INF1010

Programmation Orientée-Objet

Travail pratique #6

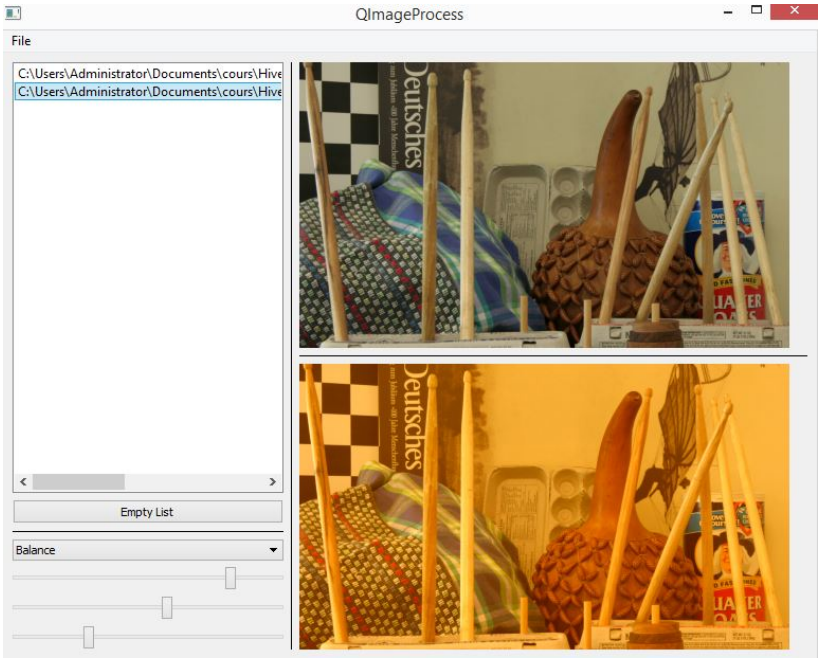
Interfaces Utilisateurs et exceptions

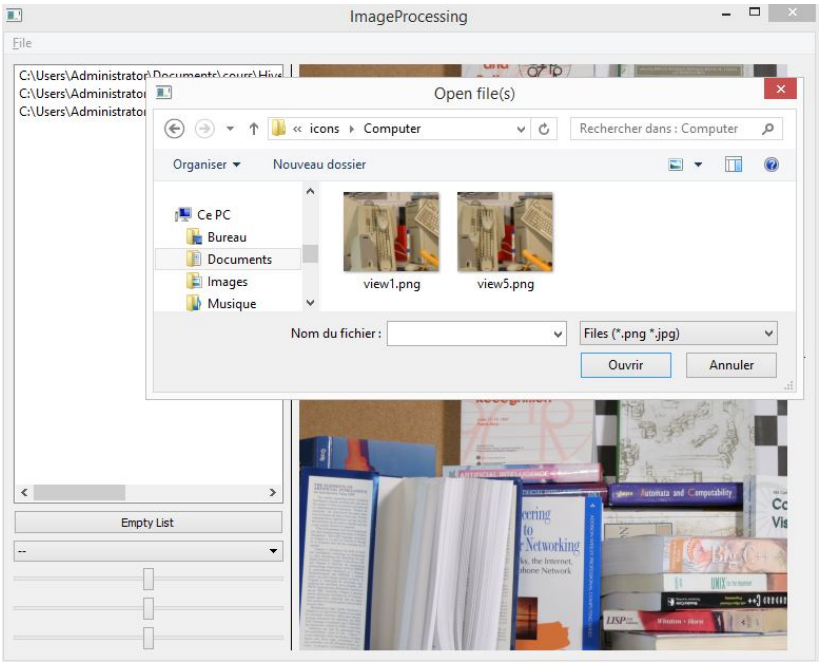
Objectifs :	Permettre à l'étudiant de se familiariser avec les concepts d'interfaces utilisateurs et d'exception en C++.
Remise du travail :	mardi 20 Avril avant, 8h
Références :	Notes de cours sur Moodle.
Documents à remettre :	Fichiers .cpp et .h sous la forme d'une archive .zip.
Directives :	Directives de remise des Travaux pratiques sur Moodle Les en-têtes et les commentaires sont obligatoires. Veuillez suivre le guide de codage

Travail à réaliser

Un de vos collègues a écrit une classe *ImageProcessing* qui permet de faire deux opérations simples sur des images au format RGB : Changer la luminosité et le contraste de l'image (cf. http://tuto.crea-multimedia.org/reglage-de-la-luminosite-du-contraste-et-de-la-saturation-sous-gimp/). Cependant, seuls les informaticiens aiment les programmes en lignes de commande et votre collègue a également développé une interface pour ouvrir des images d'extension .png et les traiter. Ce programme comporte une fenêtre *MainWindow* avec neufs éléments :

1. Un *QMenu* « File » avec deux *QAction* :
    a. Une action pour ouvrir plusieurs images et les ajouter à la liste *QList<QImage>*
    b. Une action pour sortir de l'application
2. Un *QListWidet* qui liste les paths des images chargées. Lorsqu'on clique sur un path, l'image associée est affichée et traitée
3. Deux *QLabel* de taille fixe pour afficher l'image courante et le résultat du traitement
4. Un *QPushButton* « Empty List» pour vider la liste d'images et de paths
5. Une *QComboBox* de 3 éléments permettant d'appeler les opérations sur l'image
    a. « -- » : Aucune opération, l'image traitée est la même que l'image input
    b. « Luminosity » : Changer la luminosité de l'image
    c. « Contrast » : Changer le contraste
6. Un *QSlider*, pour les traitements. Il varie entre -255 et 255





L'interface est presque terminée et **il vous reste à écrire le code pour gérer les exceptions** dans la classe *MainWindow*. En effet, il se peut que le code génère des erreurs (**cela est normal**) et comme il s'agit d'une interface, il ne faut pas que le programme plante. Les exceptions sont définies dans la classe *ImageException* qui vous est également fournie. Vous ne travaillerez donc uniquement dans la classe *MainWindow*. Enfin, des images aux formats png vous sont fournies, vous les utiliserez pour vos tests.

Fichier ImageException

Les fichiers *ImageException.h* et *ImageException.cpp* définissent trois classes d'exception : *ImageLoadingException*, *ImageProcessingException* et *NoImageException* qui dérivent de la classe *ImageBaseException*.

- *ImageLoadingException* est utilisée pour les exceptions liées au chargement des images.
- *ImageProcessingException* est utilisée pour les exceptions liées au traitement des images.
- *NoImageException* est utilisée lorsqu'aucune image n'est chargée.

Classe MainWindow

La classe MainWindow est la fenêtre principale de l'application. Elle hérite de la classe *QMainWindow* et comprend les attributs suivants :

- Deux *QAction *open_action_* et **exit_action_*
- Un objet *process_* de la classe *ImageProcessing*
- Une liste d'images *images_list_* de type *QList<QImage>* contenant les images chargées, une image *image_process_* qui est le résultat du traitement
- Deux QLabel **image_label_* et **image_process_label_* pour afficher les images
- Un *QComboBox *process_combobox_* pour choisir le traitement
- Un *QSlider *slider_* pour faire varier les traitements
- Un *QPushButton *empty_button_* pour vider la liste d'image et de paths
- Un *QListWidget *list_paths_* contenant les paths des images chargées

Seules les méthodes en rouge doivent être modifiées :

- Un constructeur par défaut ne prenant aucun paramètre. Il appelle les méthodes *setUI()*, *setMenu()* et *setConnection()*.
- Une méthode privée *setUI()* qui créer les différents éléments de l'interface.
- Une méthode privée *setMenu()* qui créer les différents éléments du menu.
- Une méthode privée *setConnection()* qui connecte les évènements des éléments de l'interface aux méthodes publique slot. On se refera au header pour les détails.
- Une méthode privée *checkImageIsValid()* qui vérifie si l'image chargée n'est pas corrompue. Si c'est le cas, cette fonction doit envoyer une exception de type *'ImageLoadingException'*. On se refera au header pour les détails.
- Une méthode privée *checkImageIsAlreadyLoaded()* qui vérifie si l'image chargée est déjà présente dans la liste. Si c'est le cas, cette fonction doit envoyer une exception de type ' *'ImageLoadingException* '. On se refera au header pour les détails.
- Une méthode privée *addImages()* qui ajoute des images à la liste. Elle appelle *checkImageIsValid()* et *checkImageIsAlreadyLoaded()*.
- Une méthode publique slot *load()* sans paramètre qui appelle *addImages()* et gère les exceptions de type '*ImageLoadingException'* générées par *checkImageIsValid()* et *checkImageIsAlreadyLoaded()*. On se refera au header pour les détails.

- Une méthode publique slot *empty()* qui est en charge de vider les listes *images_list_* et *list_paths_* et est connectée au bouton *empty_button_*.
- Une méthode publique slot *setImage(int index)* qui affiche l'image à l'index index.
- Une méthode publique slot *setImage(QListWidgetItem *),* surcharge de *setImage(int index).* Elle est connectée au *QListWidget *list_paths_* afin de mettre à jour l'image courante lorsque l'utilisateur clique sur un path dans la liste.
- Une méthode privée *processImageIntensity()*. Elle fait appelle à la méthode *intensity (…)* de la classe *ImageProcessing*. Lors de l'appel, vous devez vérifier si le retour de cette méthode est valide (== 0). Si ce n'est pas le cas, vous devez générer une exception de type *'ImageProcessingException'*. Si l'intensité moyenne de l'image traitée est trop haute, alors *'intensity(...)*' retourne 1. A l'inverse, si l'intensité moyenne est trop basse, la méthode retourne -1. Enfin, la méthode retourne 0 si l'intensité moyenne est correcte.
- Une méthode privée *processImageContrast()*. Elle fait appelle à la méthode *constraste(…)* de la classe *ImageProcessing*. Lors de l'appel, vous devez vérifier si le retour de cette méthode est valide (== 0). Si ce n'est pas le cas, vous devez générer une exception de type *'ImageProcessingException'*. Si l'intensité moyenne de l'image traitée est trop haute, alors *constraste(...)* retourne 1. A l'inverse, si l'intensité moyenne est trop basse, la méthode retourne -1. Enfin, la méthode retourne 0 si l'intensité moyenne est correcte.
- Une méthode privée *processImageAccordingToCombobox()* qui traite l'image courante. Elle est connectée au slider et à la combobox pour être appelée dès que l'utilisateur change de méthode ou faut varier les paramètres. Cette méthode appelle *processImageContrast()* et *processImageContrast()*.
- Une méthode publique slot *processImage()* sans paramètre qui appelle *processImage-AccordingToCombobox()* et gère les exceptions générées par *processImageContrast(), processImageContrast()* et *processImageAccordingToCombobox()*. On se refera au header pour les détails.

Main.cpp

Le programme principal permet de charger l'interface graphique et de tester les différentes méthodes.

Correction

La correction du TP5 se fera sur 20 points. Voici les détails de la correction:

- (10 points) Compilation et exécution exacte des différentes méthodes ;
- (04 points) Utilisation adéquate aux connexions en Qt ;
- (04 points) Utilisation adéquate aux exceptions;
- (01 point) Respect des consignes de l'énoncé ;

- (01 point) Documentation du code et norme de codage ;