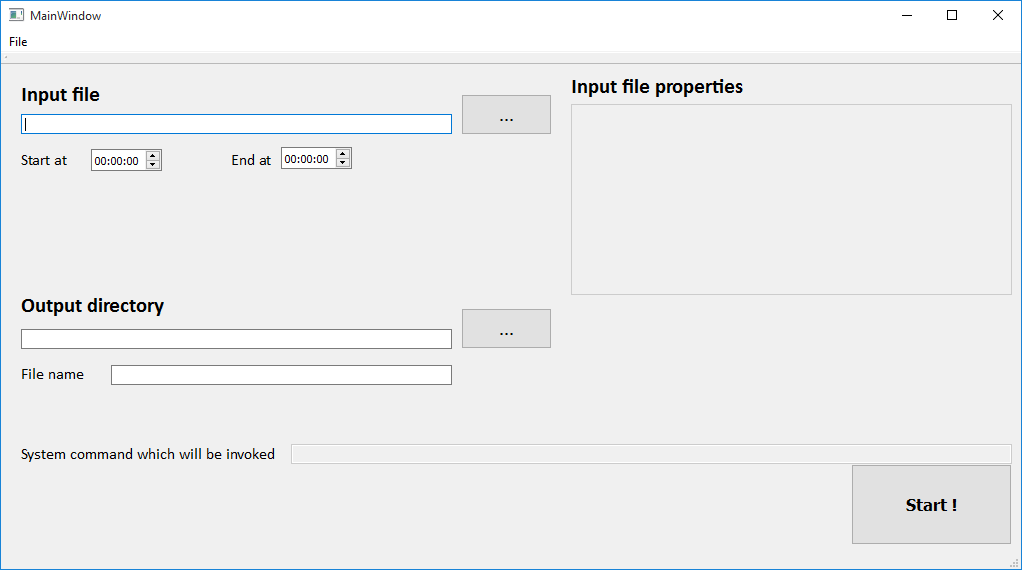
Pour notre premier laboratoire d’IHM, le but est de créer une interface QT pour un programme en ligne de commande. Ce dernier, permet de faire du traitement vidéo (conversions de format, stream audio et vidéo, …). Dans ce laboratoire nous allons nous concentrer sur 2 fonctions fondamentales.

* Obtenir des informations sur le fichier vidéo choisi via « ffprobe », dans le but de les afficher à l’utilisateur.
* Sélectionner uniquement une partie dans une vidéo. Par exemple, si un fichier vidéo dure 1min, il sera possible via notre GUI de prendre seulement les 10 première seconde.

Au niveau de l’interface, nous allons utiliser la librairie C++ QT. Et ce d’ailleurs ici que sera note principale difficulté, car nous n’avons pas eu beaucoup l’occasion d’utiliser cette technologie.

# Interface

L’interface étant le point central de ce laboratoire, nous avons consacré plus de temps que d’habitude à sa confection. Voici le résultat que nous avons obtenu.



A première vue, notre interface est particulière propre, et le but voulu. Car l’utilisateur va tout de suite être attiré par les champs importants de l’application. Qui sont l’ « input » et l’ « output ».

En terme de structure, nous avons fait le choisi d’ordonner les actions à effectuer du haut vers le bas. De manière plus explicite, l’utilisateur va porter son regard naturellement (sens de la lecture) en haut à gauche. Ce qui va l’amener à choisir le fichier en « input ».

A cette étape, l’utilisateur devra cliquer sur le bouton « … » pour ouvrir l’explorateur de fichier pour sélectionné sa vidéo. En faisant cela, le compteur « End at » va prendre la durée maximale possible, et le carré sur la droite va afficher toutes les informations relatives au média en affichant la sortie, en format json, de ffprobe.

A partir de là, l’utilisateur pourra sélectionner quelle durée dans la vidéo l’intéresse, pour cela il lui suffira de modifier les compteurs.

Si on continue à « lire » notre interface, on arrive dans la partie « output ». Et là c’est possible, en clique sur le bouton « … », d’ouvrir à nouveau l’explorateur, mais cette fois-ci pour sélectionner un dossier de sortie. Une fois que l’utilisateur aura fait cela, il pourra choisir le nom donné à son fichier une fois le programme de traitement lancé. L’interface prévoit de prévenir l’utilisateur en cas où le nom du fichier qui va être créé en remplace un autre.

Une fois que ces 2 parties sont entrées, l’interface va afficher la commande qui va être utilisé avec l’exécutable de ffmpeg. Il ne reste plus qu’a appuyé sur le bouton « start » pour lancer le traitement.

# Problèmes

## Lancé un processus via QT

Nous n’arrivions pas à exécuté un processus externe via les commandes de QT. Le problème, c’est parce que nous n’utilisions pas correctement l’object « QProcess », car lors de sa création il faut séparer le chemin de l’exécutable des paramètres. Ce qui nous donne l’appelle suivant.



Après pour récupérer le résultat, une simple méthode permet de retourne le tout en format QString.



## Explorateur de fichier

Au début, nous voulions lancer le processus « explorer.exe » pour afficher un explorateur, mais après plusieurs tentatives, il s’est avéré que ce n’était pas la bonne méthode. En cherchant un peu dans QT, il se trouvait qu’il propose des méthodes toute faites pour lancer un explorateur.

Nous utilisons maintenant ces 2 méthodes pour ouvrir un explorateur. La première permet de retourner le chemin d’un fichier, et la seconde d’un dossier.



Cela vous nous retourner dans les 2 cas, le chemin amenant à ce dossier/fichier.

# Checklist design

1 : Discoverability

Il est possible de voir aisément dans quel état est l’application, grâce au différents pop-ups.

2 : Feedback

Même réponse que pour avant.

3 : Conceptual model

Nous avons utilisé les bons types de champs pour représenter les informations. Et nous n’avons pas jugé utile d’ajouter des pictogrammes.

4 : Affordances

Même réponse qu’avant les champs que nous avons utilisés sont les plus approprié pour chaque fonction.

5 : Signifiers

Etant donné que nous avons peu d’entrées possibles dans notre application, nous n’avons pas eu besoin d’expliquer beaucoup de chose à l’utilisateur.

6 : Mapping

Nous avons placé les champs dans l’ordre de lecture. C’est-à-dire que les premiers champs à remplir sont en haut à gauche.

7 : Constraints

Les types des champs que nous avons choisis limitent au maximum l’utilisateur. Par exemple, le compteur permet seulement de rentré une durée correcte.