UNIVERSITÉ DE NANTES

Université de Nantes

Interface Homme-Machine

Totoscope

Logiciel de rotoscopie avec Qt

Étudiants :
Marie Lenogue
Nicolas Brondin

Encadrant :
M. Languénou

1er Mars 2015

Table des matières

1 Introduction

2 Choix d'implémentation

Totoscope est un logiciel de rotoscopie utilisant la librairie Qt. Nous avons donc essayer d'utiliser les fonctionnalités de cette librairie au mieux.

Notre logiciel est composé de plusieurs fenêtres, nous allons tout d'abord décrire les choix opérés pour la fenêtre principale.

2.1 Fenêtre d'ouverture

2.2 Fenêtre principale

Comme décrit dans notre premier rendu, nous avons décidé de rendre les fonctionnalités principales le plus accessible possible.

Notre interface bien que simple et épurée, permet d'effectuer les actions essentielles à l'utilisation du logiciel.

La fenêtre peut être divisée en cinq parties distinctes.

2.2.1 La barre de menu principale

Elle répertorie toutes les actions possibles de l'application. Elle est fractionnée en trois sous-menus :

Fichier
: Ce menu contient des actions telles que créer un nouveau projet, ouvrir un projet déjà exitant, enregistrer ou exporter le travail courant ou encore fermer le projet et le logiciel.

Édition
: Le menu Édition présente les actions fonctionnelles de l'application comme le fait d'annuler la dernière action, ou de la rétablir, l'activation ou non des pelures d'oignon et de la vidéo en arrière plan, et les actions concernant la lecture des dessins déjà effectués.

Aide
: Ce menu quant à lui, ne contient que l'action A Propos qui donne accès aux informations concernant l'application.

2.2.2 La barre de contrôle de la vidéo

2.2.3 La barre latérale de dessin

2.2.4 La zone de dessin

2.2.5 Les miniatures

2.3 Fenêtre d'exportation

2.3.1 Exportation au format images

2.3.2 Exportation eu format vidéo

3 Interactions entre les classes

3.1 Patron de conception

Pour le patron de conception, il était évident d'utiliser un modèle en MVC.

Le plus compliqué dans un patron comme celui-ci est d'en faire une implémentation propre avec une réelle séparation des différents composants, tout en gardant une architecture facilement compréhensible.

3.1.1 L'interface

Pour l'interface, nous avons choisi de découper chaque fenêtre (à l'exception des pop-ups) dans une nouvelle classe. Ce qui permet de pas surcharger la classe de la fenêtre principale, tout en gardant un nombre de classes raisonnable pour notre application.

Nous avons aussi choisi de découper deux grosses parties de la fenêtre principale qui sont la zone de dessin ainsi que les miniatures qui sont des classes à part, héritants de classes graphiques Qt.

3.1.2 Le modèle

Toutes les classes du modèle sont des classes qui héritent de la super-classe QObject car cela permet d'utiliser la technologie des signaux/slots et ainsi rendre cette partie asynchrone.

Pour les intéractions avec le système, nous avons utilisé la classe QProcess afin que ces intéractions soient faciles à mettre en place et à obserer.

3.1.3 Le controleur

Le controleur est la pièce maitresse de ce modèle, c'est réellement lui qui fait le lien entre le modèle et l'interface (qui ne discutent JAMAIS directement ensemble).

Il hérite de QObject pour avoir accès aux signaux/slots et n'effectue que très peu d'opérations, il est principalement pour coordonner les actions de l'utilisateur avec les réactions de l'application.

3.2 Diagramme de classe

4 Conclusion