Application Qt de visualisation d'images fixes

Nous allons construire maintenant une application qui ressemble à ceci :

Application Qt de visualisation d'images fixes

1 Description de l'interface à construire

La classe `ImageViewer` sera la classe de la fenêtre principale, héritant de `QMainWindow`. Elle possède une barre de menus comportant le menu *File*, comportant lui-même deux sous-menus : `Open File` et `Quit`.

Elle possède ensuite une zone d'affichage d'image sous forme d'objet `ImageCanvas` ainsi que deux `QLabel` permettant de connaître l'image chargée ainsi que les coordonnées de la souris lorsque celle-ci se déplace sur l'image et les coordonnées couleurs RGB du pixel correspondant. Afin d'organiser les différents composants graphiques, il est nécessaire de les ranger dans un layout `QVBoxLayout`, par exemple.

La classe `ImageCanvas` hérite de `QWidget`. Elle possède une image comme variable d'instance. L'image est chargée à partir d'un fichier dans un object `QImage`. Qt supporte les formats PNG, BMP, XBM, XPM et PNM (PBM, PGM, PPM, ASCII ou raw) ainsi que, si les options de compilation l'ont spécifié : JPEG, PNG et GIF.

Lorsque le sous-menu `OpenFile` est sélectionnée, un nom de fichier sera saisi en utilisant un objet de la classe `QFileDialog`. Pour relier l'action de sélection du sous-menu`OpenFile` à une méthode d'ouverture du fichier `openFile()`, il sera nécessaire de spécifier que cette méthode est un slot de la classe appropriée (à vous de choisir entre `ImageViewer` et `ImageCanvas`). Pour cela, le mot clef `Q_OBJECT` devra être ajouté après l'en-tête de la classe. Modifiez le `Makefile` à l'aide de la commande `qmake` et regardez les modifications. Lancez la compilation et observez les nouveaux fichiers créés.

La fenêtre se redimensionne aux dimensions de l'image chargée et affiche l'image en réécrivant la méthode :

```
void ImageCanvas::paintEvent ( QPaintEvent * event) { ... }
```

Le principe consiste à créer un object de type `QPainter` de père `ImageCanvas` dans lequel l'image est dessinée à l'aide de la méthode `QPainter : :drawImage`.

2 Lecture des pixels - Gestion de la souris

C'est la classe `ImageCanvas` qui va gérer les événements associés aux déplacements de la souris. En appelant la méthode `setMouseTracking`, héritée de la classe `QWidget`, avec une valeur booléenne "TRUE", on active la réception des événements de la souris.

Il reste alors à implémenter la méthode :

```
void ImageCanvas::mouseMoveEvent( QMouseEvent *e) {...}
```

pour traiter ces événements. La classe `QMouseEvent` comporte des méthodes d'accès aux coordonnées de la souris.

On affichera les coordonnées de la souris dans un des `QLabel` de la fenêtre principale (`ImageViewer`) ainsi que les composantes rouge, verte et bleue de l'image si celle-ci existe et si la souris est effectivement positionnée sur l'image.

3 Améliorations

Implémenter les améliorations possibles pour rendre plus conviviale votre application.