Université de la Rochelle - Licence 3 Informatique
Programmation Évènementielle

TP 3 - Une application complète en PyQt

© B. Besserer, R. Péteri

Année universitaire 2018-2019

1 Cahier des charges

Le but du TP est d'écrire une application complète sous PyQt permettant d'afficher graphiquement un histogramme à partir de valeurs contenues dans un fichier texte, que le programme devra ouvrir et lire. Il aura aussi la possibilité de *serialiser* les objets pour les sauvegarder.

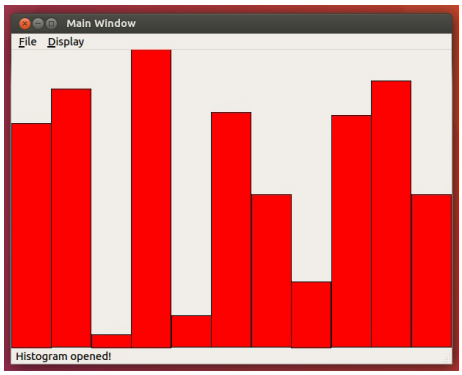

FIGURE 1 – Aspect final de l'application Histogramme

Un code source partiel du projet (`TP3_PyQt_base.py`), est à récupérer sous Moodle. Il contient une classe héritée de `QMainWindows`, qui sera la fenêtre principale de notre application et qui permettra de définir des menus ou une barre d'états (Status bar).
Le code fourni contient aussi la classe `myHisto` qui est dérivée de `QLabel`. Il s'agit du widget qui affichera l'histogramme. La classe disposera de données membres capables de mémoriser les valeurs de l'histogramme : `m_list` qui est une liste d'instance de la classe `Intervalle`. La classe `Intervalle` possède 2 données membres : l'indice du bin de l'histogramme `m_x` et sa valeur `m_amount` (voir Fig.1)
Nous fournissons sous Moodle la base du programme `TP3_PyQt_base.py` qui permet l'affichage d'une fenêtre principale, d'un menu, d'un item de menu `Bye`, et une barre d'état pour afficher des messages. Il contient les classes `Histogramme` et `Intervalles` qui seront utilisées pour stocker l'histogramme (voir figure 1).
L'histogramme comprendra 10 bins (c'est à dire 10 "rectangles").

1. Après avoir testé le programme fourni, ajouter dans le menu `File` de la fenêtre principale de l'application les items : `Open`, `Save`, `Restore`, avec des raccourcis clavier.

2. Ajouter aussi un menu `Display` avec les items `Clear` et `Color` qui serviront dans la suite du TP.
3. Pour chaque item de menu (`QAction`), créer les slots associées (`Open`, `Save`,...) et leurs connexions. La barre d'état (status bar) devra afficher respectivement "*Histogram opened !*", "*Histogram saved !*","*Histogram cleared !*" et "*Histogram restored !*" lors de la sélection des items de menu précédents.
4. Implémenter aussi le slot associé à l'item 'Bye' pour sortir de l'application.

2 Chargement de l'histogramme

Chargement de l'histogramme par fichier

On souhaite écrire la méthode pour charger l'histogramme à partir du fichier disponible sous Moodle `histovalues.dat`. Ce fichier comprend 10 lignes, la première colonne est l'indice du bin et la deuxième colonne sa valeur.
Le fichier doit pouvoir être ouvert en passant par le menu file, et une boite de dialogue standard de sélection de fichier. La classe de la boite de dialogue pour la sélection du fichier est : QFileDialog et filtrera les fichiers sur l'extension `.dat`.
Vous rechercherez les autres informations nécessaires pour l'usage de cette boite de dialogue (n'hésitez pas à regarder le cours). Lorsque vous aurez récupéré le chemin du fichier à ouvrir, il faut bien sûr décoder le fichier pour remplir les données membres de la classe `myHisto` pour l'affichage. On utilisera donc la classe `QFileDialog` pour ouvrir une boite de dialogue permettant de sélectionner le fichier à charger, la classe `QFile` pour lire un fichier, et en enfin `ReadLine()` pour lire dans le fichier ligne par ligne.

Au niveau du tracé, l'histogramme devra s'adapter aux redimensionnements de la fênêtre (abscisse et ordonnée) et sera tracé en rouge (voir figure 1). Pour vous aider, on donne dans le code fourni la méthode `max(self)` qui permet d'obtenir la valeur maximale dans l'histogramme.

☑ 1 (validation du chargement / tracé de l'histogramme et de l'implementation de la barre d'état)

Chargement de l'histogramme par Drag & Drop

On peut activer des fonctionnalités de Drag & Drop pour les widgets. Dans notre cas, en glissant et en déposant le fichier de données de l'histogramme, le résultat doit être similaire à l'ouverture de même fichier par l'intermédiaire du menu file→open.

Pour cela, vous devez surcharger les méthodes :
`dropEvent(self,event)` et `dragEnterEvent(self, event)`
Dans votre fichier, cela ressemblera à :

```
def dragEnterEvent(self, event):
    if event.mimeData().hasUrls():
       event.accept()
    else:
       event.ignore()

def dropEvent(self, event):
    for url in event.mimeData().urls():
       filename= url.toLocalFile()
```

```
# puis lire le fichier et affecter à la variable membre
...
```

Il faudra aussi ajouter dans le constructeur de la classe `MyMainWindow` : `setAcceptDrops(true)`
Implémenter donc l'ouverture par Drag&Drop. Dans un premier temps, avant la lecture du fichier, nous vous conseillons d'afficher dans la barre d'état le chemin récupéré par Drag & Drop de ce fichier . ☑ 2

3 Remise à zéro de l'histogramme et création par valeurs aléatoire

1. Implémenter la fonction pour que l'histogramme soit remis à zero lors de l'appui sur l'item de menu `Clear`.
2. L'histogramme pourra aussi être initialisé lors de l'appui sur la touche 'R' par tirage aléatoire d'une valeur entre 0 et 99 (voir module `random`). Implémentez cette fonctionnalité.

4 Sérialisation binaire des données de l'histogramme

1. En utilisant la fonction `dump` du module `pickle`, créer la fonction de sérialisation permettant de sauvegarder la donnée membre `m_list` de la classe `Histogramme` dans le fichier 'saveHisto.bin' lors du choix de l'item `Save` du menu.
2. Créer la fonction de désérialisation permettant de restaurer les données de l'histogramme contenues dans le fichier 'saveHisto.bin' lors du choix de l'item `Restore` du menu. ☑ 3

5 Changement de couleur du tracé de l'histogramme

On va maintenant sélectionner la couleur du tracé et l'afficher à côté de l'article de menu Display→Color (voir Fig. 2) :

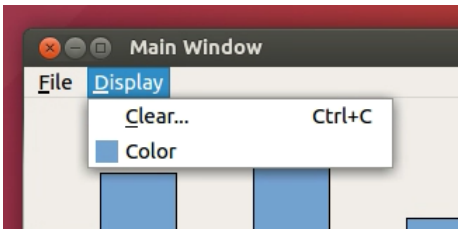

FIGURE 2 – Après changement de couleur du tracé

Une boite de dialogue de sélection de couleur doit s'ouvrir lors d'un clic sur l'article de menu "Color". Une fois la couleur choisie récupérée dans la donnée membre `colorIcon`, on l'utiliser pour peindre la `QIcon` associée la `QAction` de l'article de menu "Color" : `self.colorAct.setIcon(QIcon(self.colorIcon))`

1. Modifiez votre code pour que la couleur de tracé initiale soit affiché dans l'icone associée à l'article de menu "Color".
2. Compléter pour que le tracé et l'affichage de l'histogramme soit effectué dans la couleur choisie, ainsi que l'icône associée à l'article de menu "Color" (Fig. 2). ☑ 4