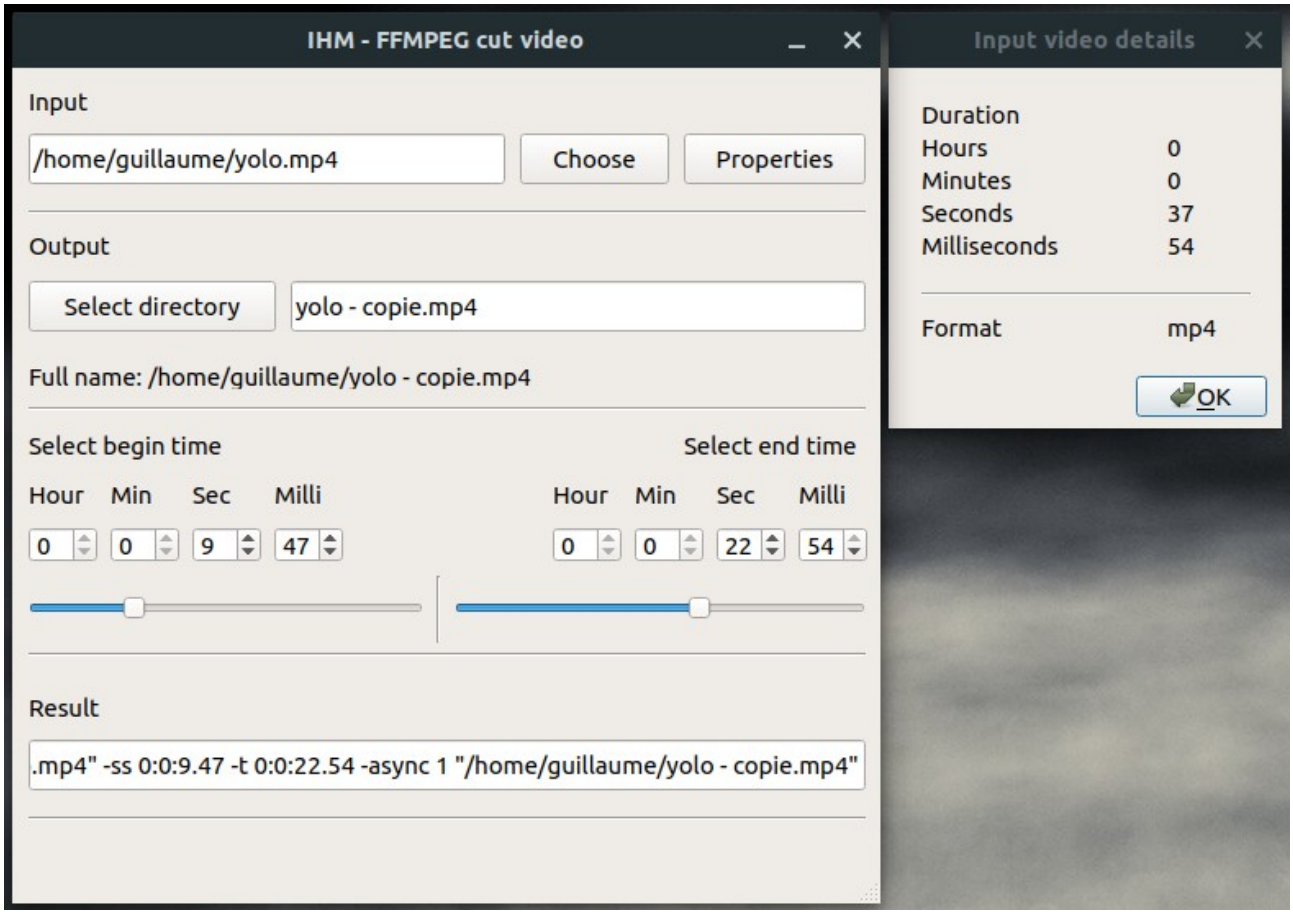
IHM – A GUI for FFMPEG

Apparence de l’application

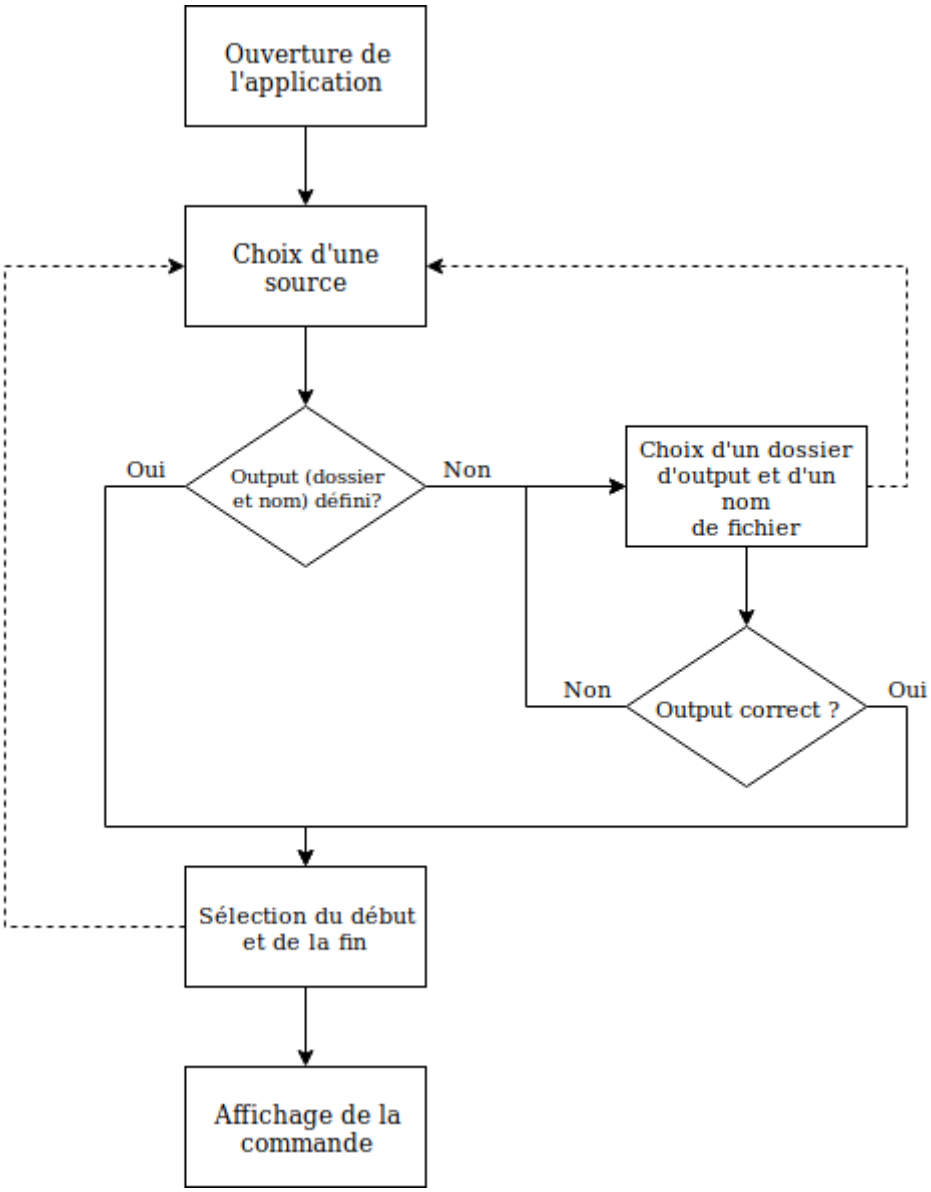


Points intéressants

- L’application oblige la sélection d’une vidéo parmi des types mime autorisés et évite ainsi la sélection d’un fichier non supporté
- Elle oblige la sélection d’un dossier de sortie et d’entrer un nom de fichier et affiche en temps réel le nom complet du fichier de sortie en dessous
- Elle détecte si le fichier de sortie existe déjà et informe l’utilisateur
- Elle détecte si l’utilisateur a entré l’extension dans le nom de sortie, sinon l’application l’ajoutera automatiquement
- Elle détecte si l’utilisateur a entré un chemin comme nom de fichier de sortie et affiche une erreur afin d’éviter de générer un fichier dans un dossier non existant (gestion simple, ne fonctionnera pas sous windows, on test le « / »)
- Utilisation des champs en lecture seule pour l’input et la commande
- Mise à jour du temps lorsqu’il y a une incohérence (début plus tard que la fin et inverse)

Schéma d’activité de l’application

Voici le schéma représentatif du flux principal d’activité de l’application.



Nous avons divisé notre application en quatre étapes importantes :

1. Choix de la source
2. Choix de la sortie (peut générer des erreurs)
3. Choix du temps
4. Affichage de la commande

La validation d’une étape entraîne l’ouverture de l’étape suivante, mais sa non validation bloque toutes celles qui viennent après. Ainsi, une erreur dans le choix de la sortie (fichier existe déjà) entraîne un blocage du choix du temps et de cacher la commande finale.

Erreurs et problèmes

Problèmes d'interface

Nous n'avons pas réussi à rendre la fenêtre des détails dépendante de la principale. Fermer la fenêtre principale n'entraînera pas la fermeture de celle de détails et ne mettra pas fin au processus ce qui n'est pas logique en soit.

Nous n'avons pas réussi à implémenter un double slider comme nous voulions à la base, nous nous sommes donc rabattus sur deux sliders qui semblent relativement logiques.

Erreurs dans le code

Nous avons un problème avec la lecture de l'output de la commande ffprobe ou ffmpeg quelle qu'elle soit. En effet une erreur est toujours présente, nous sommes donc obligés de lire le flux d'erreur et de le parser à la main à l'aide de substring. Nous ne récupérons donc que la durée totale de la vidéo, le format est récupéré en parsant le nom du fichier fourni en entrée.

Erreurs d'implémentation

Nous avons complètement sous-estimé la complexité des interfaces utilisateurs sous Qt et la quantité de code à générer pour implémenter tous les signaux et slots nécessaires au bon fonctionnement de notre application.

Nous aurions du mieux réfléchir à la structure générale de notre application, mais nous avons appris à utiliser Qt pendant le laboratoire ce qui a engendré pas mal de duplication de code et de logique répétée à plusieurs endroits.

Implémentation en étapes

Nous aurions du plus nous concentrer sur l'implémentation des étapes logiques et beaucoup plus s'appuyer sur les pointeurs en C++ qui nous auraient bien facilité la vie et éviter de nombreuses ré-assignations.

Conclusion

Malgré les difficultés rencontrées et le temps qu'il nous a fallut pour comprendre que Qprocess nous donnait de l'information que par le flux d'erreur, le laboratoire était très intéressant à réaliser. La manière dont Qt gère les interfaces et les interactions à l'aide des signaux et des slots est intéressante à mettre en parallèle avec d'autres langages plus haut niveau comme Java ou Javascript, mais la manière de faire à l'aide d'évènements reste similaire.

C'est difficile de faire une interface parfaitement adaptée à la compréhension de chaque utilisateur, voir même impossible, l'oubli d'un élément dans le schéma d'activité peut entraîner beaucoup de refractoring. Notre interface n'est pas parfaite mais nous sommes très satisfaits du résultat, qui nous indique bien quelles sont les prochaines étapes à réaliser pour atteindre notre but.