TD n°5 : Création d'interfaces graphiques
Graphic User Interface (GUI)

MASTER 2 mention Electronique, Energie Electrique, Automatique
Parcours Traitement de l'Information et Instrumentation pour l'Ingénieur (T3I)
Semestre 10
Campus Roannais / LASPI

2018 - 2019

1 Objectifs

L'objectif de ce TD est de :
— "dessiner" une interface graphique à l'aide du logiciel Qt Designer,
— importer cette interface dans Python pour l'afficher,
— réagir aux événements (appui bouton, ...).

2 Prérequis

— fonctions Python,
— légère connaissance sur les objets [1].

Table des matières

1 Objectifs **1**

2 Prérequis **1**

3 Cahier des charges **3**

4 Le dessin **3**
4.1 Créer une fenêtre vide avec Qt Designer 3
4.2 Ajout de widgets (ou contrôles) 5
4.3 Propriété des Widgets 5
4.4 Cas particulier du wdGraph 7
4.5 Test et dernières retouches 8

5 Codage en Python **8**
5.1 Appel de l'interface graphique dans le programme 8
5.2 Gestion des événements 9
5.3 Accès aux contenus d'une fenêtre 10
5.4 Tracé d'une courbe 11

6 Conclusion **11**

7 Bibliographie **11**

Introduction

Les programmes que vous avez réalisés en Python ont une interaction assez limitée avec l'utilisateur : vous tapez des commandes dans la console, saisissez éventuellement des valeurs. Les seuls éléments graphiques que vous avez affichés jusqu'à présent sont des courbes.

Nous allons maintenant regarder comment réaliser une interface graphique (GUI) avec des boutons, zone de texte, ... permettant de contrôler l'application.

En Python, il existe plusieurs boîtes à outils (packages) permettant de réaliser des GUI :

— Tkinter : systématiquement fourni avec Python, fonctionne sur tous les systèmes d'exploitation (MAC OS, Linux, Windows) mais un peu vieux,

— PyQt : plus moderne, dispose d'un outil permettant de "dessiner" les fenêtres. Attention, pour les projets non commerciaux, la licence est de type GNU GPL est gratuite, cependant elle devient payante pour les projets commerciaux !

— wxPython, ...

Vous trouverez des exemples d'utilisation de Tkinter dans [2] et assez facilement sur internet. Nous allons utiliser PyQt car il dispose d'un outil permettant de dessiner des fenêtres qui simplifie fortement la conception. Une description plus approfondie est disponible dans [3] accessible sur http ://unr-ra.scholarvox.com (gratuit à l'université).

Une fois que vous aurez compris les principes de création d'une interface graphique, vous pourrez facilement les transposer pour d'autres outils Python et d'autres langages de programmation (Matlab, Java, C++, ...).

3 Cahier des charges

Nous allons réaliser une application permettant de saisir une fréquence, un niveau de bruit et comportant un bouton valider. Lorsque l'utilisateur cliquera sur valider, la courbe sera affichée.

Nous allons procéder en deux étapes : le dessin de l'interface, puis le codage des actions associées aux boutons en Python.

4 Le dessin

Avec l'explorateur de fichier, allez dans le dossier *anaconda* situé soit à la racine du disque "$C : \$", soit dans le répertoire "$C : \ProgramData\Anaconda3$" [1] puis dans le sous dossier $Library\bin$. Vous y trouverez une application Designer.exe que vous lancerez en double cliquant dessus.

4.1 Créer une fenêtre vide avec Qt Designer

Au démarrage, Designer, vous demande quel type de fenêtre vous voulez créer (figure 1). Vous avez le choix entre :

- *une boîte de dialogue* permettant à l'utilisateur de saisir des informations et de valider (exemple, ouverture de fichier, impression, préférences, ...),
- *une fenêtre* contenant un menu qui sera affichée en permanence et constituera la partie visible de votre application (exemple, fenêtre Word où vous tapez votre compte rendu, fenêtre Excel avec les cellules),
- *widget* qui est l'élément de base (bouton, zone pour saisir du texte, ...).

Choisissez "**Main Window**", une fenêtre apparait (cf figure 2. Cette fenêtre est entourée différentes boîtes à outils :

- *MainWindow - untitled* : c'est la fenêtre que vous êtes en train de concevoir,
- *Boîte de widget* : contient les éléments que l'on peut ajouter dans la fenêtre (bouton, liste, zone pour saisir du texte, ...),
- *Editeur de propriétés* : indique les propriétés de l'élément sélectionné dans la fenêtre (texte affiché, couleur, taille, ...).
- *Inspecteur d'objet* : Liste ce que contient la fenêtre.
- ...

Si une des boîtes à outils n'apparait pas, vous pouvez utiliser le menu Affichage pour l'afficher.

Nous allons maintenant remplir cette fenêtre.

1. attention, il faut taper la localisation dans l'explorateur pour accéder à ce répertoire invisible

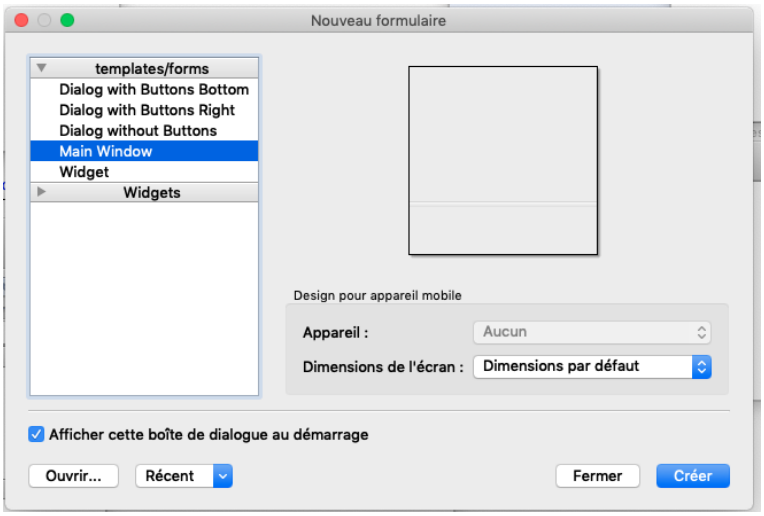

FIGURE 1 – Nouveau formulaire

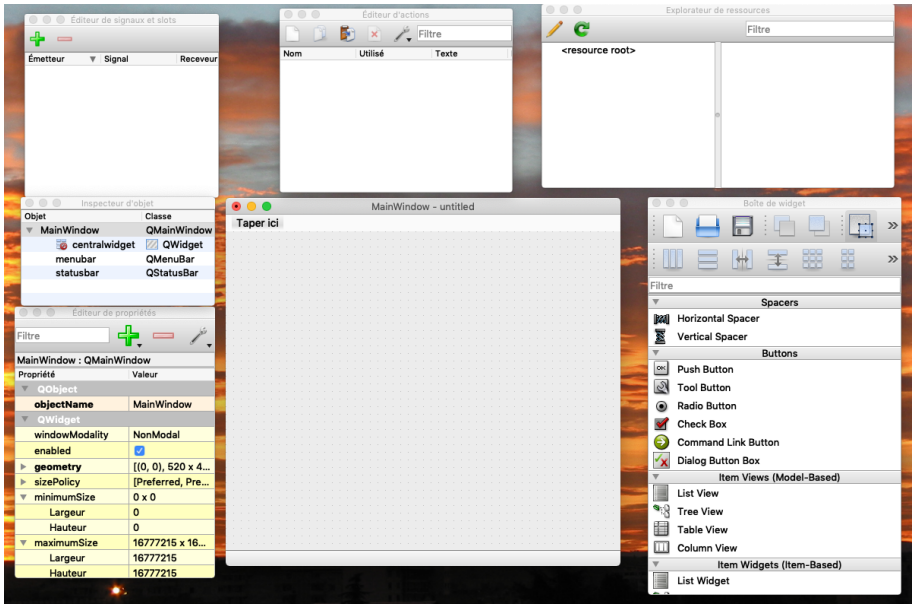

FIGURE 2 – Fenêtre vierge et boîtes à outils associées

4.2 Ajout de widgets (ou contrôles)

Cliquez sur l'élément "**Label**" qui se situe en bas de la boîte de widget puis, en maintenant le bouton gauche de la souris enfoncé, déplacez le sur la fenêtre MainWindow et relâchez le bouton de la souris. Un élément TextLabel apparaît dans cette fenêtre.

Faites de même avec des éléments de type **Line Edit** et **Push Button**. Faites glisser un élément **Widget** à l'endroit où vous voulez afficher la courbe.

Vous devriez avoir une fenêtre qui ressemble à la figure 3. L'élément Widget est visible à l'aide des 8 carrés bleus sur la figure.

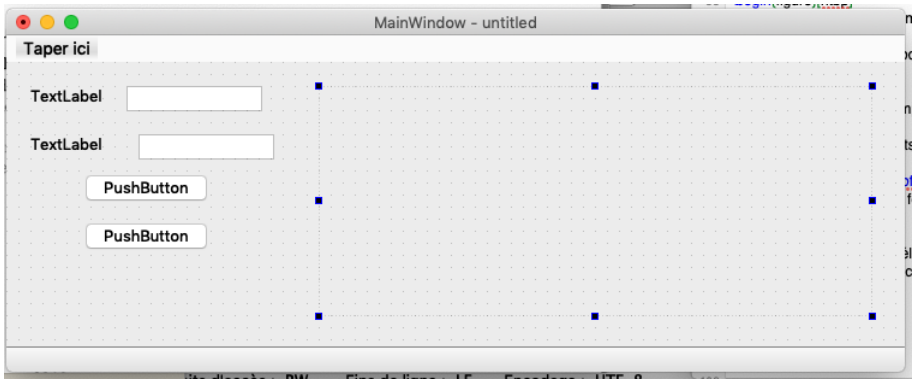

FIGURE 3 – Fenêtre avec des widgets

Nous allons maintenant éditer les propriétés des différents Widgets (contenu, nom, ...).

4.3 Propriété des Widgets

Séléctionner le Widget TextLabel en haut à gauche en cliquant dessus. L'éditeur de propriétés affiche maintenant tous les paramètres qui lui sont associés. Parmi la longue liste, on peut citer :

- *objectName* : nom du widget que l'on utilisera pour le désigner dans le programme Python,
- *text* : texte affiché sur l'interface graphique,
- *windowTitle* (uniquement pour la fenêtre) : titre affiché dans la barre de la fenêtre.

En allant sur chaque contrôle et en vous aidant des repères indiqués sur la figure 4, modifiez les paramètres comme indiqué dans la table 1.

Vous cliquerez dans une zone grise de la fenêtre et indiquerez objectName : Grapheur et windowTitle : Grapheur.

Enregistrer ensuite votre projet dans le répertoire où vous travaillez avec Python sous le nom de Grapheur.ui

n°	objectName	text
1	lblFreq	Fréquence
2	txtFreq	1000
3	lblBruit	Amplitude Bruit
4	txtBruit	0.1
5	btValider	Valider
6	btQuitter	Quitter
7	wdGraph	

TABLE 1 – Changement des propriétés

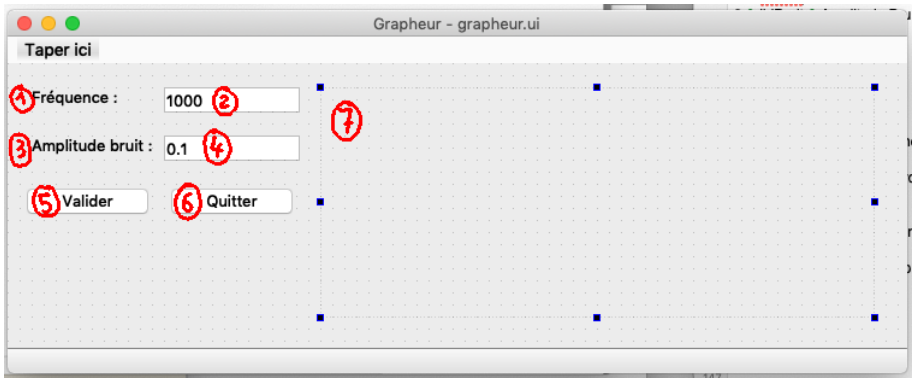

FIGURE 4 – Changement des propriétés

Remarque : Il est possible de changer texte associé à un élément en double cliquant dessus.

4.4 Cas particulier du wdGraph

Comme il n'existe pas de classe pour gérer directement les graphes faits avec Matplotlib, nous avons triché en insérant un widget wdGraph.

Nous allons maintenant demander à Designer de changer le type de ce Widget. Pour cela, sélectionnez le graphe puis cliquez avec le bouton droit de la souris et choisissez Promouvoir en...

Dans la fenêtre Widget promus, indiquez *MplWidget* comme non de la classe promue, et *mplwidget* (sans .h) comme fichier d'en-tête (voir figure 5. Cliquez ensuite sur ajouter puis promouvoir.

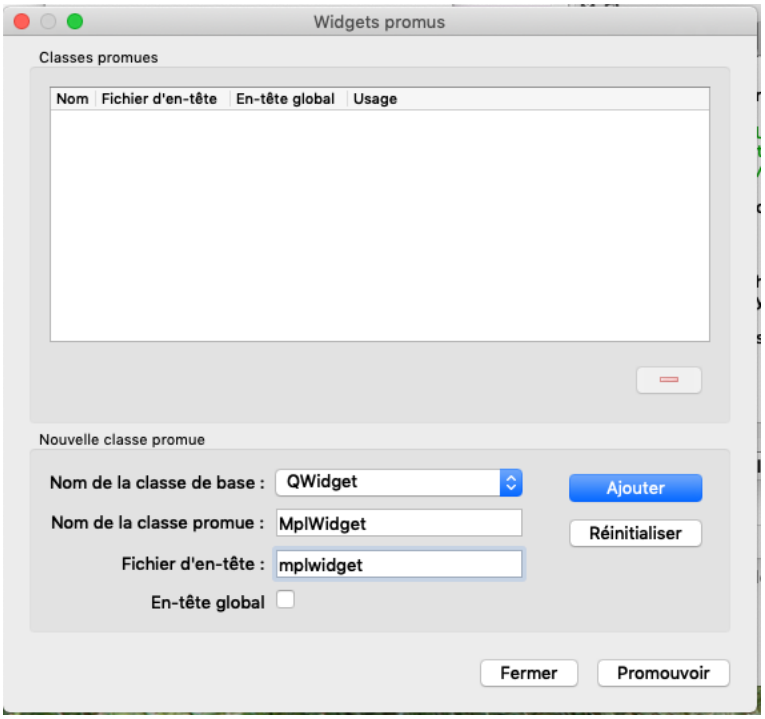

FIGURE 5 – Promotion de QWidget vers MplWidget

Vous pouvez maintenant vérifier dans l'éditeur de propriétés que le type du wdGraph est bien mplWidget.

4.5 Test et dernières retouches

Allez dans le menu Formulaire puis choisissez Prévisualisation avec Style Windows pour tester votre fenêtre.

Vous constaterez que l'appui sur le bouton Quitter ne fait rien. Pour y remédier, allez dans l'éditeur de signaux et slots, cliquez sur le bouton vert + et configurez-le comme l'illustre la figure 6 (il faut double cliquer sur un élément pour modifier la configuration).

Vous venez d'indiquer *sans une ligne de code* que si l'on clique sur le bouton btQuitter, on dit à la fenêtre Grapheur de ce fermer. Testez ce comportement en utilisant Formulaire -> Prévisualisation dans le menu et en appuyant sur votre bouton Quitter.

FIGURE 6 – Communication entre le bouton et la fenêtre

Cette présentation est un tour rapide de Qt Designer. Beaucoup plus d'options sont disponibles (notamment pour la mise en page). Pour vous perfectionner vous pouvez consulter [3] ou les nombreux sites web consacrés à PyQt5.

Votre fenêtre est maintenant créée, nous allons maintenant demander de générer le code Python associé à cette fenêtre.

5 Codage en Python

5.1 Appel de l'interface graphique dans le programme

Votre interface graphique est stockée dans le fichier grapheur.ui. Pour la transformer en code Python :

- Lancez la ligne de commande Anaconda Prompt,
- si vous avez accès a la ligne de commande :
  - aller dans le répertoire où ce trouve votre fichier grapheur.ui (utilisez cd pour changer de répertoire),
  - tapez la commande **pyuic5 grapheur.ui -o grapheur_ui.py**,
- sinon :

— lancer Spyder,
— aller dans le répertoire où ce trouve grapheur.ui en utilisant l'onglet explorateur de fichier de Spyder,
— tapez les commandes suivantes dans Spyder :

```
import os # importe les fonctions liées au système d'exploitation
os.system("pyuic5 grapheur.ui -o grapheur_ui.py")
```

— ouvrez le fichier généré avec Python et parcourez-le rapidement.

Copiez le fichier mplwidget.py dans le même répertoire que le fichier grapheur_ui.py.

Créez un nouveau script Python nommé grapheur.ui qui contient le code suivant :

```
import sys # Requis pour PyQt5

from PyQt5.QtWidgets import QMainWindow,QApplication,QLabel,QLineEdit,QPushButton
from PyQt5.QtCore    import pyqtSlot


# Importe la classe Ui_grapheur contenue dans grapheur_ui.py
from grapheur_ui import Ui_Grapheur

# Crée une classe définie à partir QMainWindow et de Ui_Grapheur
class GraphWindow(QMainWindow,Ui_Grapheur):
    # Méthode appelée lorsque l'on crée l'objet
    def __init__(self,parent=None):
        # Appel la méthode __init__ de QMainWindow
        super(GraphWindow,self).__init__(parent)
        # Appelle la méthode setupUi de Ui_grapheur pour remplir la fenêtre
        self.setupUi(self)


# Appellé au lancement le l'application
if __name__ == '__main__':
    # Initialise le gestionnaire Qt si cela n'a pas encore été fait
    if 'app' not in locals():
        app = QApplication(sys.argv)
    # Crée un objet associé à la fenêtre
    graphWin = GraphWindow()
    # Affiche la fenêtre
    graphWin.show()
    rc = app.exec_()
    sys.exit(rc)
```

Lancez le code, vous devriez voir la fenêtre s'afficher. Attention, seul le bouton quitter fonctionnera, comme pour la Prévisualisation dans Qt Designer. Pour réutiliser ce bout de code avec un autre projet, remplacez simplement Ui_Grapheur et grapheur_ui par le nom de la classe et le nom du fichier généré par pyuic5.

5.2 Gestion des événements

Un événement se produit lorsque l'on effectue une action dans la fenêtre : par exemple, lorsque l'on cliquer sur un bouton. Pour gérer un événement, nous allons associer une méthode permettant de réagir à l'appui sur le bouton Valider.

Pour cela, à l'intérieur de la classe GraphWindow, tapez @pyqtSlot() puis à la ligne suivante définissez une méthode on_btValider_clicked(self). Vous mettrez print("Valider pressé") à l'intérieur de la méthode pour réagir à l'appui sur le bouton. Le résultat sera :

```
# Crée une classe définie à partir QMainWindow et de Ui_Grapheur
class GraphWindow(QMainWindow,Ui_Grapheur):
    # Méthode appelée lorsque l'on crée l'objet
    def __init__(self,parent=None):
        # Appel la méthode __init__ de QMainWindow
        super(GraphWindow,self).__init__(parent)
        # Appelle la méthode setupUi de Ui_grapheur pour remplir la fenêtre
        self.setupUi(self)

    @pyqtSlot() # Définit un slot pour recevoir un signal
    def on_btValider_clicked(self): # signal clicked associé à l'élément btValider
        print("test") # On affiche simplement test sur la console
```

Le nom de la méthode suit la forme on_*nom widget*_*nom du signal* où le nom du signal peut être obtenu à l'aide de l'éditeur de signaux et slots de Qt Designer. Le décorateur @pyqtSlot() dit à Python que notre classe reçoit le signal et le transmet à la méthode (ou slot) définie dessous.

Essayez votre application : lorsque vous cliquez sur valider, vous devriez avoir un texte qui s'affiche dans la console.

5.3 Accès aux contenus d'une fenêtre

Lorsque vous avez créé la fenêtre, vous avez indiqué des noms d'objet (objectName) pour chaque élément de la fenêtre conformément à la table 1. Vous avez pu configurer l'objet (notamment le champ text associé) à l'aide de l'éditeur de propriétés.

Il est possible d'accéder aux propriétés associées aux objets dans une méthode de la classe GraphWindow à l'aide d'une commande de type *self.set_nom_propriete(valeur)* pour l'affectation et *valeur=self.nom_propriete* pour la lecture.

Par exemple, le code si dessous permet de :

— lire ce qu'il y a dans la zone de texte txtFreq,
— créer un message indiquant sin à xxx Hz (où xxx est la fréquence),
— de remplacer le texte du bouton Valider par le message.

```
@pyqtSlot() # Définit un slot pour recevoir un signal
def on_btValider_clicked(self): # signal clicked associé à l'élément btValider
    freq=self.txtFreq.text() # Range le contenu de la zone de texte txtFreq dans
 freq
    message="Sin à "+freq+" Hz" # Crée un message (ici freq est un texte)
    self.btValider.setText(message) # Ce message devient le texte associé au
 bouton valider
```

Remplacez la méthode on_btValider_clicked par la méthode ci-dessous et testez-la.

Remarques :

— La méthode text renvoie une chaîne de caractère. Si vous avez besoin d'un chiffre, il faudra le convertir en int ou float.

— Pour obtenir toutes les méthodes associées à un objet, il faut utiliser la commande dir. Par exemple, txtFreq étant du type QLineEdit (cf. Qt Designer), la commande dir(QLineEdit) vous donnera la liste des propriétés associées à txtFreq.

Modifier la méthode on_btValider_clicked en suivant les étapes.

1. Faites le nécessaire pour convertir les deux champs saisis (fréquence et amplitude bruit) en nombre flottant.
2. Générer ensuite à l'aide des paramètres saisis par l'utilisateur un signal sinusoïdal x de 1000 échantillons avec une fréquence d'échantillonnage de 200 kHz et un bruit blanc gaussien.
3. Afficher la courbe (pour l'instant à l'extérieur de la fenêtre en utilisant plt.plot).

5.4 Tracé d'une courbe

Le widget graphique s'appelle wdGraph. Pour dessiner sur ce graphe, il faut suivre deux étapes :

1. remplacer *plt.plot* par *self.wdGraph.canvas.ax.plot* et plus généralement *plt* par *self.wdGraph.canvas.ax*,
2. terminer par *self.wdGraph.canvas.draw()*,
3. si vous voulez effacer le contenu du graphe, vous pouvez utiliser la commande *self.wdGraph.canvas.ax.cla()*.

Faites le nécessaire pour afficher le graphe dans la fenêtre.

6 Conclusion

Nous avons créé une interface graphique simple permettant d'afficher une courbe en fonction des paramètres saisis par l'utilisateur. Il est possible de construire une application complète en Python (tous les widgets nécessaires sont présents dans Qt).

Pour réaliser une application plus complète, il vous sera nécessaire de lire des livres dédiés comme ceux proposés dans la bibliographie.

7 Bibliographie

Références

[1] B. Cordeau and L. Pointal. *Une introduction à Python 3*, chapter La programmation Orientée Objet. 2018.

[2] B. Cordeau and L. Pointal. *Une introduction à Python 3*, chapter La programmation Orientée Objet graphique. 2018.

[3] T. Cuvelier. *Applications graphiques en Python avec PyQt5*. 2017.