Chapitre 3
La disposition en PyQt

1. Introduction

Le chapitre Inventaires des widgets PyQt nous a permis d'inventorier les principaux contrôles (pour rappel ceux-ci sont nommés « widgets ») dans un contexte PyQt. Dans cet inventaire, nous avons vu que la disposition sur la fenêtre se faisait sans mode opératoire particulier. Jusqu'à maintenant, on place nos widgets à un point de coordonnées donné, en indiquant en général les coordonnées du coin gauche du widget manipulé.

Cette approche a ses limites et est sous certains aspects peu pratique. C'est pour cela que des widgets de disposition (*layout widgets*) existent en PyQt pour permettre d'agencer facilement les fenêtres développées. Ce type de contrôle dédié à l'organisation et à la structuration visuelle d'une fenêtre va constituer l'essentiel du contenu de ce chapitre.

2. Inventaire des widgets de disposition en PyQt

2.1 QHBoxLayout et QVBoxLayout

2.1.1 Introduction

Les deux premiers widgets de disposition que nous allons étudier sont `QHBoxLayout` et `QVBoxLayout`. Ils sont traités conjointement, car d'une part ils dérivent de la même classe `QBoxLayout` et d'autre part ils obéissent tous deux à la même logique de disposition. Cette logique de disposition est horizontale pour `QHBoxLayout` et verticale pour `QVBoxLayout`.

Les documentations en ligne des classes `QBoxLayout`, `QHBoxLayout`, `QVBoxLayout` sont respectivement aux adresses suivantes :

https://doc.qt.io/qt-5/qboxlayout.html

https://doc.qt.io/qt-5/qhboxlayout.html

https://doc.qt.io/qt-5/qvboxlayout.html

2.1.2 Premier exemple avec QVBoxLayout

Commençons par déclarer les classes dont nous allons avoir besoin.

```
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton,
QVBoxLayout
```

Créons une application et une fenêtre.

```
app = Qapplication([])
fenetre = QWidget()
```

Nous pouvons alors créer un objet de disposition verticale, c'est-à-dire que les widgets seront ajoutés les uns après les autres et selon une orientation verticale.

```
disposition = QVBoxLayout()
```

En l'occurrence, on ajoute cinq boutons qui seront ainsi alignés les uns après les autres, verticalement.

```
disposition.addWidget(QPushButton('Premier'))
disposition.addWidget(QPushButton('Second'))
disposition.addWidget(QPushButton('Troisième'))
disposition.addWidget(QPushButton('Quatrième'))
disposition.addWidget(QPushButton('Cinquième'))
```

Puis on associe notre objet de disposition à notre fenêtre grâce à la fonction `QVBoxLayout`.

```
fenetre.setLayout(disposition)
```

Enfin, on affiche la fenêtre. À aucun moment la disposition n'a nécessité l'utilisation de coordonnées relatives pour chacun des widgets mis en place.

```
fenetre.show()
```

Le code global est le suivant :

```
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton,
QVBoxLayout

app = Qapplication([])
fenetre = QWidget()

disposition = QVBoxLayout()
disposition.addWidget(QPushButton('Premier'))
disposition.addWidget(QPushButton('Second'))
disposition.addWidget(QPushButton('Troisième'))
disposition.addWidget(QPushButton('Quatrième'))
disposition.addWidget(QPushButton('Cinquième'))
fenetre.setLayout(disposition)

fenetre.show()
app.exec_()
```

Ce bout de code permet l'affichage de la petite fenêtre ci-dessous :

Utilisation d'un widget de disposition QVBoxLayout

2.1.3 Transposition de l'exemple avec QHBoxLayout

Utilisons à présent le code précédent en substituant `QVBoxLayout` par `QHBoxLayout` (voir en gras dans le bloc de code suivant) pour cette fois-ci obtenir sans effort une disposition horizontale de nos cinq boutons.

```
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton,
QHBoxLayout
app = Qapplication([])


fenetre = QWidget()

disposition = QHBoxLayout()
disposition.addWidget(QPushButton('Premier'))
disposition.addWidget(QPushButton('Second'))
disposition.addWidget(QPushButton('Troisième'))
disposition.addWidget(QPushButton('Quatrième'))
disposition.addWidget(QPushButton('Cinquième'))
fenetre.setLayout(disposition)

fenetre.show()
app.exec_()
```

Cette modification du code permet l'affichage suivant :

Utilisation d'un widget de disposition QHBoxLayout

Nous avons donc une première solution pour disposer des widgets au sein de la fenêtre, de manière verticale ou horizontale. Néanmoins, le besoin est souvent un peu plus compliqué. Le prochain contrôle de disposition présenté, qui utilise une approche par grille, permet ainsi de mettre en place une disposition plus fine.

2.2 QGridLayout

2.2.1 Introduction

L'idée du `QGridLayout` est de proposer une solution pour quadriller la fenêtre que l'on souhaite agencer. De cette manière, on peut choisir de disposer tel widget à telles coordonnées x et y du quadrillage. On maîtrise ainsi très finement la disposition et l'agencement de la fenêtre développée.

2.2.2 Exemple d'utilisation

On veut concevoir une calculatrice incluant quatre rangées horizontales de trois boutons. La valeur de chaque bouton correspond aux chiffres de 0 à 9 ainsi qu'aux symboles « + » et « = » d'une calculatrice standard (supposons donc que notre calculatrice ne permet que des additions).

De fait, nous désirons l'affichage de cette petite fenêtre :

Exemple d'utilisation de QGridLayout

Pour obtenir cette fenêtre, on instancie un `QGridLayout`.

```
disposition = QGridLayout()
```

Puis on ajoute nos boutons en spécifiant les coordonnées de chaque bouton dans la grille. Ainsi, on déclare pour chaque widget ajouté la coordonnée horizontale puis la coordonnée verticale (dans la grille).

```
disposition.addWidget(QPushButton("0"), 0, 0)
disposition.addWidget(QPushButton("1"), 0, 1)
disposition.addWidget(QPushButton("2"), 0, 2)
disposition.addWidget(QPushButton("3"), 1, 0)
disposition.addWidget(QPushButton("4"), 1, 1)
disposition.addWidget(QPushButton("5"), 1, 2)
disposition.addWidget(QPushButton("6"), 2, 0)
disposition.addWidget(QPushButton("7"), 2, 1)
disposition.addWidget(QPushButton("8"), 2, 2)
disposition.addWidget(QPushButton("9"), 3, 0)
disposition.addWidget(QPushButton("+"), 3, 1)
disposition.addWidget(QPushButton("="), 3, 2)
```

Puis on associe cette disposition (*layout*) à notre fenêtre.

```
fenetre.setLayout(disposition)
```

Le code global de l'exemple est le suivant :

```
 from PyQt5.QtWidgets import QApplication, QWidget, QPushButton,
QgridLayout

app = Qapplication([])

fenetre = QWidget()

fenetre.setGeometry(100, 100, 200, 100)
fenetre.setWindowTitle("QGridLayout")
disposition = QgridLayout()

disposition.addWidget(QPushButton("0"), 0, 0)
disposition.addWidget(QPushButton("1"), 0, 1)
disposition.addWidget(QPushButton("2"), 0, 2)
disposition.addWidget(QPushButton("3"), 1, 0)
disposition.addWidget(QPushButton("4"), 1, 1)
disposition.addWidget(QPushButton("5"), 1, 2)
disposition.addWidget(QPushButton("6"), 2, 0)
disposition.addWidget(QPushButton("7"), 2, 1)
disposition.addWidget(QPushButton("8"), 2, 2)
disposition.addWidget(QPushButton("9"), 3, 0)
disposition.addWidget(QPushButton("+"), 3, 1)
disposition.addWidget(QPushButton("="), 3, 2)

fenetre.setLayout(disposition)
fenetre.show()

app.exec_()
```

2.2.3 Plus loin avec la fonction addWidget

Un widget peut occuper plus d'une case, soit horizontalement, soit verticalement.

Prenons le prototype de la fonction `addWidget`.

```
addWidget (self, QWidget, row, column, rowSpan, columnSpan,
Qt.Alignment alignment = 0)
```