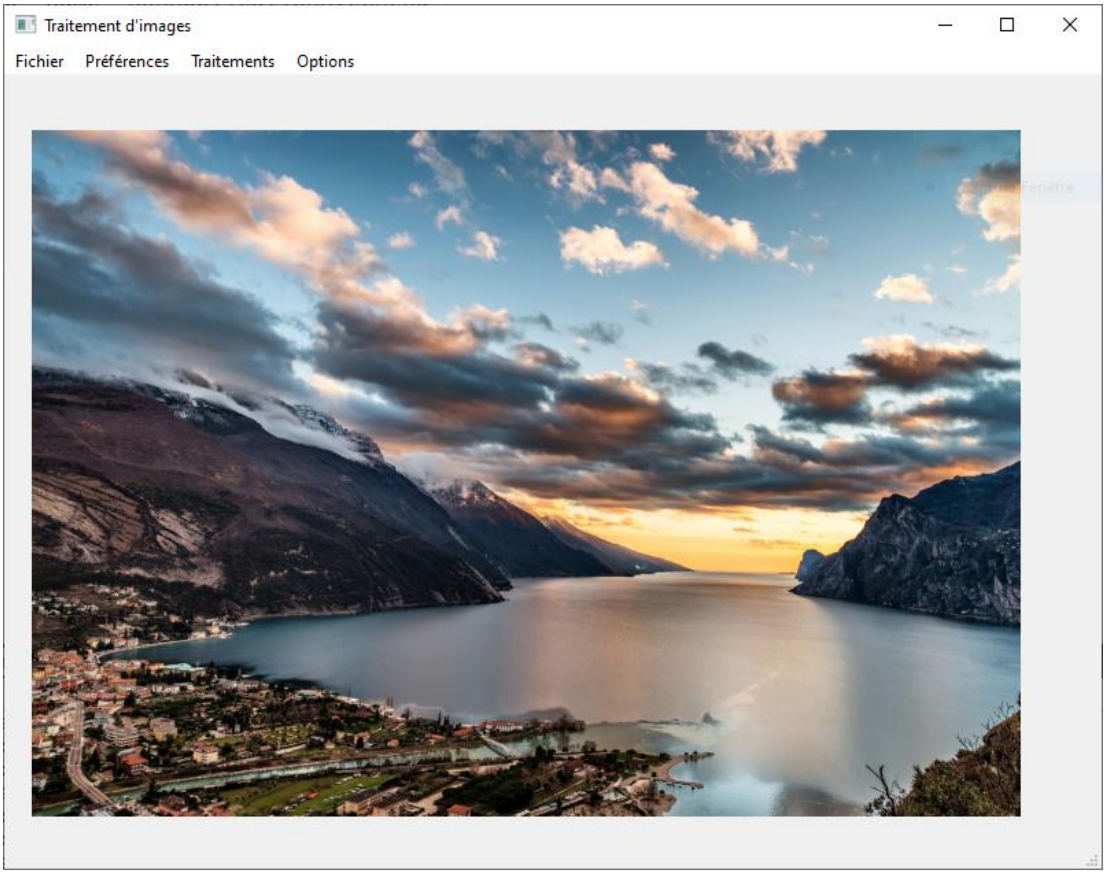
Fenêtre avec menu – Application de traitement d'images

Objectifs : On désire obtenir une application permettant quelques traitements d'image qu'on pourra faire évoluer au fur et à mesure.



Créer un projet « Qt Widget Application » et choisir une fenêtre QMainWindow. Change le nom de la classe gérant votre fenêtre en l'appelant *TraitementImage*.

Phase 1

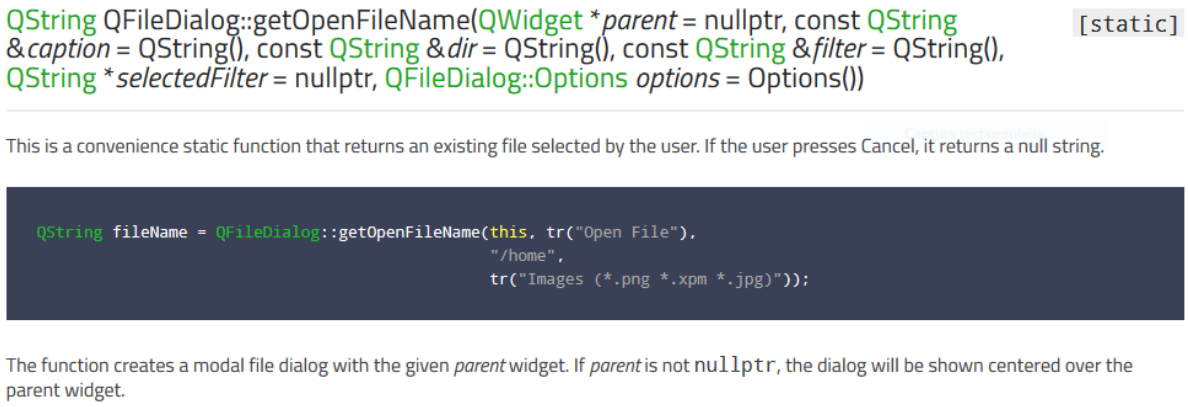
A l'aide de Qt Designer (Design de Qt Creator), ajouter les menus à votre fenêtre.

On considère l'application minimale créée c'est-à-dire une application QMainWindow avec les menus suivants:

- Fichiers
  - Ouvrir
  - Fermer
- Préférences
  - Couleurs
  - Thèmes
- Traitements
  - Convertit en 256 niveaux de gris

Fichier/Ouvrir :

Lors du clic sur le menu *Fichier/Ouvrir* on désire ouvrir le gestionnaire de fichiers de Windows (ou son équivalent linux). Sous Qt, c'est la classe *QFileDialog* qui s'en occupe. Il n'est pas nécessaire de créer un objet de la classe *QFileDialog* pour utiliser certaines de ses fonctions. En effet, les méthodes déclarées « static » peuvent être exécutées sans avoir à créer un objet de cette classe. C'est la cas de la méthode *getOpenFileName*. Voici un extrait de la documentation de Qt sur cette fonction :



QString QFileDialog::getOpenFileName(QWidget *parent = nullptr, const QString &caption = QString(), const QString &dir = QString(), const QString &filter = QString(), QString *selectedFilter = nullptr, QFileDialog::Options options = Options()) [static]

This is a convenience static function that returns an existing file selected by the user. If the user presses Cancel, it returns a null string.

```
QString fileName = QFileDialog::getOpenFileName(this, tr("Open File"),
                                                 "/home",
                                                 tr("Images (*.png *.xpm *.jpg)"));
```

The function creates a modal file dialog with the given *parent* widget. If *parent* is not `nullptr`, the dialog will be shown centered over the parent widget.

Remarquez en haut à droite « [static] », cela indique la possibilité de lancer cette fonction sans avoir à créer un objet de la classe *QFileDialog*

Dans l'exemple (rectangle noir) proposé, on ouvre le gestionnaire de fichiers Windows (ou linux), permettant une navigation dans l'arborescence des disques, le choix d'un fichier en n'affichant que les fichiers portant l'extension .png .xpm .jpg. Une fois le fichier sélectionné, le nom de ce dernier est placé dans *fileName*.

Cette fenêtre est ce qu'on appelle une fenêtre modale : tant qu'on ne ferme pas celle-ci, on ne peut pas « revenir » à la précédente : l'application est donc bloquée par l'exécution de la fenêtre modale.

L'ouverture du gestionnaire de fichiers doit se faire lorsqu'on clique sur le menu *fichier/ouvrir*. Sous Qt, cela est réalisé par le mécanismae de « signaux/slots »

Ajoutez un slot à votre application qui sera lancé au clic sur le menu : à ajouter dans votre classe

```
public slots:
    void slt_OuvrirFichier();
```

Ajoutez la définition du slot : le code de ce slot consiste à exécuter la fonction getOpenFileName()

Par le biais de la fonction *connect*, effectuez la connexion entre le signal « *triggered* » de la classe QAction (le menu) et le slot *slt_OuvrirFichier()*

Ouverture du fichier image sélectionné

Une fois le nom du fichier récupéré par le biais du gestionnaire de fichiers, il faut afficher à proprement parlé l'image dans la fenêtre. Pour cela nous allons utiliser un QLabel.

Ajoutez un QLabel dans la fenêtre (à l'aide de Qt Designer). Agrandir le plus possible ce QLabel de façon à ce qu'il occupe pratiquement toute la fenêtre.

Nommez cet objet QLabel « Image ».

Ce QLabel va être modifié et détourné de sa fonction d'origine. Au lieu d'afficher un texte non modifiable, on va lui faire afficher l'image sélectionnée.

A l'aide de la classe QPixmap et de la méthode du QLabel « *setPixmap()* », affichez l'image.

Le code est évidemment à placer dans le slot précédemment créé. C'est ce qu'on veut obtenir au final lorsqu'on clique sur le menu *fichier/ouvrir*

Evolutions

1. Il y a la possibilité de fermer le gestionnaire de fichiers sans avoir sélectionné un fichier. Or, il faudrait indiquer à l'application, dans ce cas, qu'aucun fichier n'a été sélectionné de façon à ne pas engager un quelqonque traitement d'images sans image affichée ! On pourrait utiliser une variable booléenne dans la classe de notre application pour indiquer qu'une image est bien présente (ou pas) dans la fenêtre.
2. Il serait souhaitable de conserver le nom du fichier en permanence dans la classe de façon à permettre la sauvegarde des modifications

Phase 2

Ajout de la conversion en 256 niveaux de gris :

1. Ajouter un slot qui sera lancé lorsqu'on choisira l'entrée «Convertit en 256 niveaux de gris »
2. Dans ce slot, transformer l'image en niveaux de gris à l'aide de la méthode convertToFormat() de la classe QImage. Vous constaterez alors, que l'objet dont on dispose est un Qpixmap et non un QImage. Il faut donc d'abord récupérer l'image Qpixmap associée au label, la transformer en QImage, opérer la conversion et retransformer cette QImage en Qpixmap et mettre à jour le label

Phase 3

Adaptation de l'image dans le label :

- Faîtes en sorte que l'image soit automatiquement adaptée en taille dans le label (toute l'image doit pouvoir être visible dans la fenêtre)
- Ajouter dans le menu « Traitements », la possibilité d'effectuer une rotation horaire et antihoraire
- Ajouter deux boutons pour zoomer et dézoomer l'image et/ou ajouter un slider (Qslider) (moins à gauche pour dézoomer, plus à droite pour zoomer)
- On pourra utiliser la roulette de la souris pour effectuer le zoom/dézoom