Introduction à PyQt

Gilles BAILLY

gilles.bailly@sorbonne-universite.fr

Diapositives inspirées de Sylvain Malacria et Eric Lecolinet

Qu’est ce que Qt ?

Librairie graphique écrite en C++ par la société TrollTech

- Mécanisme pour interagir :
  - avec l’utilisateur (bouton, liste déroulante, ..)
  - avec le système (OpenGl, Xml, SQL, sockets, plugin…)

Multi-Plateforme

- Windows I Mac I Linux
- Android I iOS I WinRT I BlackBerry* I SailFish
- Embedded Linux I Embedded Android I Windows Embedded

Gratuit (GPL), mais dispose aussi d’une licence commerciale

Approche : Ecrire une fois, compiler n’importe où

Historique

1990	Haavard & Eirik ont l'idée de créer une librairie graphique
1995	**Qt 0.9** Première distribution publique pour X11/Linux
1996	**Qt 1.0** (licences commerciales et open source)
1999	**Qt 2.0** en Open Source (Licence QPL)
2001	**Qt 3.0** support Mac et *Qt designer*
2005	**Qt 4.0** (licence GPL 2.0 pour toutes plateforme)
2008	Nokia rachète Trolltech (société mère de Qt) et ses 250 employés
2009	Distribution Qt 4.5 avec *QtCreator*
2008	**Qt 4.5**
2010	**Qt 4.6** animation; GraphicScene; machine à état; gestes
2011	Qt est racheté par Digia (objectif *Android, iOS et Windows 8*)
2012	**Qt 5.0** Qt Quick (création d'interfaces dynamiques)
2015	20ème anniversaire de la première distribution publique
2016	Support Qt Mobile (iOS, Android)

Pourquoi Qt?

Performance (C++)

Relativement Simple

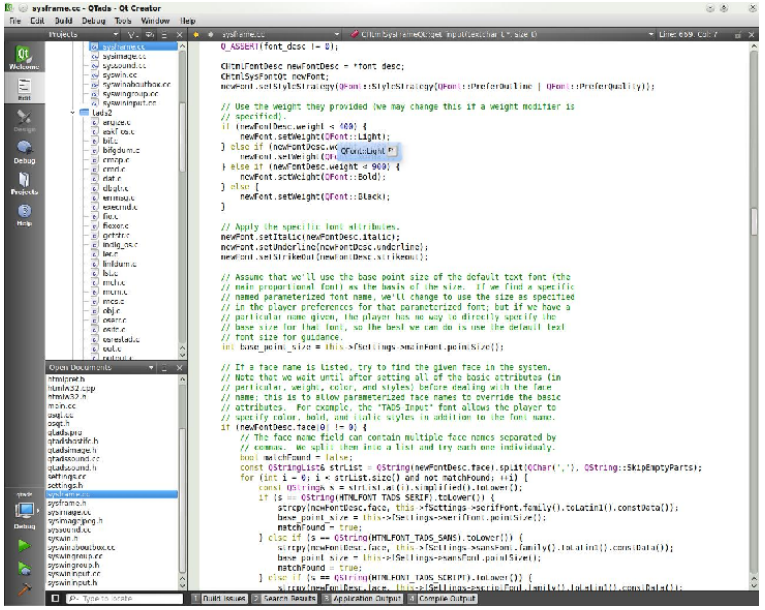
Gratuit (GPL) et code source

Nombreux outils

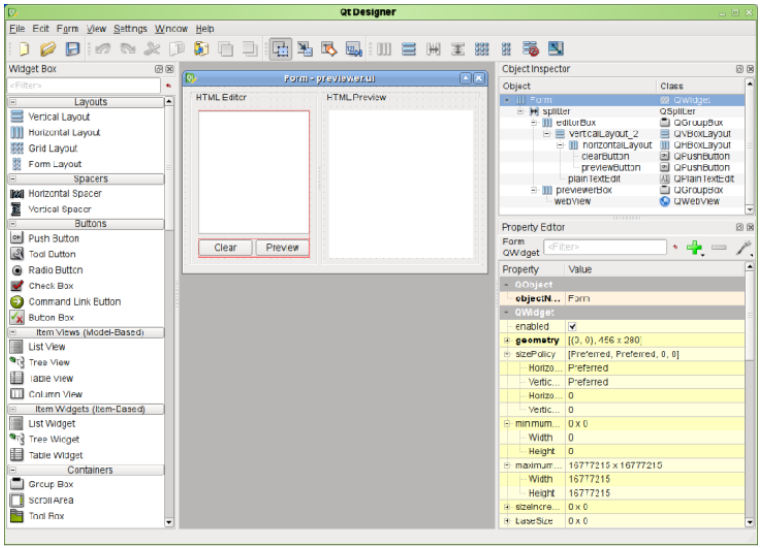
- Générateur d'interface : Qt Designer
- Internationalisation : Qt Linguist
- Documentation : Qt Assistant
- Examples : Qt Examples
- Programmation : Qt Creator (eclipse)

Multi-Plateformes

- Look and feel simulé



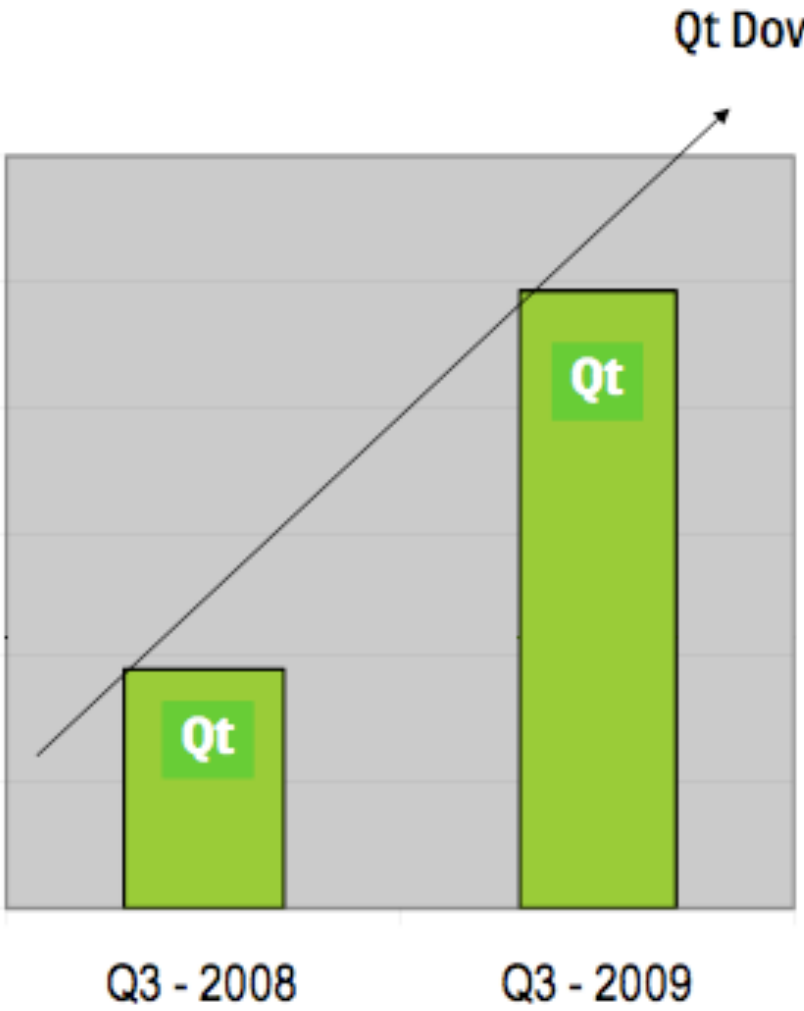
Qt creator



Designer

Aujourd'hui

Augmentation de 250% des téléchargements de la GPL



Qt 5 : 10 000 téléchargement par jour

Qt Developer Days 2009

Qt 5.2 reaches 500,000 downloads in record time

THU, FEB 06, 2014 13:37 CET

Qt framework established as major player for cross-platform application development

February 6, 2014 — Digia has announced that the Qt cross-platform application and user interface (UI) development framework has exceeded 500,000 downloads since the launch of version 5.2 in mid-December 2013. The rate of uptake exceeds all previous Qt introductions and underlines the framework's arrival as a significant player for mobile app development as well as for desktop and embedded platforms. Qt 5.2 extended full support to Android and iOS in addition to 12 other operating systems.

Tommi Laitinen, Digia Qt Senior VP International Products, commented: "Qt has a large and enthusiastic user community of developers who are passionate about the framework and the stunning applications it enables them to develop on desktop and embedded platforms. Now Android and iOS developers are experiencing the same buzz about what Qt can do for them. This has driven the average download rate beyond 10,000 per day and positions Qt to become a major player in mobile and tablet app development. This tremendous uptake of Qt in 2013 and the release of Qt 5.2 has clearly solidified our developer community, which is now larger than ever."

Interest in Qt 5.2 has been spread globally with the highest download rates occurring in US, China and Europe. Qt 5.2's support for an extensive range of desktop, mobile and embedded operating system platforms, together with its enhanced core performance and functionality, make it the only cross-platform development framework to enable intuitive and highly performing platform-independent software applications no matter the target device or screen size.

Digia is hosting a series of roadshows for developers of mobile apps, starting in San Francisco, CA, on February 19 and then moving to UK and Germany in March. For details of the Qt Mobile Roadshows and to register, visit the Digia website http://qt.digia.com or http://qt.digia.com/Qt-Mobile-Roadshow.

Aujourd'hui

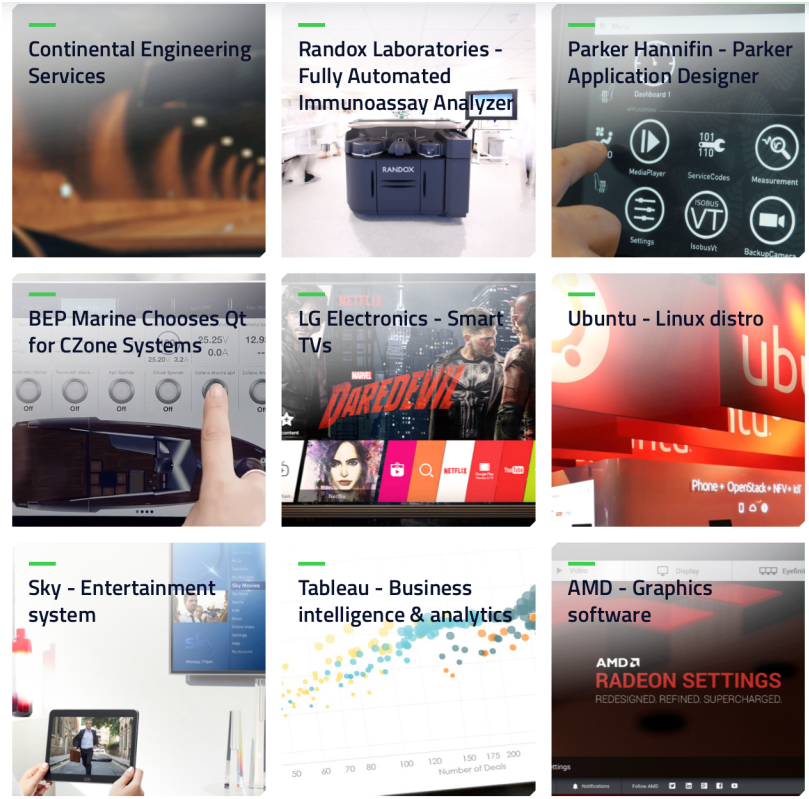
Utilisateurs de Qt :

- Adobe Photoshop, Autodesk Maya, Google Earth, Skype, Spotify, VLC

Bindings (java, **python**, c#)

Qt 5 language bindings

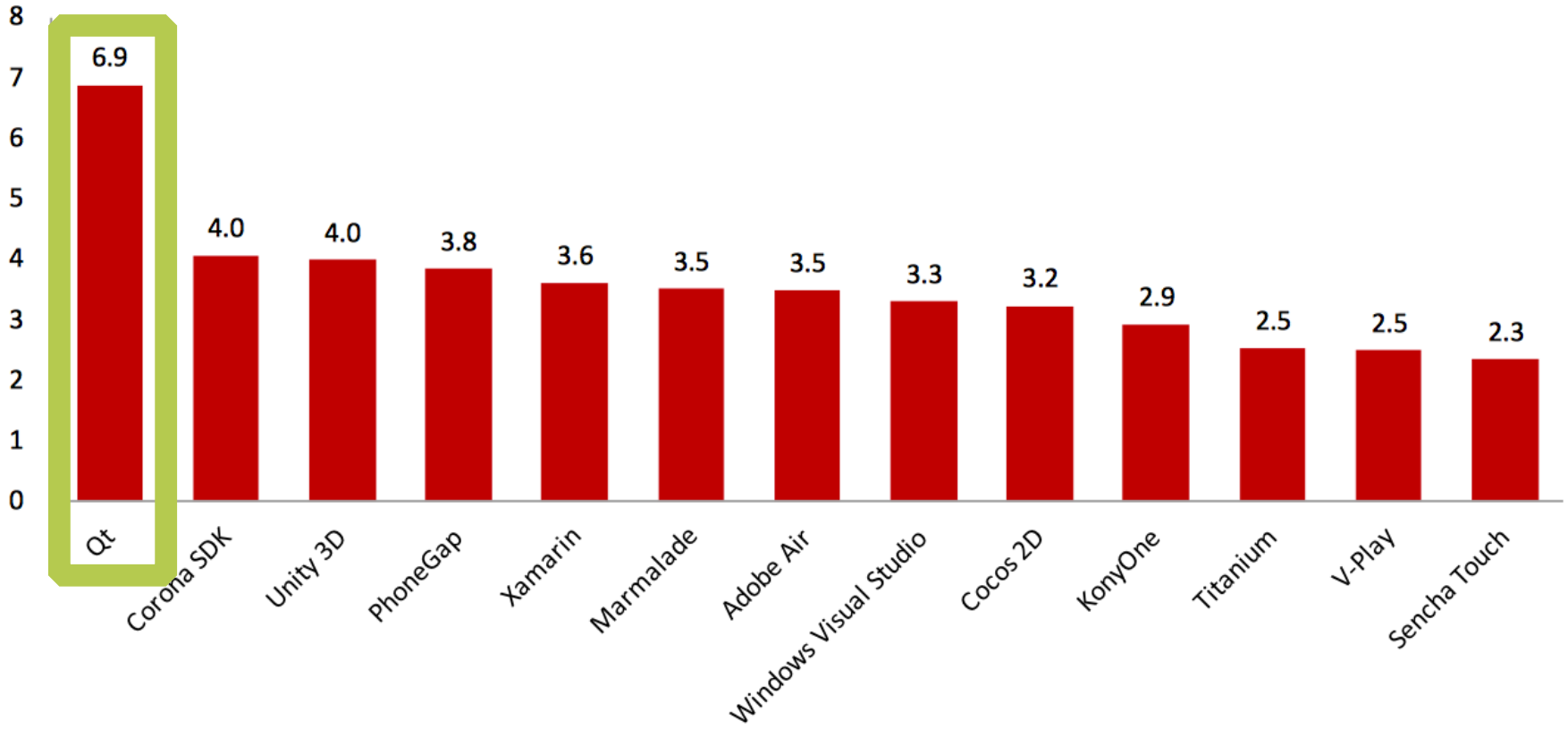
Language	Name: description of binding	License for open source applications	License for proprietary applications
C++	Qt – built-in[1]	GPL or LGPL	LGPL or Commercial proprietary[2]
C#	QtSharp	Apache License 2.0	
Crystal	qt5.cr	MPL2	
Go	qt[3] – therecipe/qt	LGPL[4]	
Go	Go QML – only for QML	LGPL	
Haskell	Qtah	LGPL	
Haskell	HsQML	3-clause BSD	
JavaScript	QtQuick – built into Qt[5]	LGPL	LGPL or Commercial proprietary[2]
JavaScript	BrigJs – Node.js binding for QML	MIT[6]	
Java	Qt Jambi 5+ [7]	LGPL	LGPL or Commercial proprietary[2]
Julia	QML.jl – only for QML[8]	LGPL	
Pascal	Lazarus with Qt5 interface	LGPL	
Python	PyOtherSide – only for QML	ISC license	
Python	PyQt[9]	GPL	Commercial proprietary
Python	PySide2 – Qt's official Python bindings[10]	LGPL	LGPL or Commercial proprietary[2]
Python	PythonQt[11]	LGPL	
OCaml	lablqml – QML support	LGPL	LGPL or Commercial
QML	QtQuick – built into Qt[12]	LGPL	LGPL or Commercial proprietary[2]
Ring	RingQt	MIT License	
Ruby	ruby-qml – only for QML	MIT License	
Rust	qmlrs – only for QML	MIT License or Apache License 2.0	
D	QtE5	GPL	
Language	Name: description of binding	License for open source applications	License for proprietary applications



Qt is the leader of cross-platform app development

Qt is the leader in true cross-platform app development. Users of Qt publish their apps on almost 7 different platforms, whereas all the other users release their apps on 4 or less platforms.

research2guidance 26: Average number of platforms which users publish their apps developed with a CP Tool on

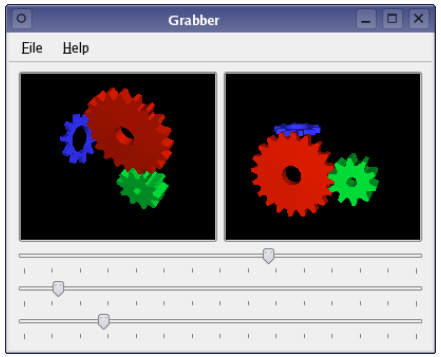


research2guidance 39: Top 10 Cost performance-ratio

Rank	Tool	Poor value or costly	Average	Okay or good value	# Ratings
1	Qt	-2%	0%	98%	104
2	Titanium	-6%	2%	92%	51
3	Unity	-4%	5%	91%	103
4	Corona SDK	-7%	2%	91%	97
5	Windows Visual Studio	0%	16%	84%	64
6	Cocos 2D	0%	17%	83%	54
7	Adobe Air	-4%	13%	83%	82
8	Xamarin	-7%	13%	80%	99
9	PhoneGap	-3%	17%	80%	88
10	KonyOne	-11%	11%	78%	55
Benchmark (Average all tools)		**-5%**	**14%**	**81%**	

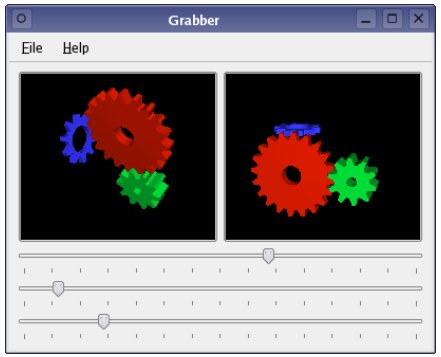
Research2guidance, CPT Benchmarking 2014

Aujourd'hui

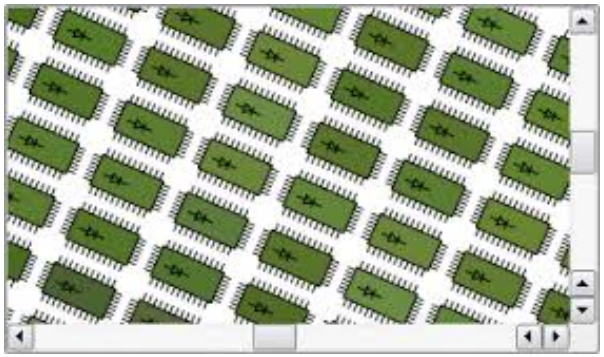


Qt Widgets

Aujourd'hui



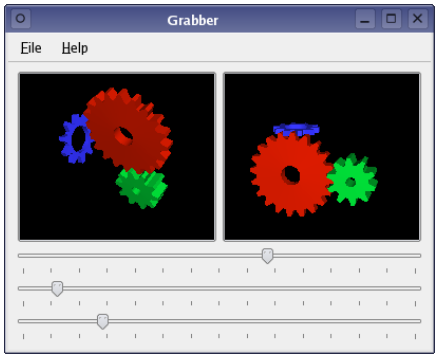
Qt Widgets



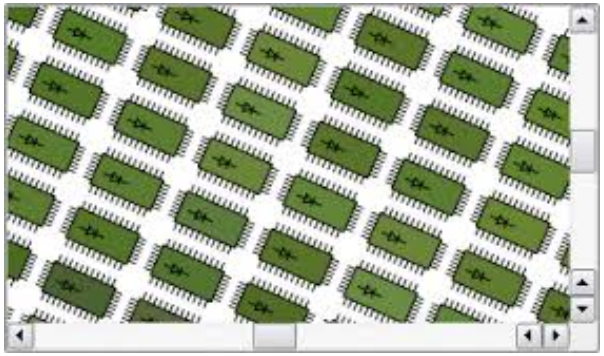
Qt Graphics view

Qt quick /QML

Aujourd'hui



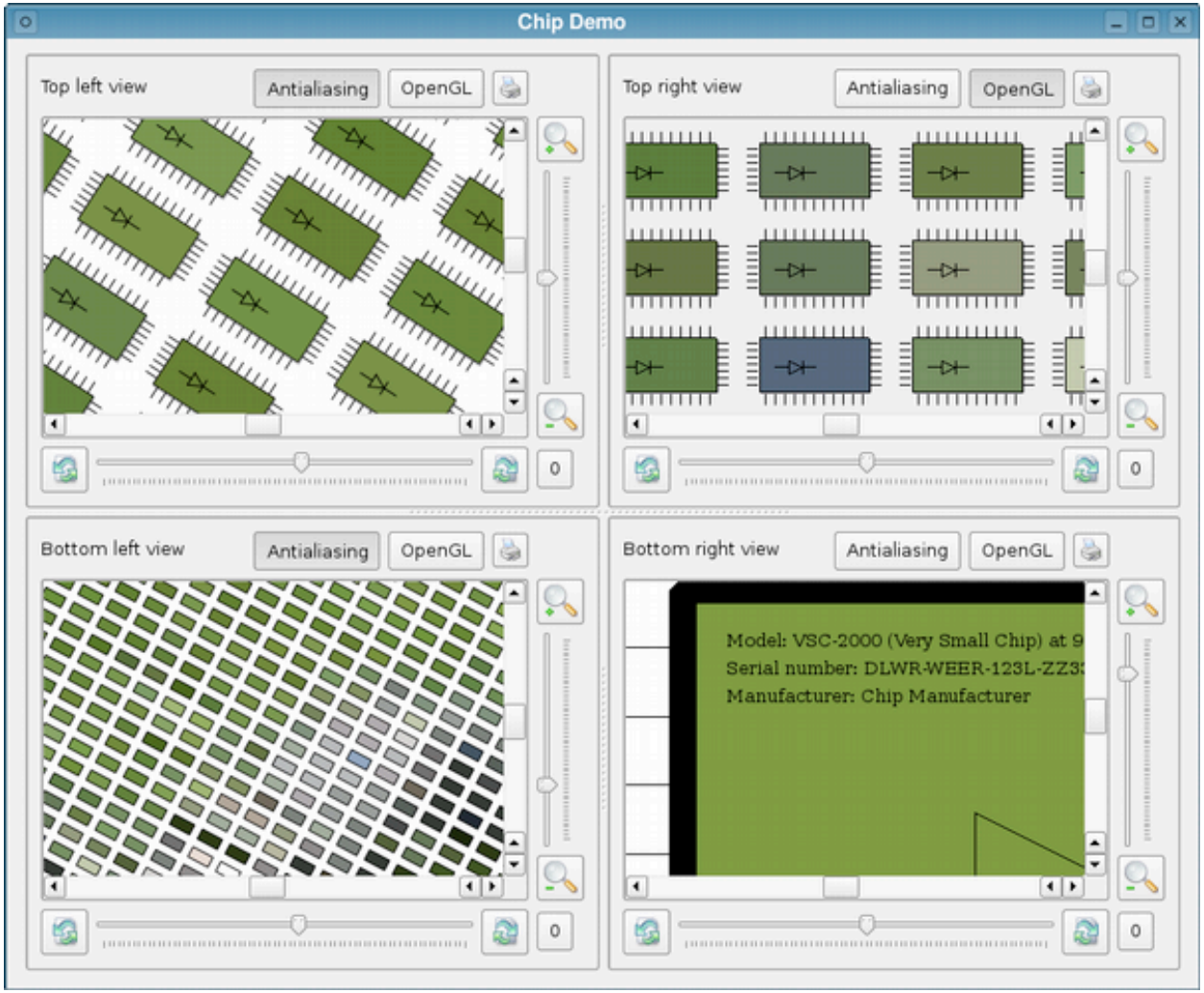
v.s.



Qt Widgets

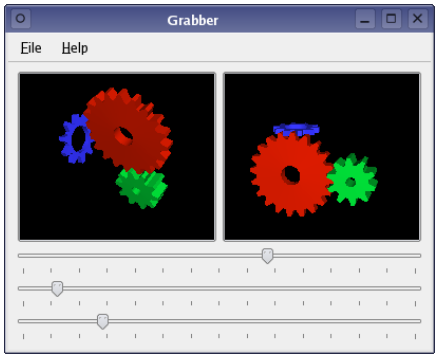
Qt Graphics view

- Widgets ne peuvent pas être *transformés*
- Widgets utilisent des coordonnées en *pixels ;* GraphicsItems en unités logiques (int vs doubles)
- Widgets peuvent être utilisés dans des layouts
- 4000000 widgets rament, mais 4000000 items fonctionnent très bien



http://doc.qt.io/qt-5/qtwidgets-graphicsview-chip-example.html

Aujourd'hui



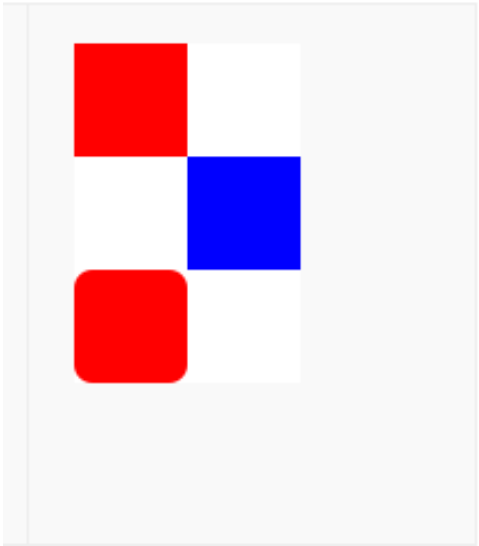
V.S.


```
import QtQuick 2.0

 Rectangle {
      id: canvas
      width: 200
      height: 200
      color: "blue"

      Image {
          id: logo
          source: "pics/logo.png"
          anchors.centerIn: parent
          x: canvas.height / 5
      }
 }
```

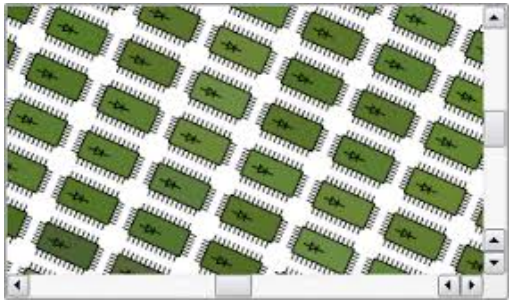
Qt Widgets Qt quick/QML Language déclaratif

- QML est basé sur JSON (Language); QtQuick (library)
- QWidgets sont plus matures, flexibles and ont plus de fonctionnalités
- Qt Quick se concentre sur les animation and transition
- Qt Quick est (pour l'instant) plutôt pour dispositifs mobiles
- Qt Quick va (peut-être) remplacer QWidgets un jour
- Qt Quick est (peut-être) mieux pour les designers (non-informaticiens)

```qml
// application.qml
import QtQuick 2.3

Column {
    Button { width: 50; height: 50 }
    Button { x: 50; width: 100; height: 50; color: "blue" }
    Button { width: 50; height: 50; radius: 8 }
}
```



Aujourd'hui



v.s.


```
import QtQuick 2.0

 Rectangle {
      id: canvas
      width: 200
      height: 200
      color: "blue"

      Image {
           id: logo
           source: "pics/logo.png"
           anchors.centerIn: parent
           x: canvas.height / 5
      }
 }
```

QGraphicsView Qt quick/QML Language déclaratif

- Qt Quick Graphics engine only works with OpenGL
- Drawing complex shapes easier with QGraphicView
- Qt Quick: QML & Javascript
- QML is more compatible with Widget

PyQt

Bindings Python v2 et v3 pour Qt

Développé par Riverbank Computing Limited

Existe pour les mêmes plateformes que Qt

Binding le plus populaire avec *PySide*

PyQt5 pour **Qt5**, PyQt4 pour Qt4

Licences différentes de Qt (GNU GPL 3 et license commerciale)

PyQt peut générer du code python depuis Qt Designer

Possibilité d'ajouter des widgets écrits en PyQt à Qt Designer

PyQt n'est pas QUE pour la programmation de GUI !!!

Objectifs de ce cours

Comment installer Qt?

(Rappels) programmation **Python**

Premiers pas avec **PyQt**

Introduction aux **Signaux** et **Slots** de Qt

Aperçu des principales classes de Qt

Comment installer pyqt

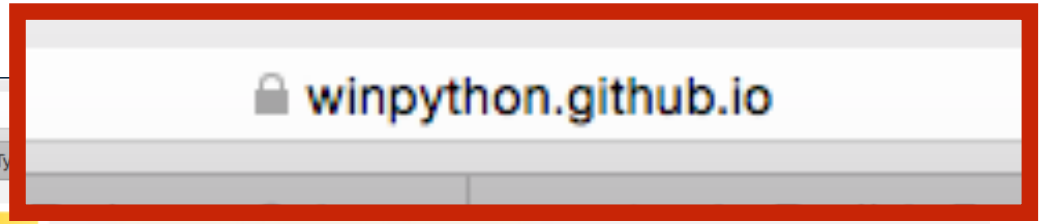
Utilisateurs Windows (winPython)

Utilisateur Mac (Anaconda)

Salle Machine

Utiliser de préférence les salles machines où pyqt est déjà installé
si ce n’est pas possible favoriser Windows à MAC

Utilisateurs Windows



El Chapo », prince... | Cours : Conception c... | #8854 *Ty

WinPython

The easiest way to run Python, Spyder with SciPy and friends out of the box on any Windows PC, without installing anything!

Project Home is on Github, downloads page are on Sourceforge, Discussion group is on Google Groups, md5 and sha1 there

Recent Releases

Release 2018-04 of November 4th, 2018

Highlights (**): Ipython-7.7.1, scikit-learn-0.20.0, bokeh-1.0.0, jupyterlab-0.35.3 (Zero Version)

- WinPython **3.6**.7.0Qt5-64bit (*) Changelog, Packages and Downloads or Github Downloads
- WinPython **3.6**.7.0Qt5-32bit (*) Changelog, Packages and Downloads
- WinPython **3.7**.1.0-64bit (*) Changelog, Packages and Downloads
- WinPython **3.7**.1.0-32bit (*) Changelog, Packages and Downloads

Release 2018-03 of September 4th, 2018

Highlights (**): pandas-0.23.4, scikit-learn-0.19.2, spyder-3.3.1, jupyterlab-0.34.7 , rise-5.4.1, idlex-1.18 (Zero Version)

- WinPython **3.6**.6.2Qt5-64bit (*) Changelog, Packages and Downloads or Github Downloads
- WinPython **3.6**.6.2Qt5-32bit (*) Changelog, Packages and Downloads
- WinPython **3.7**.0.2-64bit (*) Changelog, Packages and Downloads
- WinPython **3.7**.0.2-32bit (*) Changelog, Packages and Downloads

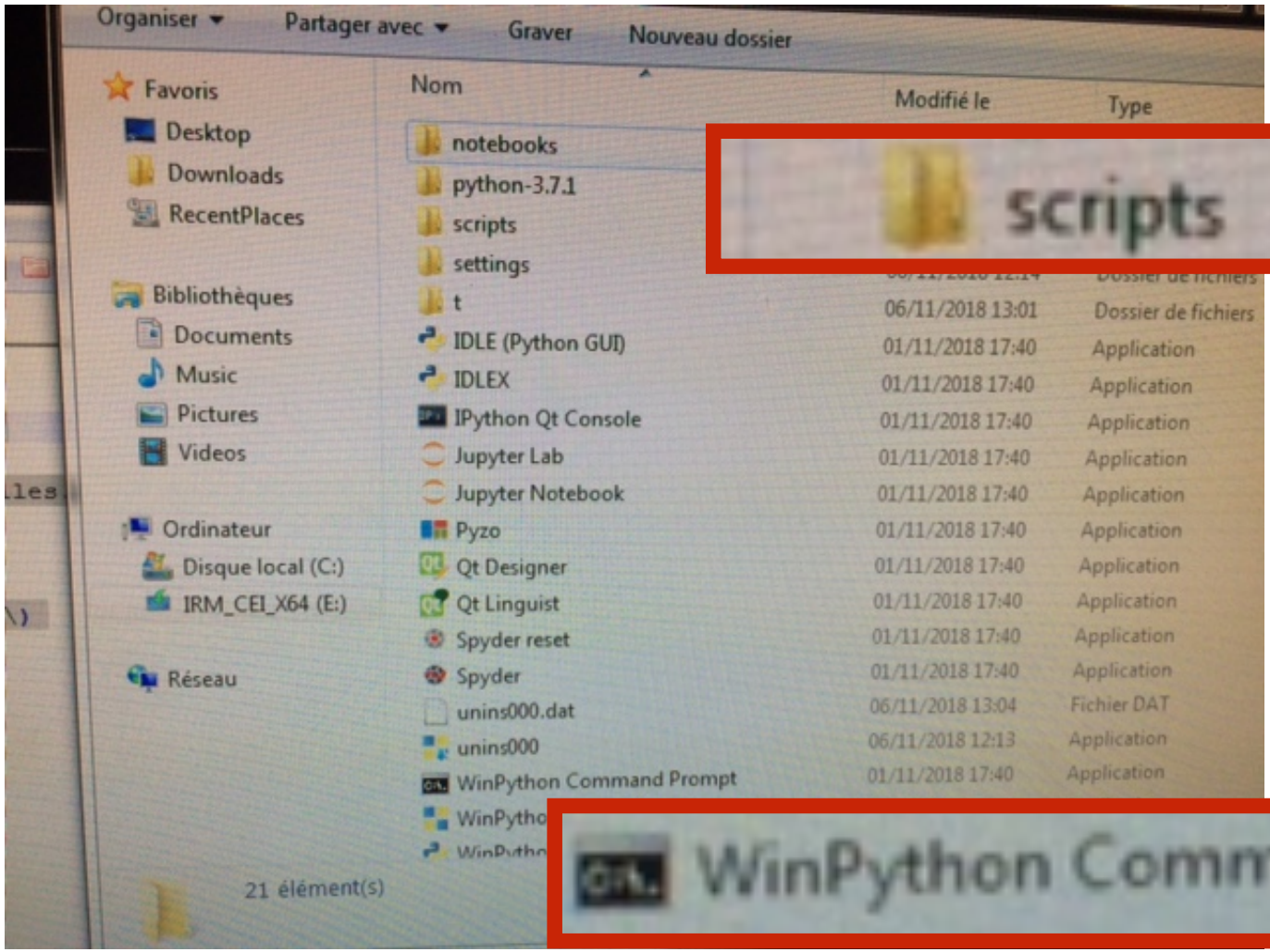
Release 2018-02 of July 21st, 2018

Highlights (**): Python-3.7.0, pandas-0.23.3, jupyterlab-0.32.1 (beta 2) + nodejs-8.11.2, scipy-1.1.0, spyder-3.3.0 (Zero

WinPython

WinPython installe Python, Qt ainsi que d'autres librairies utiles en python.

Après installation, vous trouverez un répertoire avec ce contenu:



pour mettre vos fichiers python

pour lancer votre programme

Editer et executer votre programme sous Windows

Utiliser NotePad++ (pas NotePad) pour éditer votre programme

C’est un éditeur de code simple

Lancer le terminal (winPython command prompt ; voir slide precedent) pour pouvoir executer votre programme

Aller dans le repertoire scripts (voir slide precedent) si ce n’est pas deja le bon repertoire

lancer votre programme: python <nom_du_program.py>

Comment installer PyQt sous MAC?

Utilisation de Anaconda

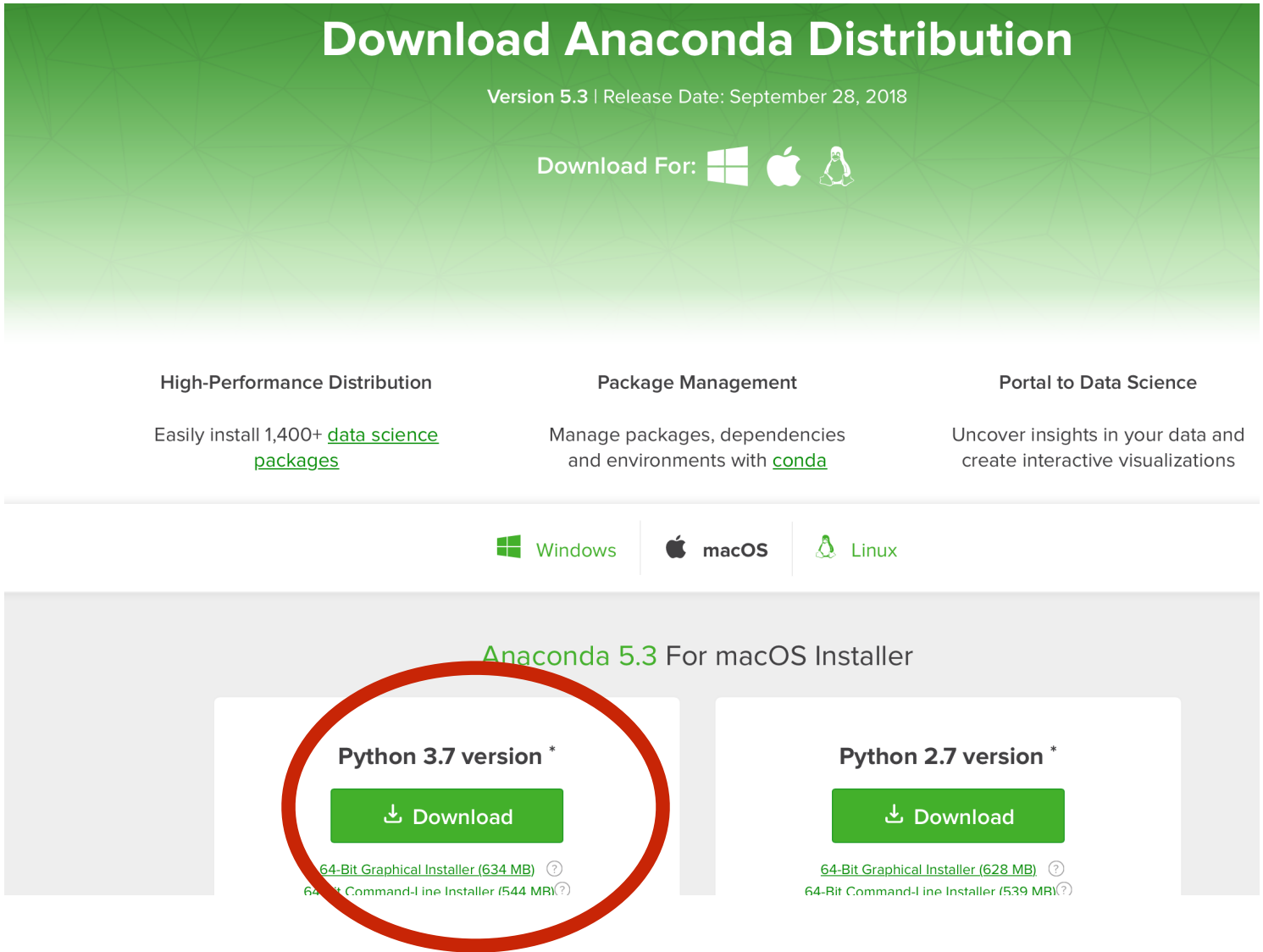
plateforme multi plateforme (Win, Mac) qui installe à la fois python et pyqt

permet également d'installer de nombreuses librairies scientifiques

https://www.anaconda.com/download/

Diapositives adaptées de John Smith

Installation: https://www.anaconda.com/download/



Comment éditer votre programme?

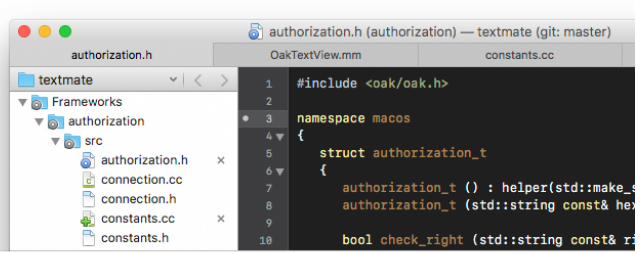
Vous pouvez utiliser TextMate (https://macromates.com) qui est gratuit

TextMate for macOS

Powerful and customizable text editor with support for a huge list of programming languages and developed as open source.

Download TextMate 2.0

Requires macOS 10.9 or later.



Multiple Carets

Making multiple changes at once, swapping pieces of code, and a lot more is made trivial with TextMate's easy way to add multiple insertion points.

File Search

Select what you want to search, what you want to search for, and TextMate will present the results in a way that makes it easy to jump between matches, extract matched text, or preview desired replacements.

Version Control

See what files have changes in the file browser view, what lines have changes in the editor view, bring up a diff of the current file's changes, commit a subset, TextMate supports it all for all the major version control systems.

Scoped Settings

One file mixing languages? Projects using different build systems? Third party code with different formatting preferences? TextMate can handle it all by associating detailed scope selectors with key shortcuts, settings, etc.

Commands

The UNIX underpinnings of macOS allows custom actions to be written in any language that can work with stdin, stdout, and environment variables, and for complex interactions TextMate expose both WebKit and a dialog framework for Mac-native or HTML-based interfaces.

Snippets

Commonly used pieces of text or code can be turned into snippets with placeholders, transformations, and more, to have them adapt according to the context in which they are used.

Bundles

TextMate uses bundles for customization and supports a countless number of different languages, markup systems, workflows, and more.

Macros

Repetitive work can be eliminated with macros. Either save them for repeated use or record a scratch macro for immediate replay.

And More…

Clipboard history, custom themes, live HTML/Markdown preview, foldable code blocks, indented soft wrap, etc.

Comment exécuter votre programme

Ouvrer le terminal sous mac

Tapez le raccourci clavier **CMD + Barre d'espace** pour ouvrir spotlight search

Taper Terminal

Allez ensuite dans le repertoire contenant votre fichier:

à l'aide des commandes **cd** (pour rentrer dans un repertoire) et **ls** (pour visualiser le contenu du repertoire)

Une fois dans le bon repertoire

tapez python <nom du programme.py>

En salle Machine (14.15-509)

Qt et Pyqt a deja été installé.

- installation de PyQt5 PyQtChart pour python3

- Qt5 est hébergé dans /opt/Qt-5.3

N’hesitez pas a me contacter si vous rencontrez des problèmes

Documentation

http://pyqt.sourceforge.net/Docs/PyQt5/

Objectifs de ce cours

Comment installer Qt?

(Rappels) programmation **Python**

Premiers pas avec **PyQt**

Introduction aux **Signaux** et **Slots** de Qt

Aperçu des principales classes de Qt

Python

Langage de programmation objet

Typage dynamique fort

Placé sous licence libre

Enormément de bibliothèques

Rapide d’utilisation

Nombreuses extensions destinées au calcul numérique

Syntaxe

Syntaxe C	Syntaxe Python

`int factorielle(int n) {`
`  if (n < 2) {`
`  return 1;`
`  } else {`
`  return n * factorielle(n - 1);`
`  }`
`}`
	`def factorielle(n):`
	`  if n < 2:`
	`  return 1`
	`  else:`
	`  return n * factorielle(n - 1)`

Syntaxe

Syntaxe C	Syntaxe Python

`int factorielle(int n) {`
`  if (n < 2) {`
`  return 1;`
`  } else {`
`  return n * factorielle(n - 1);`
`  }`
`}`
	`def factorielle(n):`
	`  if n < 2:`
	`  return 1`
	`  else:`
	`  return n * factorielle(n - 1)`

Attention aux tabulations !!!

Typage dynamique fort

~~int a = 4~~

```
a = 4
type(a)                    <class 'int'>
```

```
a = 4.1
type(a)                    <class 'float'>
```

Quelques types de base

Booléen

Numériques

- int
- long
- float
- complex

Collections

- list
- tuple
- set
- dict
- etc.

Quelques types de base

Booléen

Numériques

- int
- long
- float
- complex

Collections

- **list**
- tuple
- set
- dict
- etc.

```
list1 = ['physics','chemistry',1200]
print(list1[0])
```

>>> ‘physics’

```
list1.append("bla")
print(list1[3])
```

>>> ‘bla’

```
print(list1[1:3])
```

>>> ['chemistry', 1200]

Quelques types de base

- Booléen
- Numériques
  - int
  - long
  - float
  - complex
- **Collections**
  - list
  - **tuple**
  - set
  - dict
  - etc.

```
list1 = ['physics','chemistry',1200]
print(list1[0])
```

>>> ‘physics’

```
list1.append("bla")
print(list1[3])
```

>>> ‘bla’

```
print(list1[1:3])
```

>>> ['chemistry', 1200]

Opérations de base sur collections (List, Tuple)

```
alphabetT = ('a','b','c','d','e','f','g','h','i','j','k')
alphabetL = [‘a','b','c','d','e','f','g','h','i','j','k']

alphabetT2 = ‘a’,’b','c','d',('e','é','è'),'f','g','h','i','j','k'

len(alphabetT)
```

➡ Tuple

➡ List

➡ Nombre d’éléments

```
>>> 11

len(alphabetT2[4])
```

```
>>> 3

alphabetT[-2]
```

➡ Accès *depuis la fin*

```
>>> ‘j’

for char in alphabetT:
    print(char)
```

➡ Boucle *for each*

```
for i in range(len(alphabetT)):
    print(alphabetT[i])
```

➡ Boucle *itérative*

```
for i in range(2,len(alphabetT)):
    print(alphabetT[i])
```

Entrées/Sorties fichiers

```
file  = open("text.txt", "r+")
text = file.read()
file.write("blabla\n")
file.close()
```

➡ **(r,w,a,r+)** **r**ead, **w**rite, **a**ppend, read**+**write

Classes

```
class Voiture(Vehicule):
    #commentaire
    nbRoues=4
```

➡ Déclaration d'une classe Voiture qui hérite de véhicule

```
    def __init__(self,marque,couleur):
        super().__init__()
        self.couleur=couleur
        self.marque=marque
```

➡ Déclaration d'un constructeur
➡ Appel du constructeur de la classe mère
➡ Déclaration d'une variable d'instance
➡ Déclaration d'une variable d'instance

➡ Déclaration conventionnelle d'une méthode main()

```
def main():
    audirouge = Voiture('audi','rouge')
    print(audirouge.couleur)
```

➡ Appel automatique *Main*

```
if __name__ == "__main__":
     main(sys.argv)
```

Les principaux modules

QtCore

QtWidgets

QtGUI

QtBluetooth

QtOpenGL

QtSript/QtScriptTools

QtSql

QtSvg

QtWebKit

QtXml/QtXmlPatterns

QtMultimedia

QtSensors

QtChart

Module QtCore

QObject

Type de base :

- QChar, QDate, **QString**, QStringList, Qtime,…

File systems :

- QDir, QFile,…

Container :

- QList, Qmap, Qpair, QSet, QVector,…

Graphique :

- QLine, Qpoint, QRect, QSize …

Thread :

- QThread, Qmutex, …

Autres :

- QTimer, …

QString

Codage Unicode 16 bits

- Suite de QChars
- 1 caractère = 1 QChar de 16 bits (cas usuel)
- 1 caractère = 2 QChars de 16 bits (pour valeurs > 65535)

Conversions d'une QString :

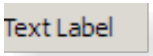
- **toAscii**( ) : ASCII 8 bits
- **toLatin1**( ) : Latin-1 (ISO 8859-1) 8 bits
- **toUtf8**( ) : UTF-8 Unicode multibyte (1 caractère = 1 à 4 octets)
- **toLocal8Bit**( ) : codage local 8 bits

Module QtWidgets

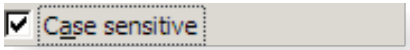
Qwidget

QPushButton

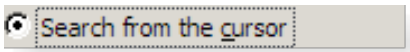
QLabel



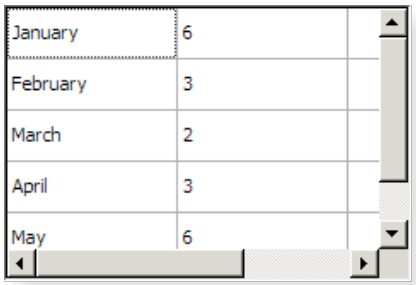
QCheckBox



QRadioButton

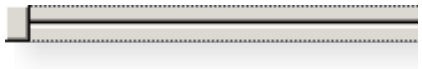


QTableView

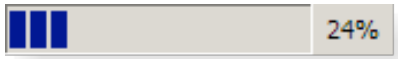


QComboBox

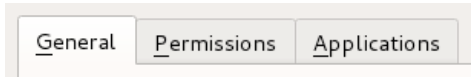
QSlider



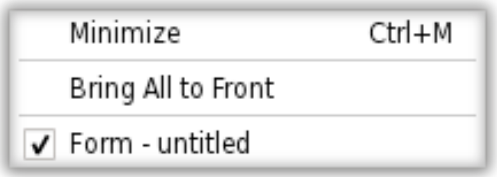
QProgressBar



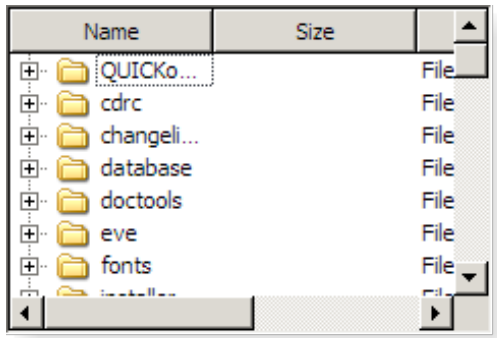
QTabBar



QMenu

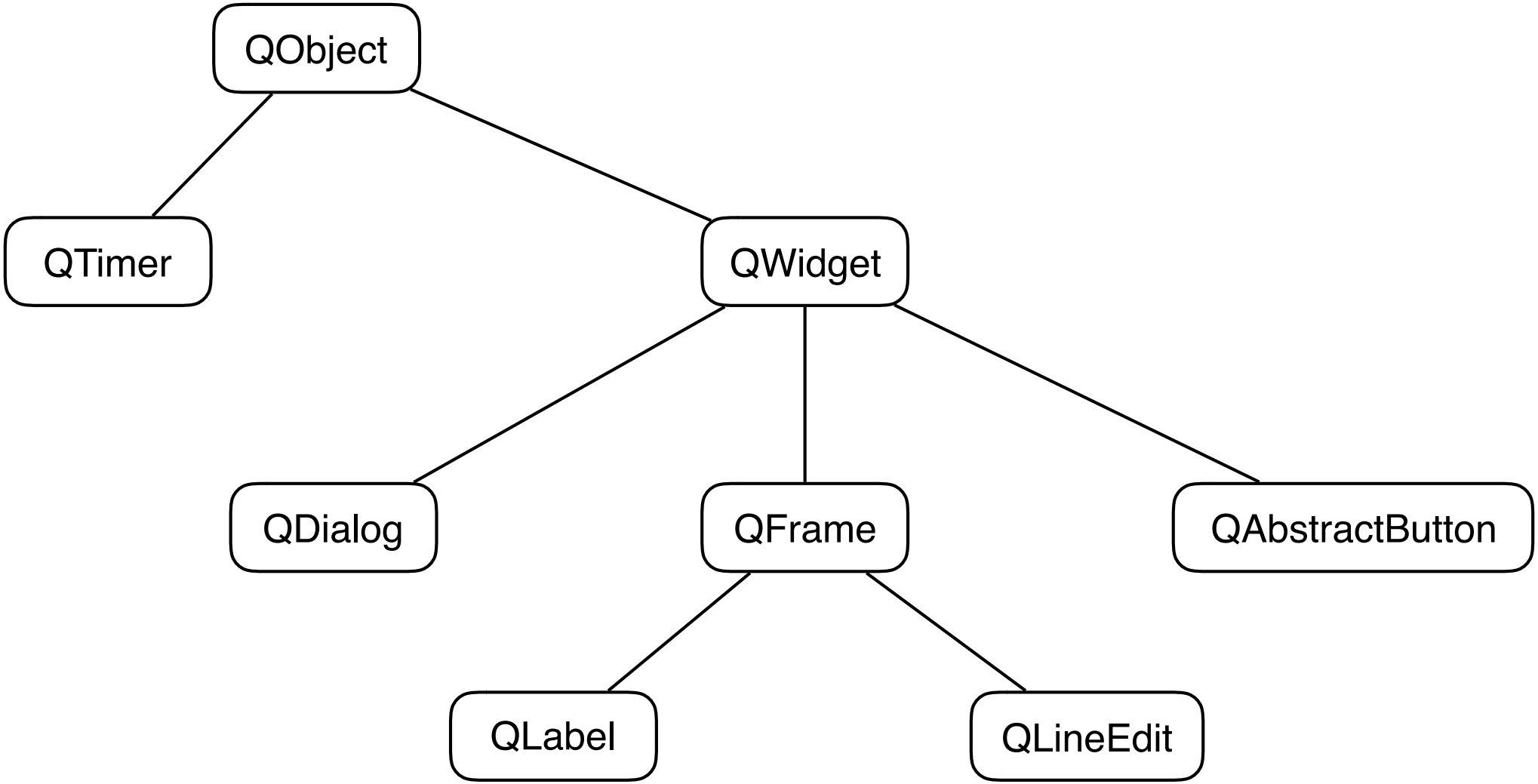


QTreeView



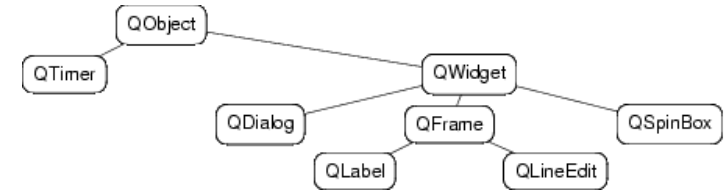
Arbre d’héritage
vs.
arbre d’instanciation

Arbres d'heritage



Principaux widgets

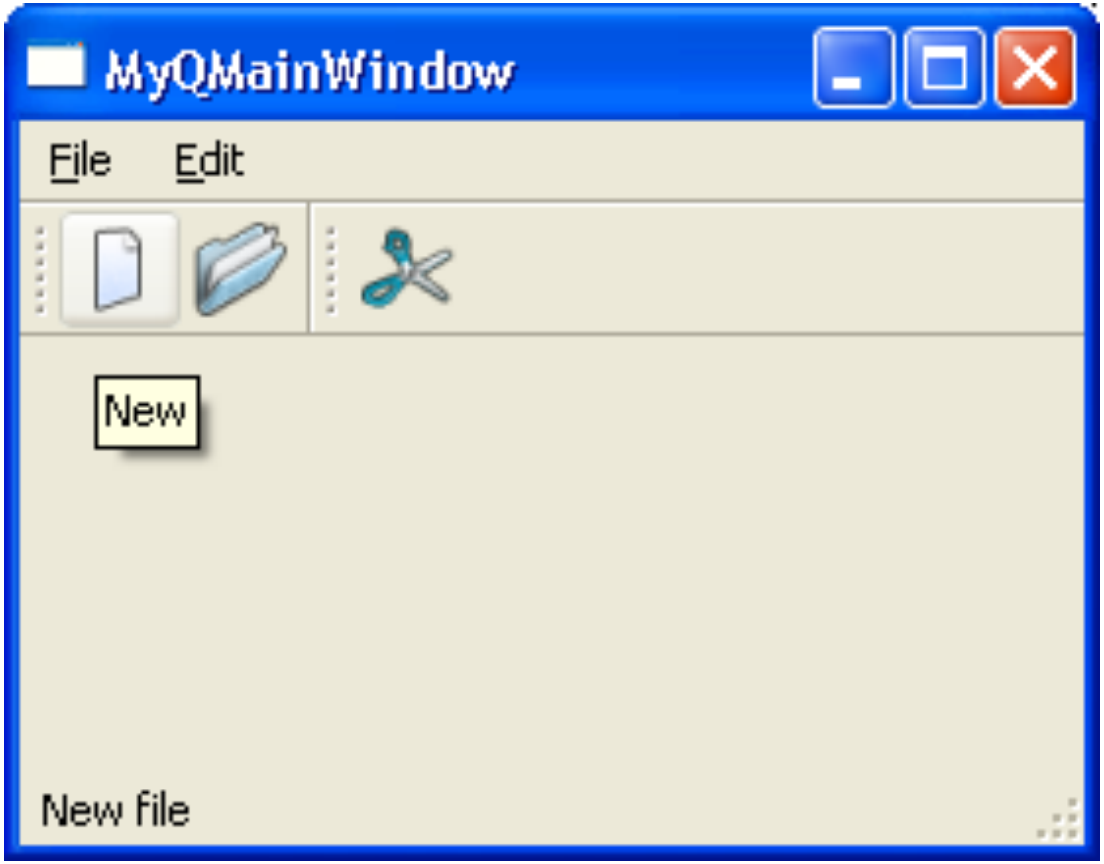
Arbre d'heritage



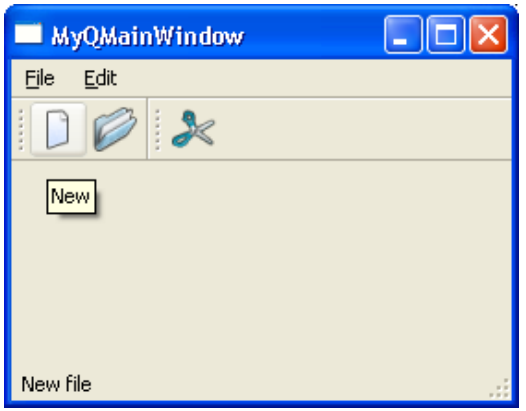
Arbre d'instanciation

Hiérarchie d'instance (=objets)

- Arbre de filiation des objets

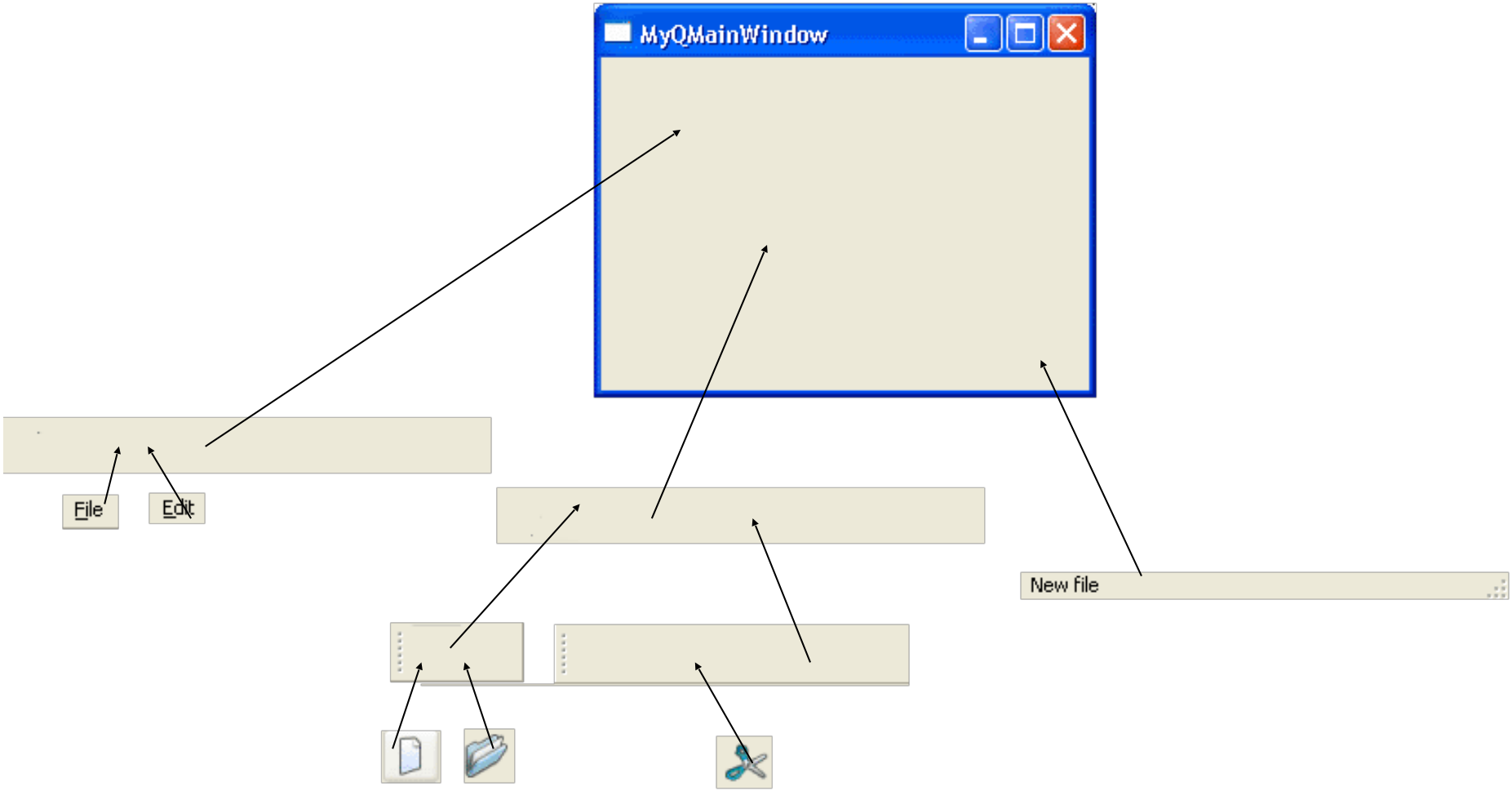


Arbre d'instanciation



Hiérarchie d'instance (=objets)

- Arbre de filiation des objets



Arbre d'instanciation

Les enfants se déclarent auprès de son parent (≠ java)

- label = Qlabel("Hello", parent);
- Exceptions
  - QFile, QApplication…

Si le parent d'un Widget est nul, le Widget est une fenêtre (Window).

Que font les parents ?

- Ils ont une liste des enfants
- Ils détruisent automatiquement les enfants quand ils sont détruits
- Enable/disable les enfants quand ils enable/disable eux memes
- Pareil pour Show/Hide

Arbre d'instanciation

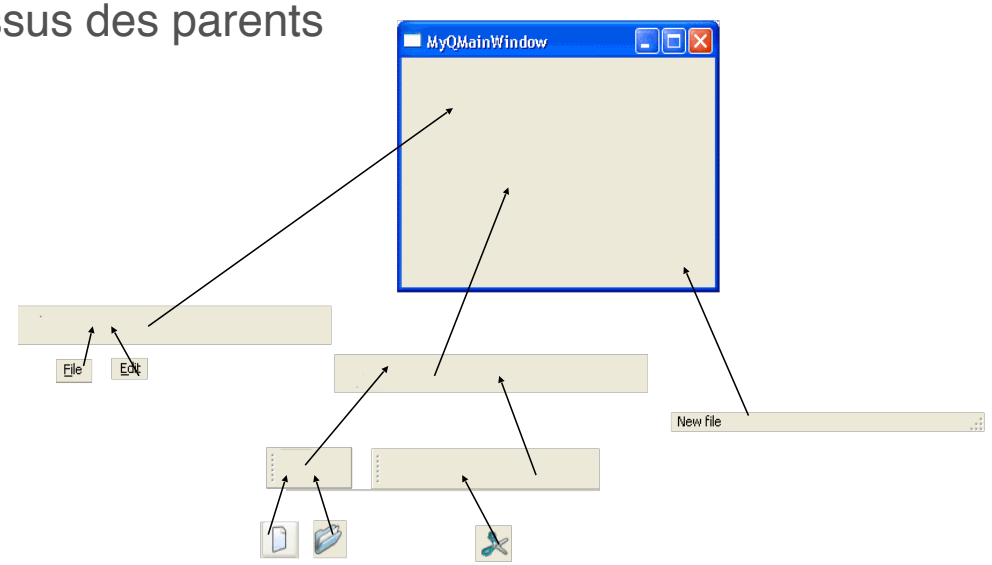
Hiérarchie d'instance (=objets)

- Arbre de filiation des objets

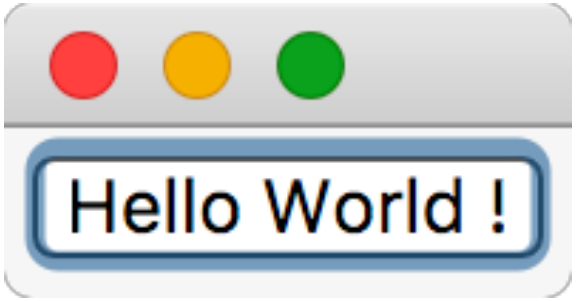
Chaque objet contient ses enfants

- Clipping : enfants inclus dans parents
- Superposition : enfants au dessus des parents

Un objet n'a qu'un seul parent



Simple window




```
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
import sys


def main(args) :
    app = QApplication(args)
    button = QPushButton("Hello World !", None)
    button.resize(100,30)
    button.show()
    app.exec_()


if __name__ == "__main__":
     main(sys.argv)
```

Simple window with button in widget


```
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
import sys


def main(args) :
    app = QApplication(args)
    widget = QWidget(None)
    widget.resize(400,90)
    button = QPushButton("Hello World !", widget)
    button.resize(100,30)
    widget().show()
    app.exec_()


if __name__ == "__main__":
     main(sys.argv)
```

Signaux et slots

Comment, à partir d'un « clic sur un bouton », je peux exécuter la partie correspondant à la logique de mon application ?
(ex: fermer l'application) ??

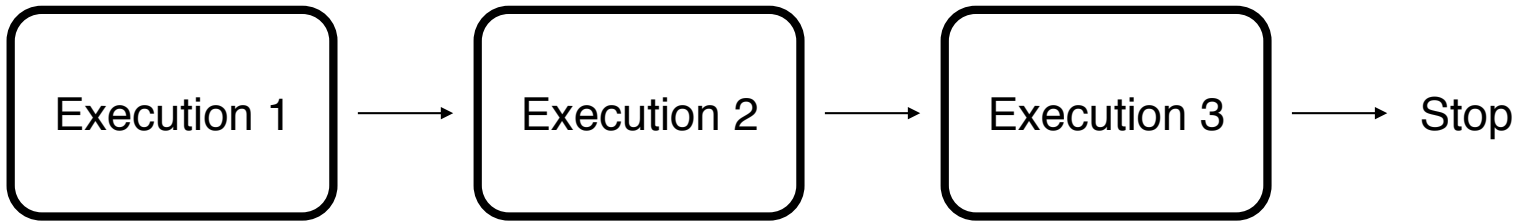
Solutions:

- MFC (introduit un langage au dessus de C++)
- Java (utilise des listeners)
- **Qt (utilise principalement des signaux et slots)**

Application algorithmique

Utilisation de procédures (fonctions) appelées de manière séquentielle

Série d’étapes à réaliser dans un certain ordre

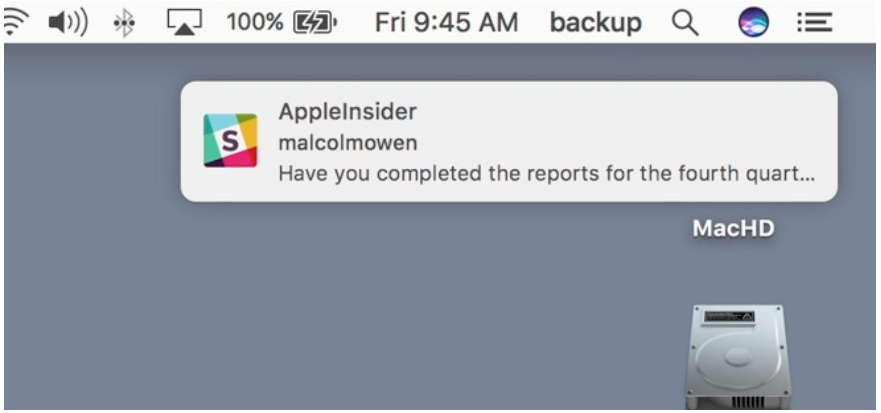


Quels évènements?

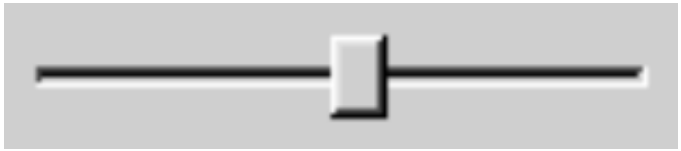
Actions utilisateurs

Notifications de processus (applications, OS, MAJ)

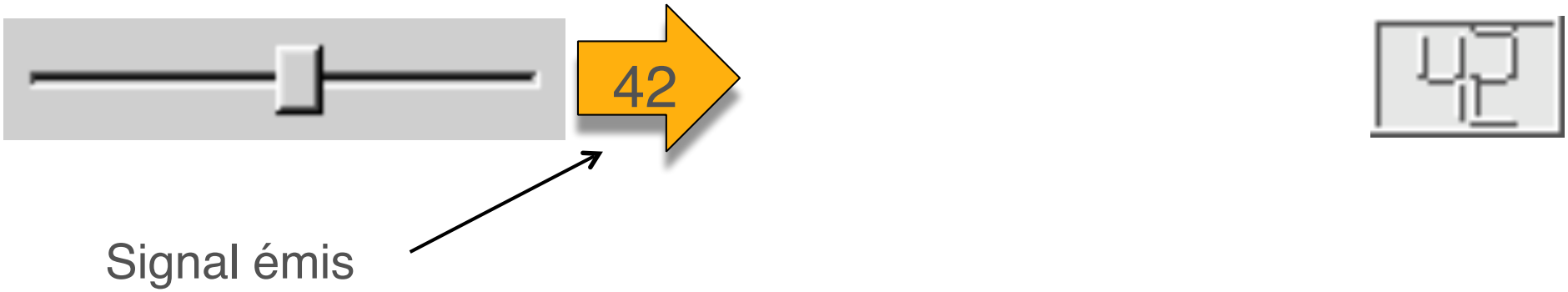
Capteurs sensorielles (info ubiquitaire)



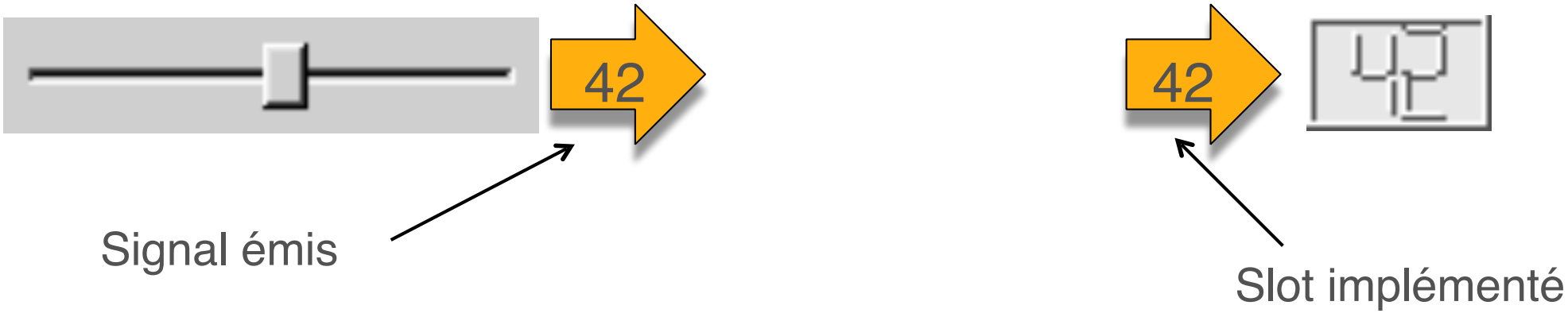
Connecter signaux et slots



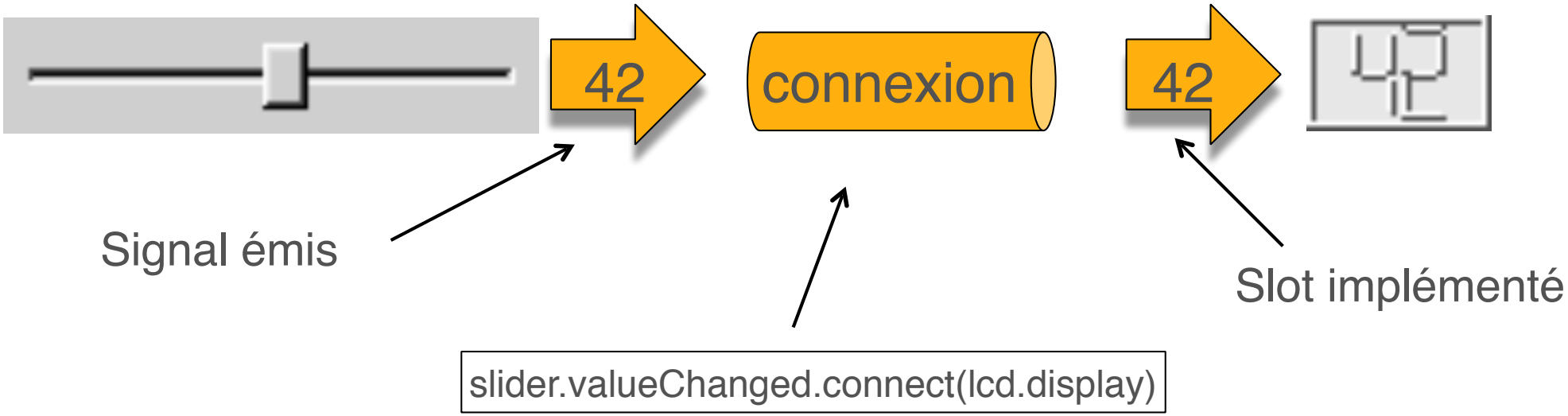
Connecter signaux et slots



Connecter signaux et slots

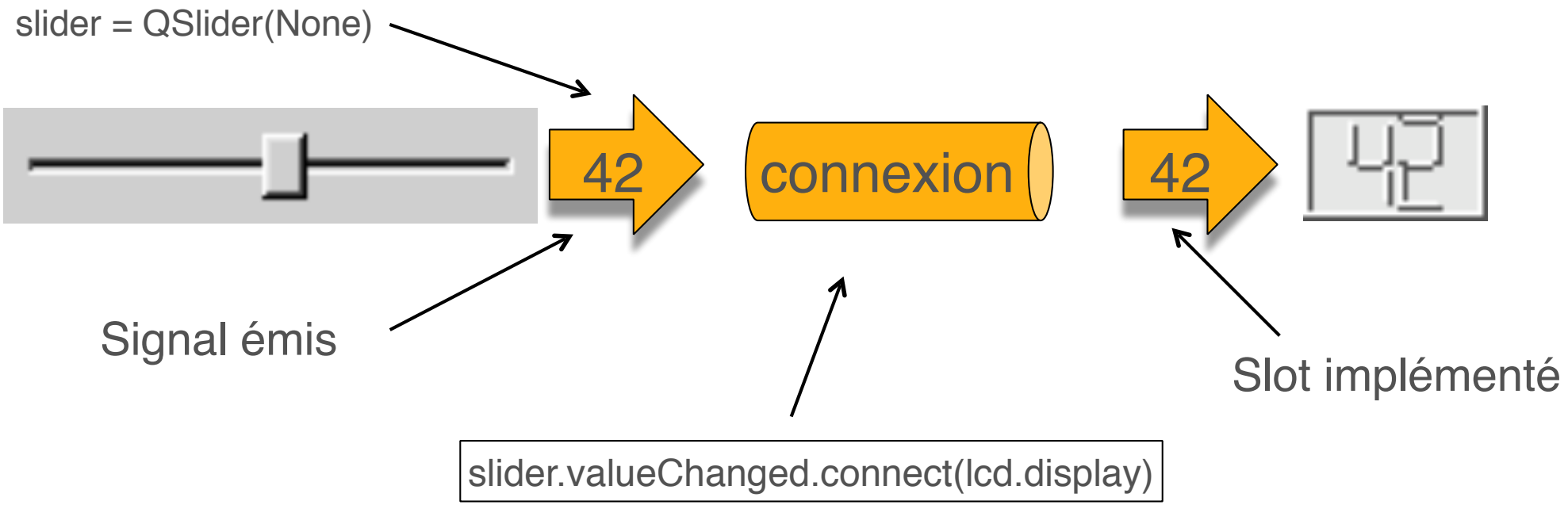


Connecter signaux et slots



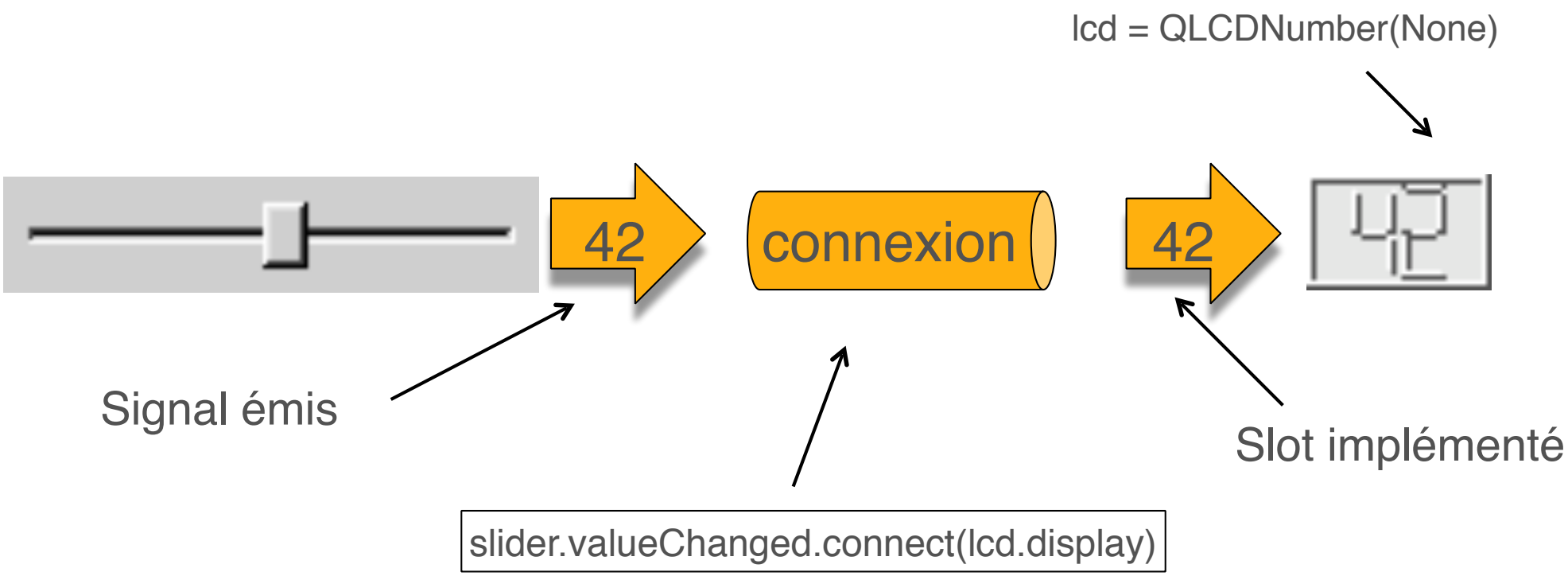
Connecter signaux et slots

```
class QSlider(QObject):
    ….
    def mousePressEvent(self)
        self.valueChanged.emit(value)
        …
```



Connecter signaux et slots

```
class QLCDNumber(QObject):


   def display(num)
      m_value = num;
      …
```



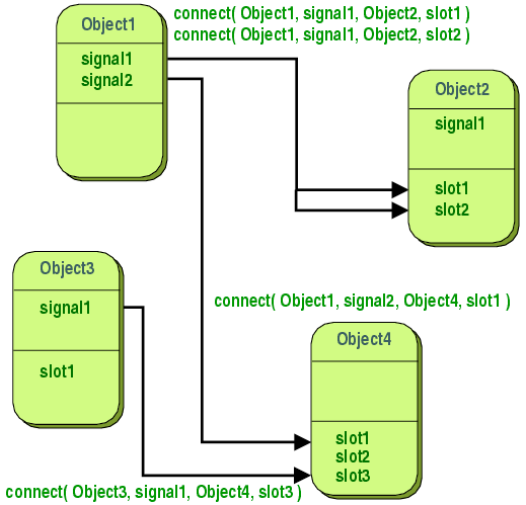
API

Une classe avec des signaux et des slots

```
class MyClass(QObject):

    mySignal = pyqtSignal(int)

    def mySlot( self, num ):
        blabla
```

- Sous class de **QObject**
- Les **signaux** ne sont pas implémentés
- Les **slots** doivent êtres implémentés

Signaux et slots



Modularité, flexibilité

- Connecter **plusieurs** signaux à **un** slot
- Connecter **un** signal à **plusieurs** slots

Philosophie

- L'émetteur n'a pas besoin de connaître le(s) récepteur(s)
- L'émetteur ne sait pas si le signal a été reçu
- Le récepteur ne connaît pas non plus l'émetteur
- Programmation par composant (indépendant, réutilisable)

Sécurité, typage fort

- Les types des paramètres doivent être les mêmes
- Un slot peut avoir **moins** de paramètres qu'un signal

Exemple : transfert d'argent entre banques

```
class BankAccount(QObject):

    balanceChanged = pyqtSignal()

    def __init__(self):
        QObject.__init__(self) # Initialize the PunchingBag as a QObject
        self.balance = 0
        self.name = ""

    @pyqtSlot()
    def setBalance(self, newValue):
        self.balance = newValue
        self.balanceChanged.emit( self.balance)


def main(args):
    account1 = BankAccount()
    account1.setName("account 1")

    account2 = BankAccount()
    account2.setName("account 2")

    # Connect les comptes en banque
    account1.balanceChanged.connect(account2.setBalance)
    account2.balanceChanged.connect(account1.setBalance)

    account1.setBalance(100)

if __name__ == "__main__":
    main(sys.argv)
```

Signal

Slot

Connexion

Exemple : transfert d'argent entre banques

```
@pyqtSlot()
def setBalance(self, newValue):
    if self.balance != newValue:
       self.balance = newValue
      self.balanceChanged.emit( self.balance)
```

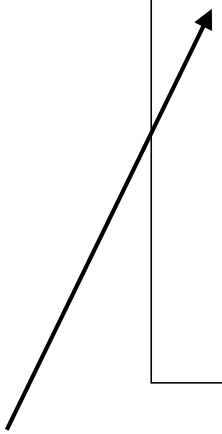
Encore un peu plus loin: auto-connect

```
class MyClass(QObject):

    mySignal = pyqtSignal(int)

    def __init__(self, parent =None):
        super(MyClass, self).__init__(parent)
        self.button = QPushButton("Button 1", self)
        self.button.setObjectName("Button1")

        ….
        QMetaObject.connectSlotsByName(self)

    @pyqtSlot(int)
    def on_button1_clicked( self ):
        blabla
```

def on_<object name>_<signal>(< signal parameters >)

Encore un peu plus loin: pyqtSlot

```
class MyClass(QObject):

    mySignal = pyqtSignal(int)

    def __init__(self, parent =None):
        super(MyClass, self).__init__(parent)
        self.button = QPushButton("Button 1", self)
        self.button.setObjectName("Button1")

        ….
        QMetaObject.connectSlotsByName(self)

    @pyqtSlot(int)
    def on_button1_clicked( self ):
        blabla
```



- auto connect
- performance
  (réduit la mémoire)

Encore un peu plus loin…

connecter / déconnecter

- connect / disconnect

Utilisation des “keyword arguments”

- act = QAction(“Name”, self)
- act.triggered.connect(self.on_triggered)

- act= QAction(“Name”, self, triggered = self.my_slot)

Encore un peu plus loin…

Imaginez que vous avez une grille de 100 boutons. Comment faire?

Solution 1: Obtenir l'émetteur

- def someSlot(self):
  self.sender()

Solution 2: QSignalMapper

- signalMapper = QSignalMapper(self)
- for i, button in enumerate(buttons)
  signalMapper.setMapping(button, **i**)
  button.clicked.connect(signalMapper.map)
- signalMapper.mapped.connect(self.my_slot)

- def my_slot(self, **index**):
  print("button: " , index)

Questions

Comment connecter un signal à un slot ?

- EmetteurObj.<nameSignal>.**connect** (Recepteur.<nameSlot>)

Quel code pour déclarer / implémenter un slot ?

- rien de particulier (mais on peut rajouter @pyqtSlot())

Est ce qu'un slot peut retourner une valeur ?

- Oui

Quel code pour déclarer / implémenter un signal ?

- mySignal = pyqtSignal()

Simple window with button in widget


```
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
import sys


def main(args) :
    app = QApplication(args)
    widget = QWidget(None)
    widget.resize(400,90)
    button = QPushButton("Hello World !", widget)
    button.resize(100,30)
    button.clicked.connect(app.quit)
    widget().show()
    app.exec_()


if __name__ == "__main__":
     main(sys.argv)
```

Les principaux modules

QtCore

QtWidgets

QtGUI

QtBluetooth

QtOpenGL

QtSript/QtScriptTools

QtSql

QtSvg

QtWebKit

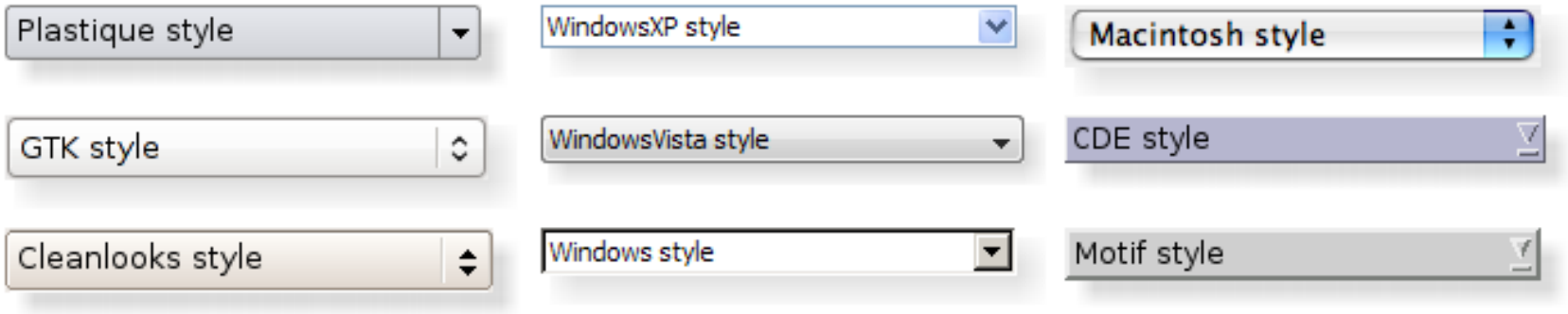
QtXml/QtXmlPatterns

QtMultimedia

QtSensors

QtChart

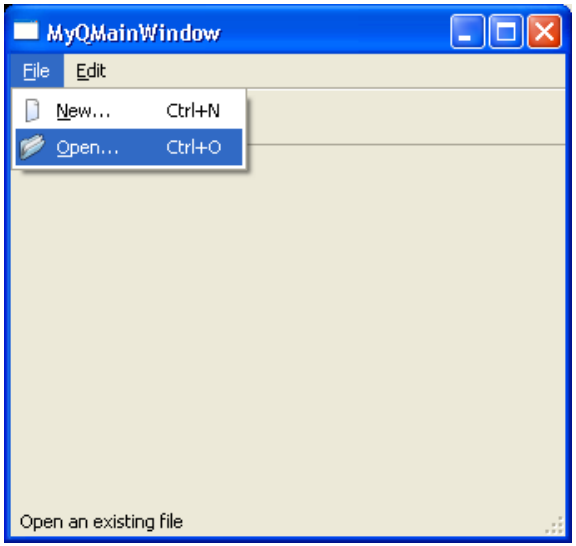
QStyle

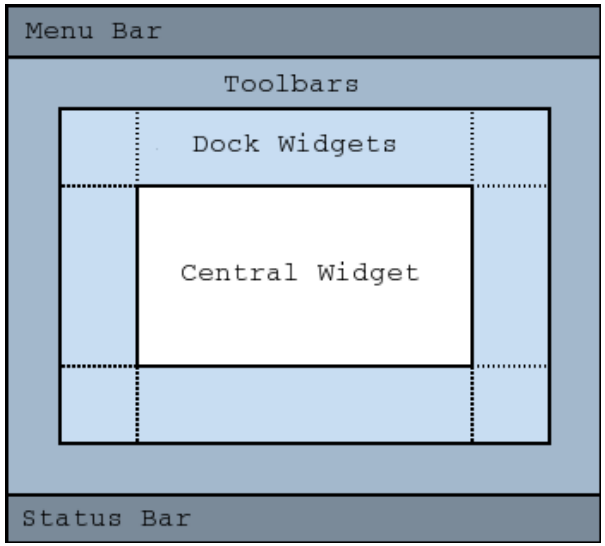


Peut être passé en argument à l'execution du programme

Ex: `python3 test.py -style Windows`

QMainWindow





~~Méthode 1:~~ créer une instance de QMainWindow

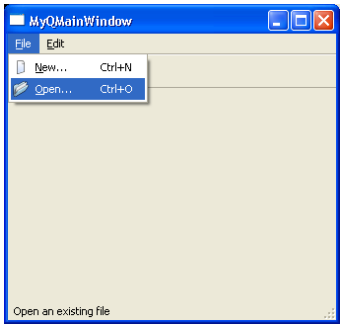
```
win = QMainWindow()
win.resize(200,300)
```

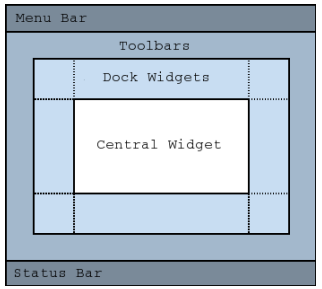
Méthode 2: créer une sous-classe de QMainWindow

```
class Win(QMainWindow):

    def __init__(self):

        self.resize(200,300)
```

QMainWindow

Menus



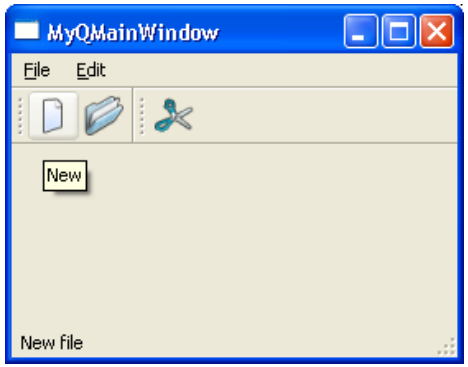



```
bar = self.menuBar()
```

si sous-classe (methode 2)
sinon win.menuBar() (methode 1)

```
fileMenu = bar.addMenu( "File" )

newAct = QAction(QIcon("path/images/new.png"), “New…”, self )
newAct.setShortcut( QKeySequence(“Ctrl+N” ) )
newACT.setToolTip(tr(“New File”))
newAct.setStatusTip(tr(“New file”))

fileMenu.addAction(newAct)

newAct.triggered.connect( self.my_new_slot_method )
```

QMainWindow



QMenuBar, QMenu, QAction

QToolBar

- fileToolBar = QToolBar("File")
- fileToolBar.**addAction**(newAct)
- newAct.**setEnabled**(false)

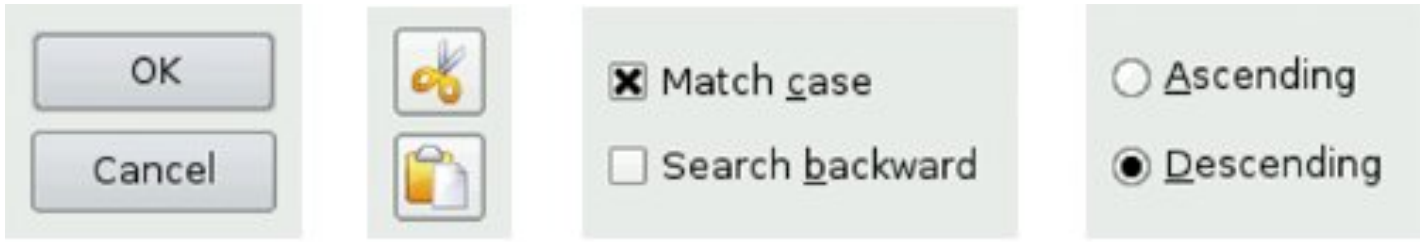
desactive (grise) la commande dans les menus et la toolbar

QToolTip, QWhatsThis

Composant central

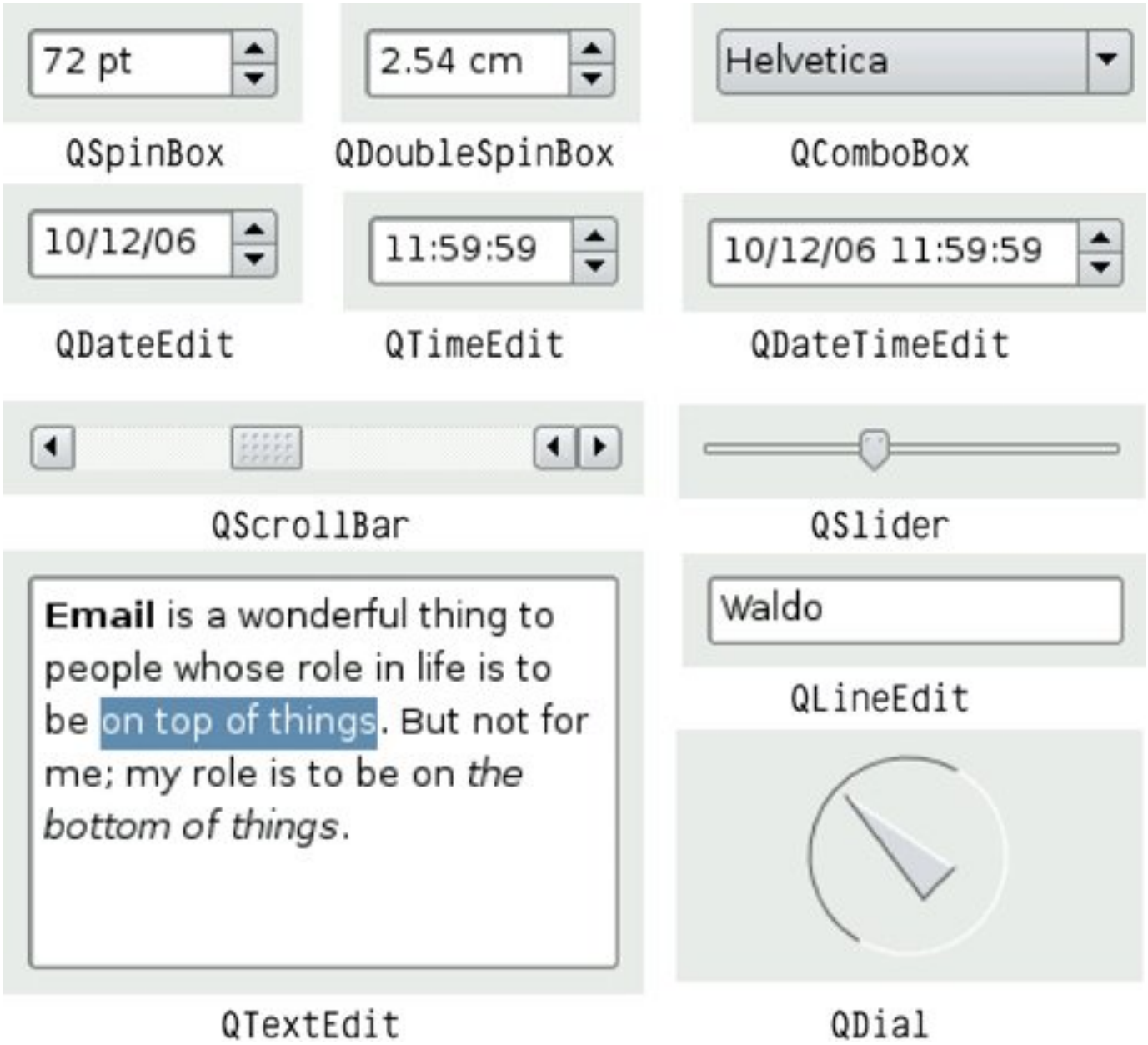
```
textEdit = TextEdit( self );

self.setCentralWidget( textEdit );
```

Buttons

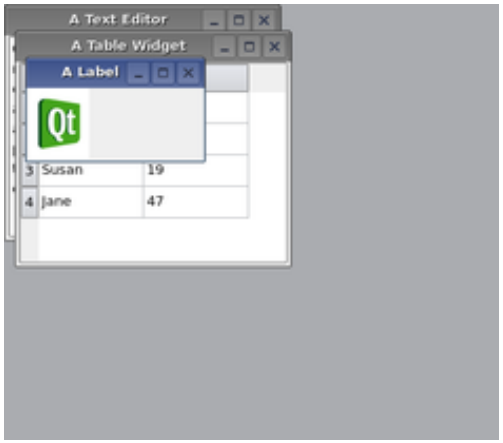


QPushButton QToolButton QCheckBox QRadioButton

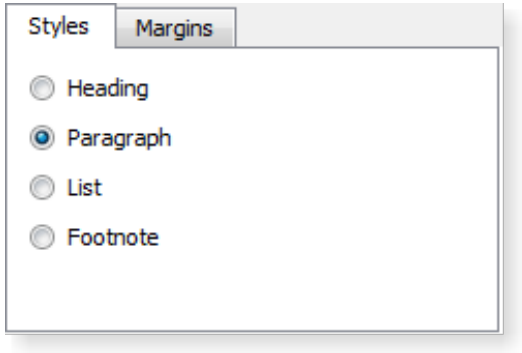
Input Widgets



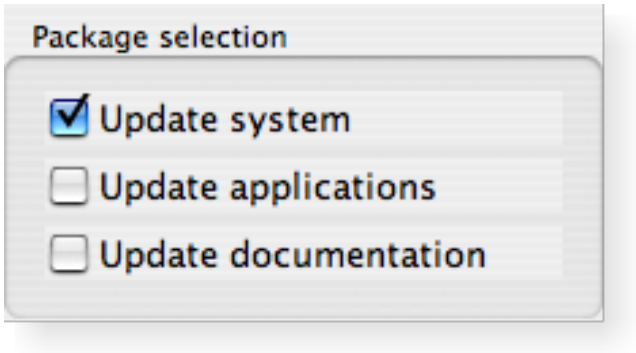
Containers



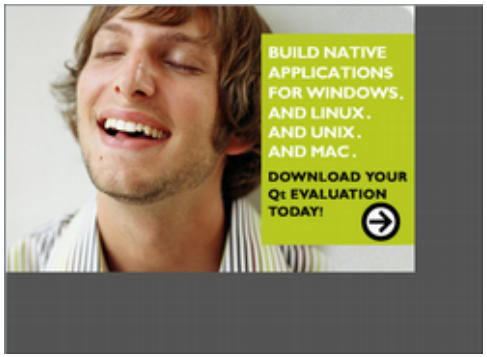
QMidArea



QTabWidget



QGroupBox



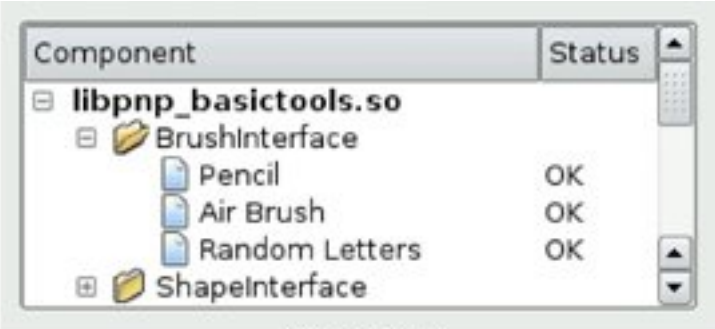
QScrollArea

QToolBox

QWidget; QFrame; QDockWidget; QStackedWidget

Views

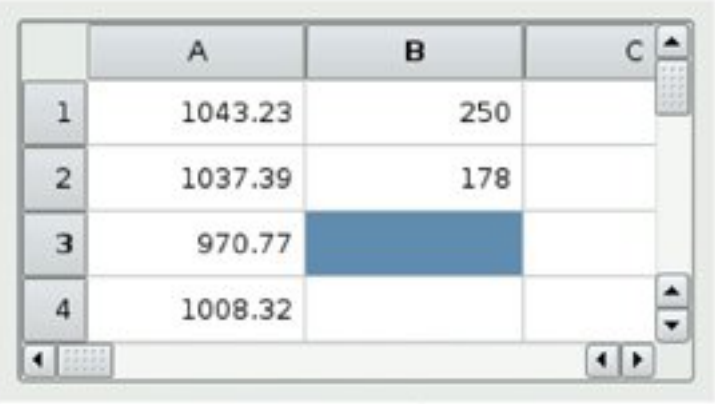
QListView (as list)



QTreeView

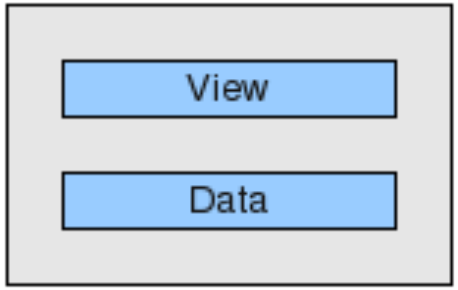
QCompleter

QListView (as icons)

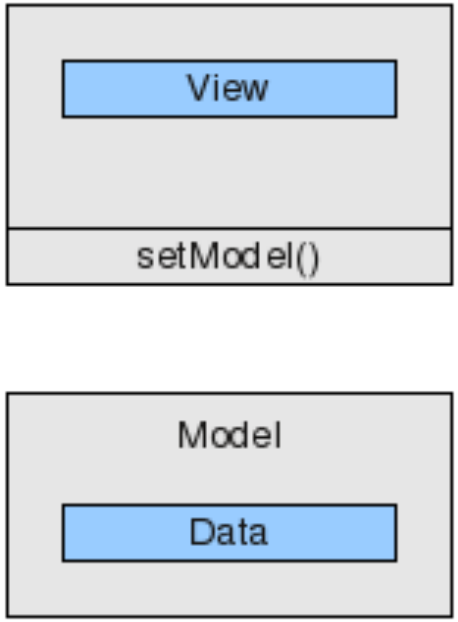


QTableView

standard widgets use data that is part of the widget



View classes operate on external data (the model)

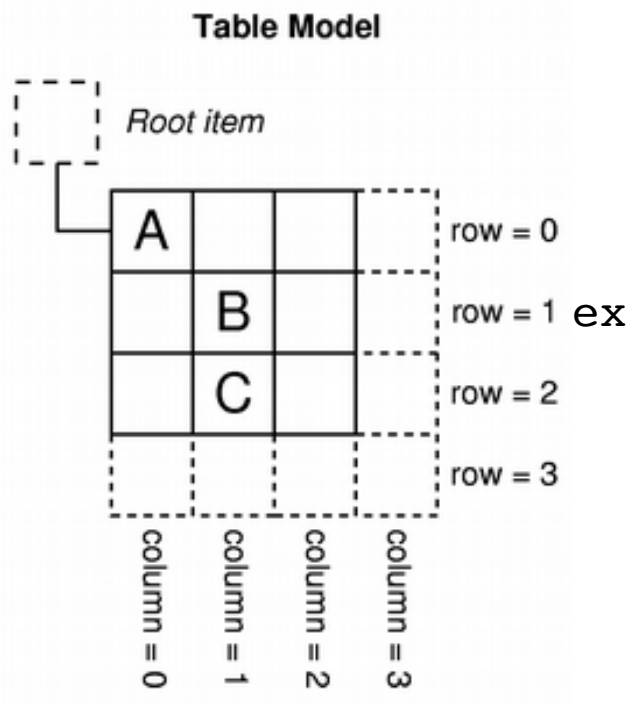


```
def main(args):
   app = QApplication(args)
   tableView = QTableView()
   myModel = MyModel()
   tableView.setModel( myModel )
   tableView.show()
   app.exec()
```

```
class MyModel(QAbstractTableModel):
   def __init__(self):
      QAbstractTableModel.__init__(self)
      self.myData = <dataBase>


   def rowCount( self, parent ):          #Type (parent) == QModelIndex
      return 2

   def columnCount( self, parent ):
      return 2

   def data( self, index, role=Qt.DisplayRole):
      if role == Qt.DisplayRole:
         return self.myData(index.row() + 1, index.column()+1)

   def setData( self, index, value, role):
```

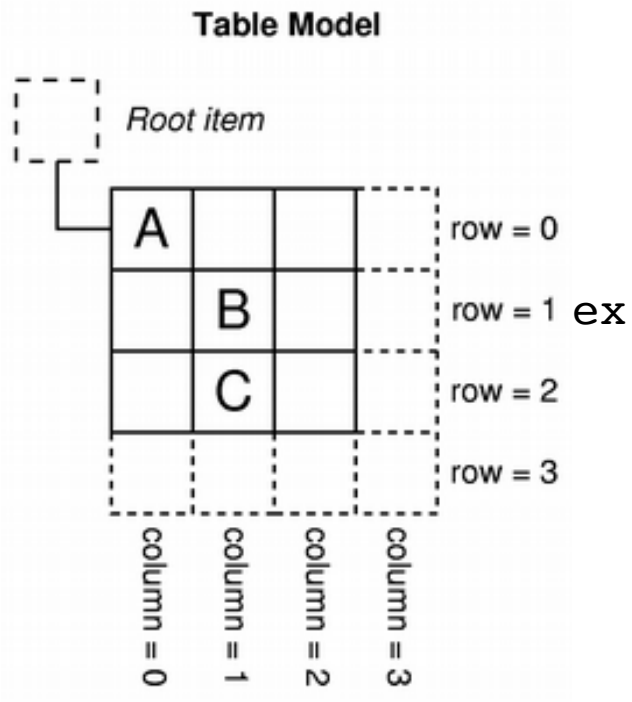
```
def main(args):
   app = QApplication(args)
   tableView = QTableView()
   myModel = MyModel()
   tableView.setModel( myModel )
   tableView.show()
   app.exec()
```

```
class MyModel(QAbstractTableModel):
   def __init__(self):
      QAbstractTableModel.__init__(self)
      self.myData = <dataBase>


   def rowCount( self, parent ):          #Type ex
      return 2

   def columnCount( self, parent ):
      return 2

   def data( self, index, role=Qt.DisplayRole):
      if role == Qt.DisplayRole:
         return self.myData(index.row() + 1, index.column()+1)

   def setData( self, index, value, role):
```




```
def main(args):
   app = QApplication(args)
   tableView = QTableView()
   myModel = MyModel()
   tableView.setModel( myModel )
   tableView.show()
   app.exec()
```

```
class MyModel(QAbstractTableModel):
   def __init__(self):
      QAbstractTableModel.__init__(self)
      self.myData = <dataBase>


   def rowCount( self, parent ):          #Type
      return 2

   def columnCount( self, parent ):
      return 2

   def data( self, index, role=Qt.DisplayRole):
      if role == Qt.DisplayRole:
         return self.myData(index.row() + 1, index.column()+1)

   def setData( self, index, value, role):
```



ex

ne pas oublier:

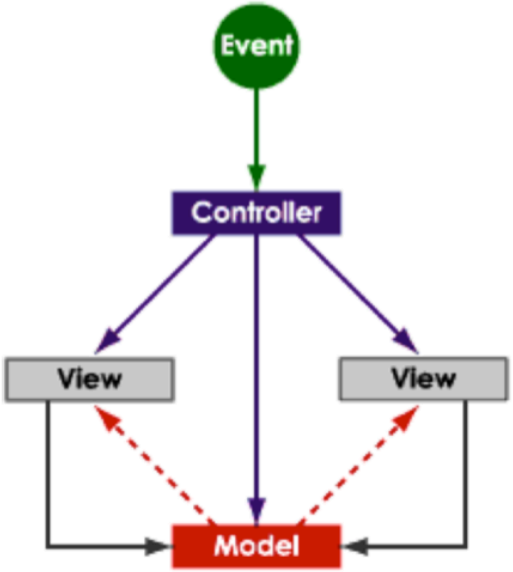
self.valueChanged.emit(index, index)

```
def main(args):
   app = QApplication(args)
   tableView = QTableView()
   myModel = MyModel()
   tableView.setModel( myModel )
   tableView.show()
   app.exec()
```

Model-View-Controller vs. Model/View

Objectifs

- mieux structurer les applications (séparer le noyau fonctionnel du GUI)
- faciliter la synchronisation des représentations multi-vues
- développement simultané
- faciliter la réutilisation du code

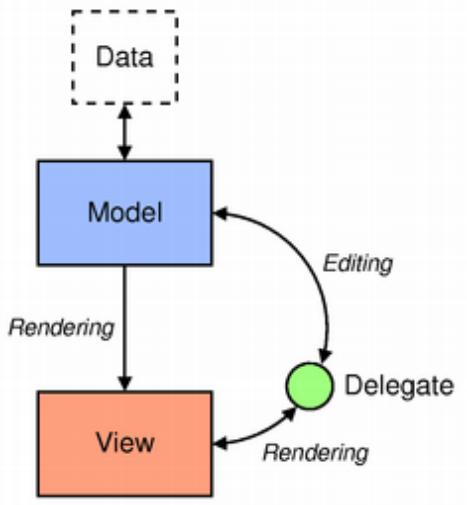


MVC

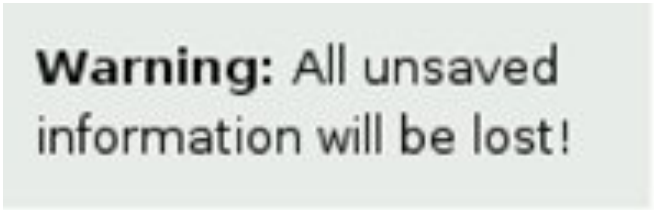
- Model: données et logique du programme
- View: représentation visuelle
- Controller: gestion des entrées et du comportement

Model/View

- Model == Model
- View == View + Controller

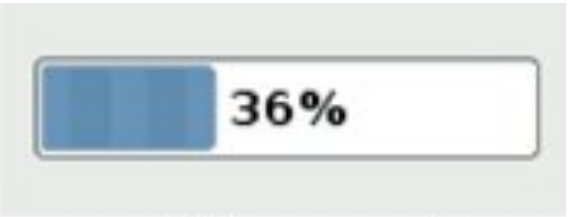


Display Widgets



QLabel (text)

QLCDNumber

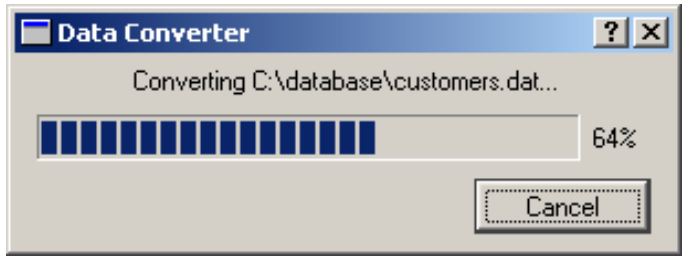


QProgressBar

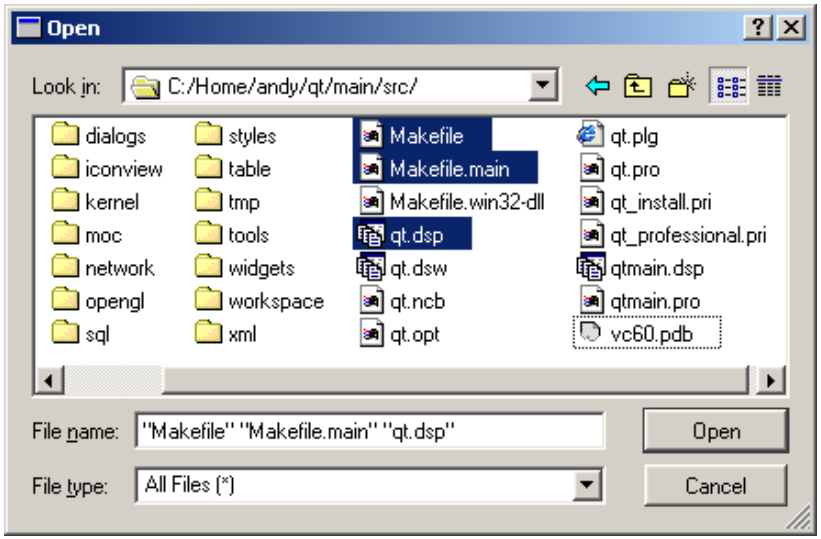
QLabel (image)

QTextBrowser

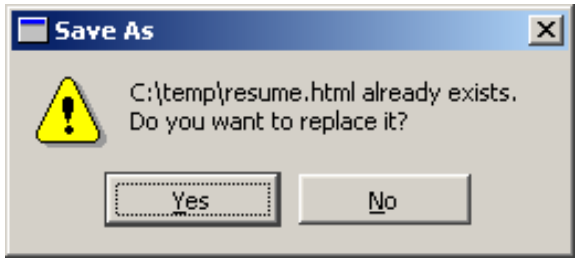
Boites de dialogue



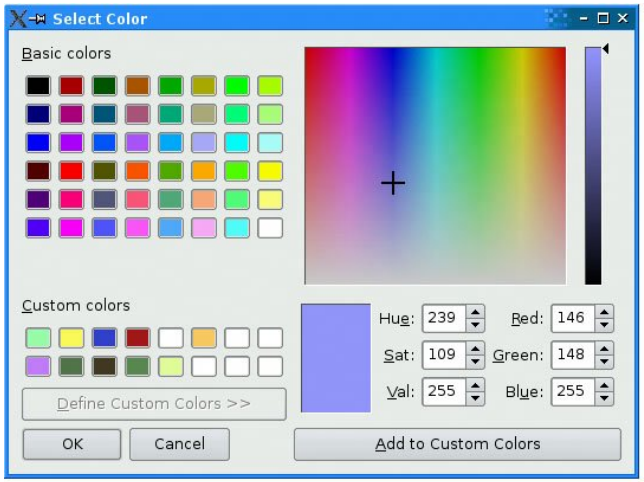
QProgressDialog



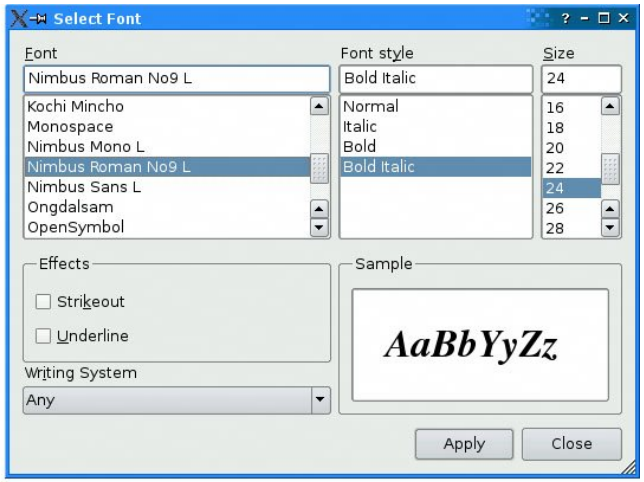
QFileDialog



QMessageBox



QColorDialog

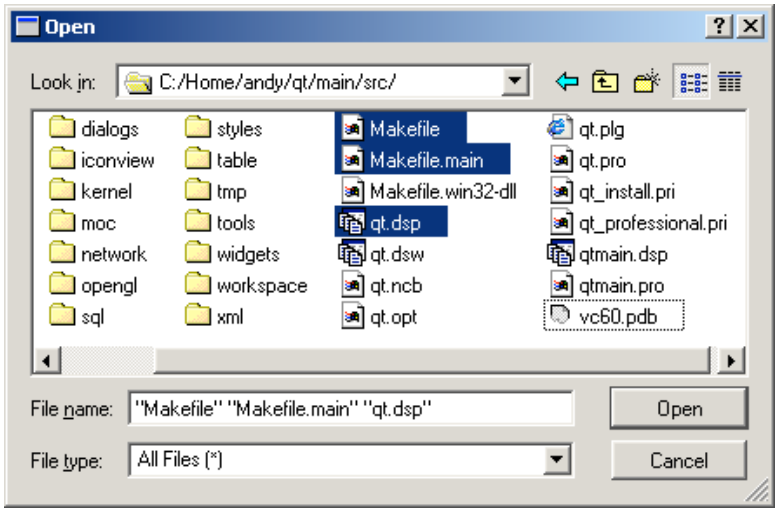


QFontDialog

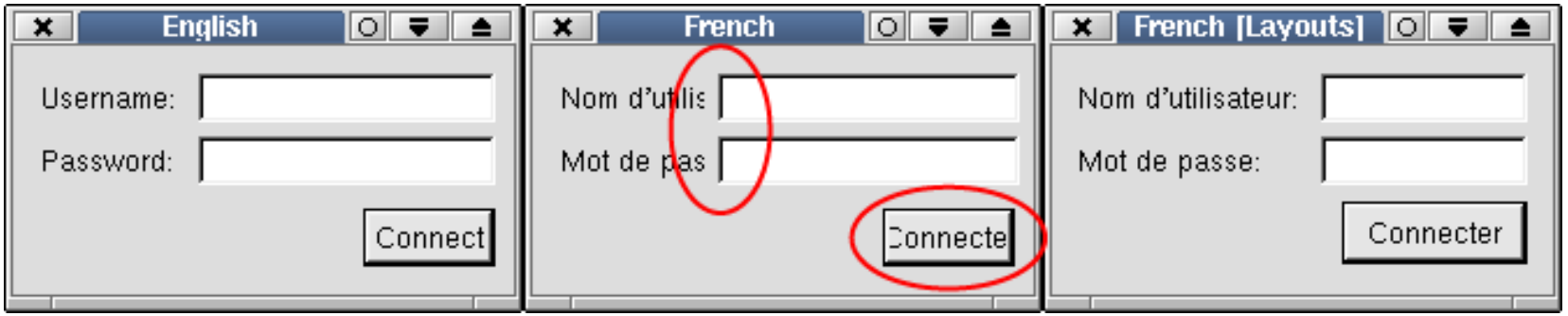
Boîte de dialogue modale

Solution simplifiée

```
fileName =  QFileDialog.getOpenFileName( self,                  //parent
                                         "Open Image",          // titre
                                         "/home/jana",          // répertoire initial
                                         "*.txt")               // filtre
fileName = QFileDialog.getSaveFileName(…)
```



Layout



Problèmes

- internationalisation
- redimensionnement
- complexité du code

Layout

QFormLayout

QHBoxLayout

QVBoxLayout

QGridLayout

Layout

Exemple

```
v_layout = QVBoxLayout( )
v_layout.addWidget(  QPushButton( "OK" )  )
v_layout.addWidget( QPushButton( "Cancel" ) )
v_layout.addStretch( )
v_layout.addWidget( QPushButton( "Help" ) )

country_list = QListBox( );
countryList.insertItem( "Canada" );
...etc...

h_layout = QHBoxLayout( )
h_layout.addWidget( country_list )
h_layout.addLayout( v_layout )

top_layout = QVBoxLayout( )
top_layout.addWidget( QLabel( "Select a country" ) )
top_layout.addLayout( h_layout );

container = QWidget()
container.setLayout( top_layout )
win.setCentralWidget(container)
win.show( )
```

Notes sur layouts :

- peuvent être emboîtés
- pas liés à une hiérarchie de conteneurs comme Java
- cf. le « stretch »

Layout

Exemple

```
v_layout = QVBoxLayout( )
v_layout.addWidget(  QPushButton( "OK" )  )
v_layout.addWidget( QPushButton( "Cancel" ) )
v_layout.addStretch( )
v_layout.addWidget( QPushButton( "Help" ) )
```

```
country_list = QListBox( );
countryList.insertItem( "Canada" );
...etc...

h_layout = QHBoxLayout( )
h_layout.addWidget( country_list )
h_layout.addLayout( v_layout )

top_layout = QVBoxLayout( )
top_layout.addWidget( QLabel( "Select a country" ) )
top_layout.addLayout( h_layout );

container = QWidget()
container.setLayout( top_layout )
win.setCentralWidget(container)
win.show( )
```

Notes sur layouts :

- peuvent être emboîtés
- pas liés à une hiérarchie de conteneurs comme Java
- cf. le « stretch »

Layout

Exemple

```
v_layout = QVBoxLayout( )
v_layout.addWidget(  QPushButton( "OK" )  )
v_layout.addWidget( QPushButton( "Cancel" ) )
v_layout.addStretch( )
v_layout.addWidget( QPushButton( "Help" ) )
```

```
country_list = QListBox( );
countryList.insertItem( "Canada" );
...etc...
```

```
h_layout = QHBoxLayout( )
h_layout.addWidget( country_list )
h_layout.addLayout( v_layout )
```

```
top_layout = QVBoxLayout( )
top_layout.addWidget( QLabel( "Select a country" ) )
top_layout.addLayout( h_layout );

container = QWidget()
container.setLayout( top_layout )
win.setCentralWidget(container)
win.show( )
```

Notes sur layouts :

- peuvent être emboîtés
- pas liés à une hiérarchie de conteneurs comme Java
- cf. le « stretch »

Layout

Exemple

```
v_layout = QVBoxLayout( )
v_layout.addWidget(  QPushButton( "OK" )  )
v_layout.addWidget( QPushButton( "Cancel" ) )
v_layout.addStretch( )
v_layout.addWidget( QPushButton( "Help" ) )
```

```
country_list = QListBox( );
countryList.insertItem( "Canada" );
...etc...
```

```
h_layout = QHBoxLayout( )
h_layout.addWidget( country_list )
h_layout.addLayout( v_layout )
```

```
top_layout = QVBoxLayout( )
top_layout.addWidget( QLabel( "Select a country" ) )
top_layout.addLayout( h_layout );
```

```
container = QWidget()
container.setLayout( top_layout )
win.setCentralWidget(container)
win.show( )
```

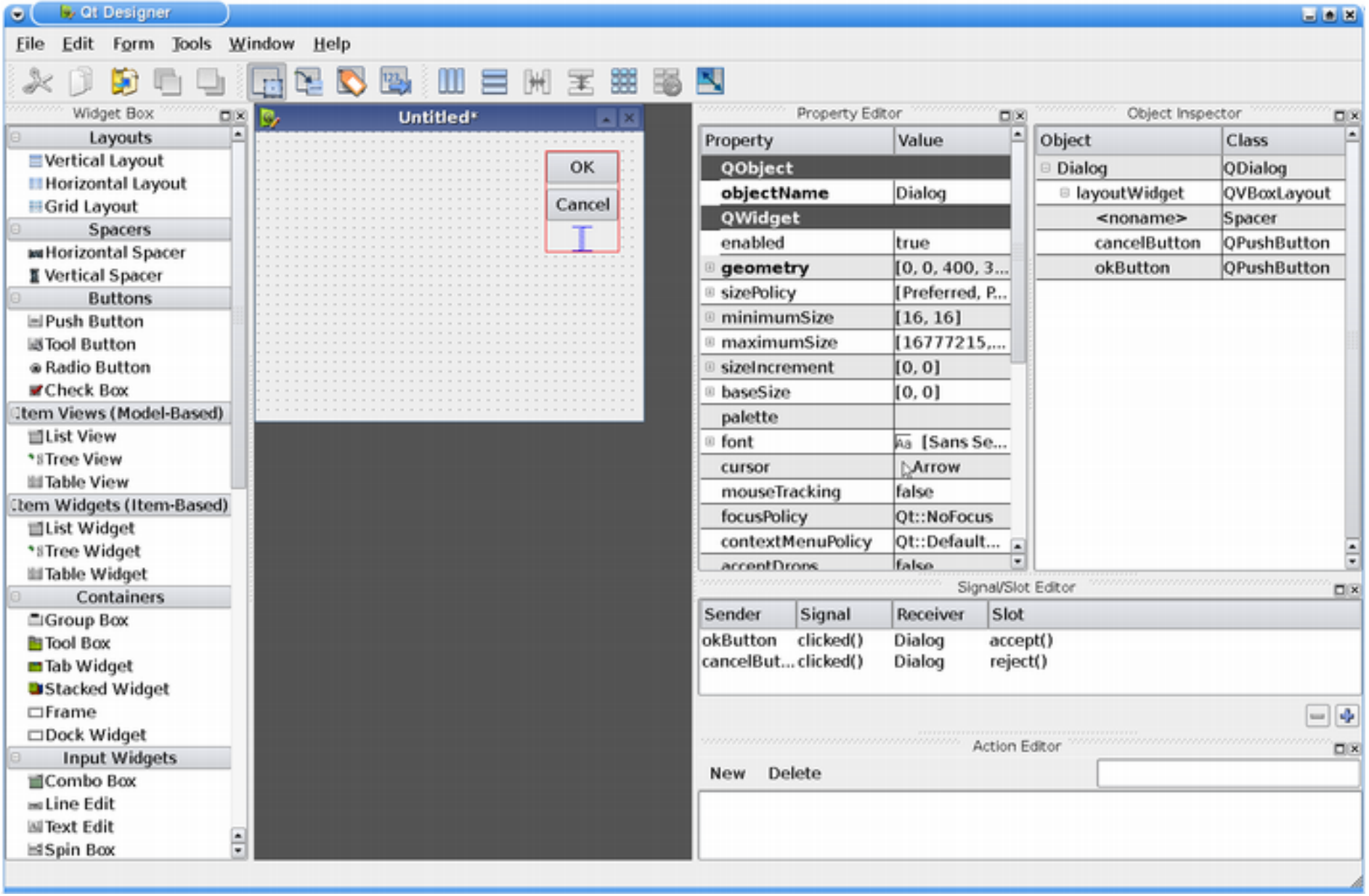
Notes sur layouts :

- peuvent être emboîtés
- pas liés à une hiérarchie de conteneurs comme Java
- cf. le « stretch »

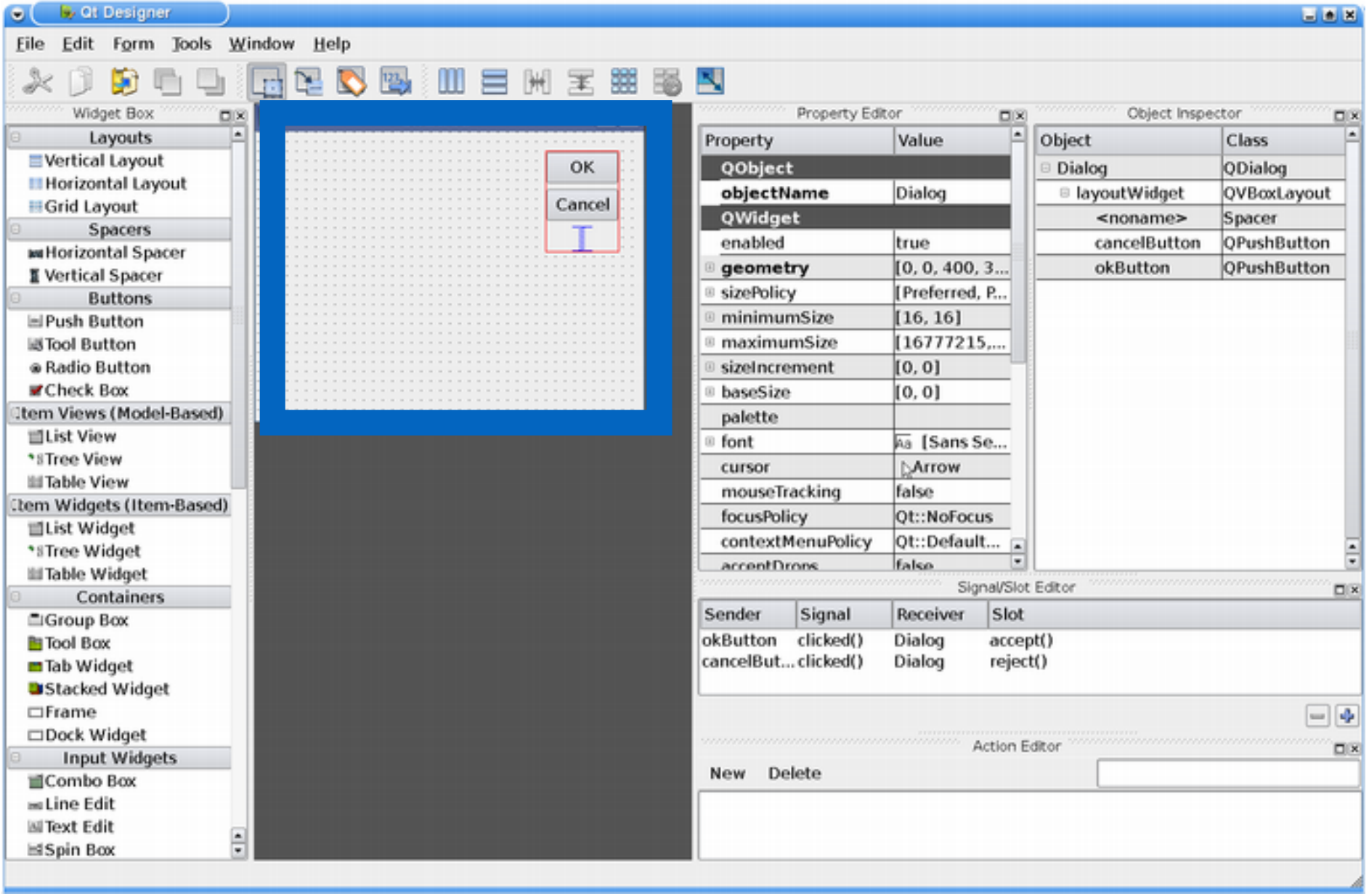
Outils

Subtitle

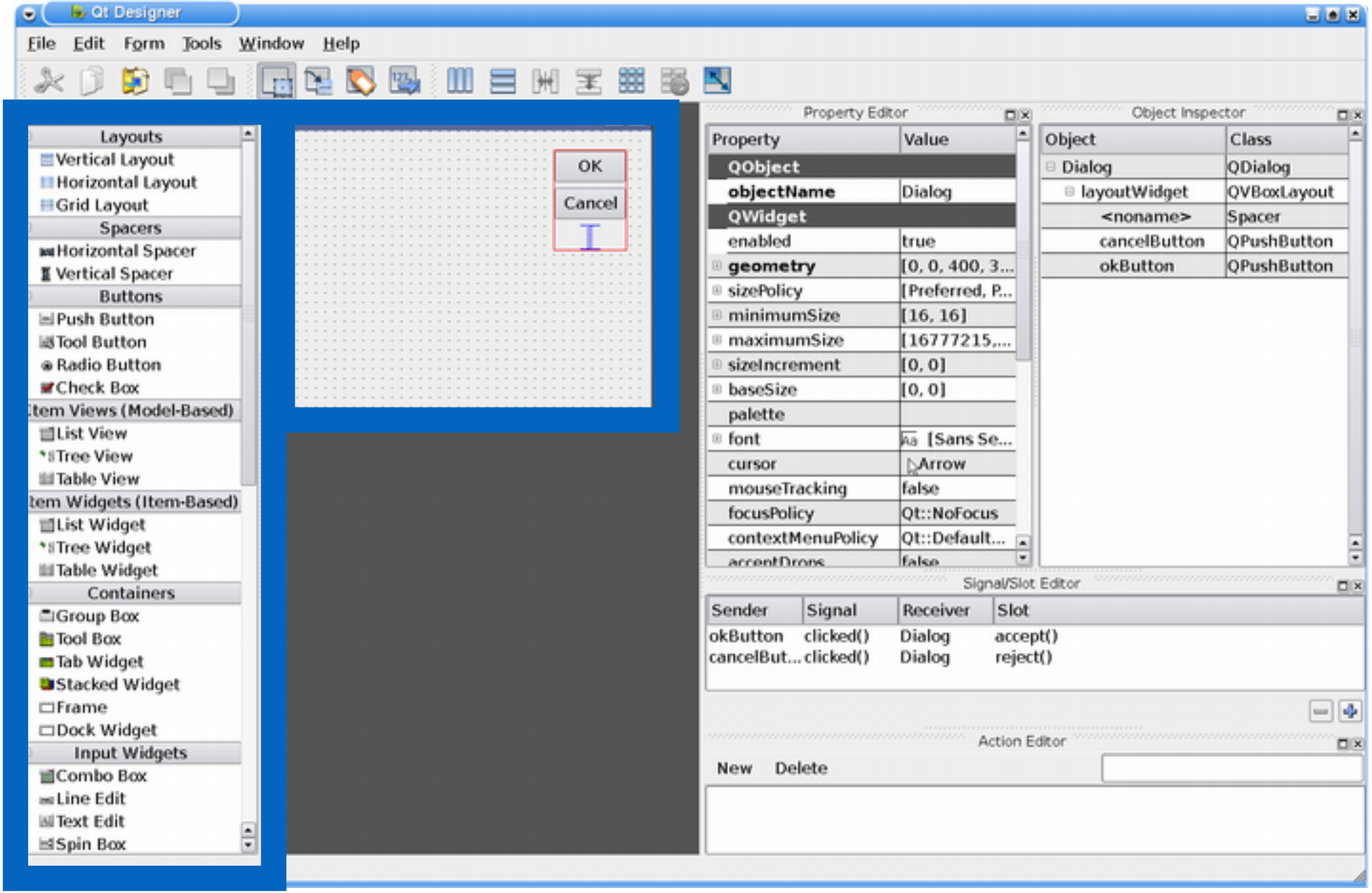
Qt Designer


Qt Designer
File Edit Form Tools Window Help
Widget Box
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Item Views (Model-Based)
List View
Tree View
Table View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Tool Box
Tab Widget
Stacked Widget
Frame
Dock Widget
Input Widgets
Combo Box
Line Edit
Text Edit
Spin Box
Untitled*
OK
Cancel
Property Editor
Property
Value
QObject
objectName
Dialog
QWidget
enabled
true
geometry
[0, 0, 400, 3...
sizePolicy
[Preferred, P...
minimumSize
[16, 16]
maximumSize
[16777215,...
sizeIncrement
[0, 0]
baseSize
[0, 0]
palette
font
[Sans Se...
cursor
Arrow
mouseTracking
false
focusPolicy
Qt::NoFocus
contextMenuPolicy
Qt::Default...
acceptDrops
false
Object Inspector
Object
Class
Dialog
QDialog
layoutWidget
QVBoxLayout
<noname>
Spacer
cancelButton
QPushButton
okButton
QPushButton
Signal/Slot Editor
Sender
Signal
Receiver
Slot
okButton
clicked()
Dialog
accept()
cancelBut...
clicked()
Dialog
reject()
Action Editor
New
Delete

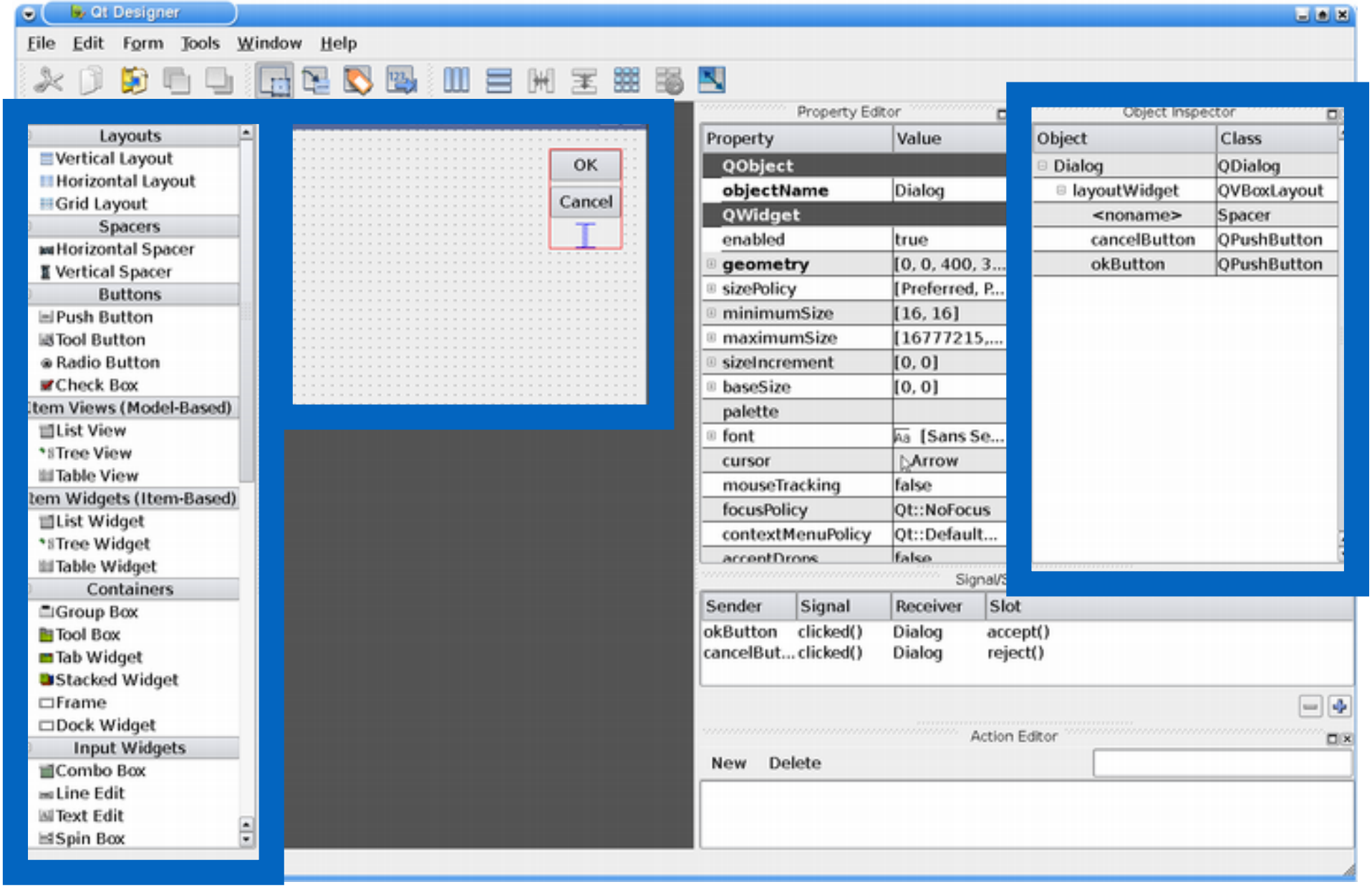
Qt Designer


Qt Designer
File Edit Form Tools Window Help
Widget Box
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Item Views (Model-Based)
List View
Tree View
Table View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Tool Box
Tab Widget
Stacked Widget
Frame
Dock Widget
Input Widgets
Combo Box
Line Edit
Text Edit
Spin Box
OK
Cancel
Property Editor
Property Value
QObject
objectName Dialog
QWidget
enabled true
geometry [0, 0, 400, 3...
sizePolicy [Preferred, P...
minimumSize [16, 16]
maximumSize [16777215,...
sizeIncrement [0, 0]
baseSize [0, 0]
palette
font [Sans Se...
cursor Arrow
mouseTracking false
focusPolicy Qt::NoFocus
contextMenuPolicy Qt::Default...
acceptDrops false
Object Inspector
Object Class
Dialog QDialog
layoutWidget QVBoxLayout
<noname> Spacer
cancelButton QPushButton
okButton QPushButton
Signal/Slot Editor
Sender Signal Receiver Slot
okButton clicked() Dialog accept()
cancelBut... clicked() Dialog reject()
Action Editor
New Delete

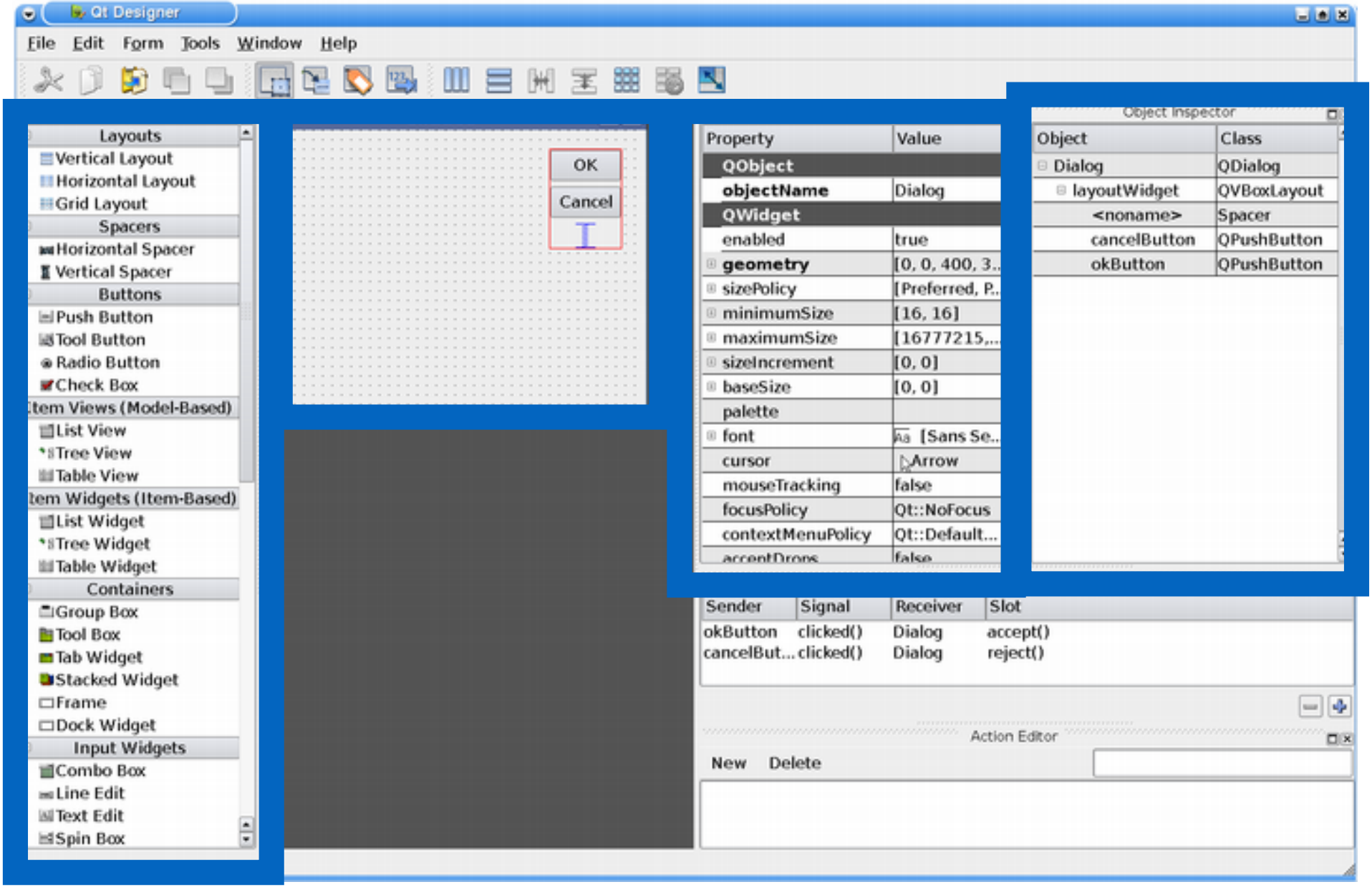
Qt Designer


Qt Designer
File Edit Form Tools Window Help
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Item Views (Model-Based)
List View
Tree View
Table View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Tool Box
Tab Widget
Stacked Widget
Frame
Dock Widget
Input Widgets
Combo Box
Line Edit
Text Edit
Spin Box
OK
Cancel
Property Editor
Property
Value
QObject
objectName
Dialog
QWidget
enabled
true
geometry
[0, 0, 400, 3...
sizePolicy
[Preferred, P...
minimumSize
[16, 16]
maximumSize
[16777215,...
sizeIncrement
[0, 0]
baseSize
[0, 0]
palette
font
[Sans Se...
cursor
Arrow
mouseTracking
false
focusPolicy
Qt::NoFocus
contextMenuPolicy
Qt::Default...
acceptDrops
false
Object Inspector
Object
Class
Dialog
QDialog
layoutWidget
QVBoxLayout
<noname>
Spacer
cancelButton
QPushButton
okButton
QPushButton
Signal/Slot Editor
Sender
Signal
Receiver
Slot
okButton
clicked()
Dialog
accept()
cancelBut...
clicked()
Dialog
reject()
Action Editor
New
Delete

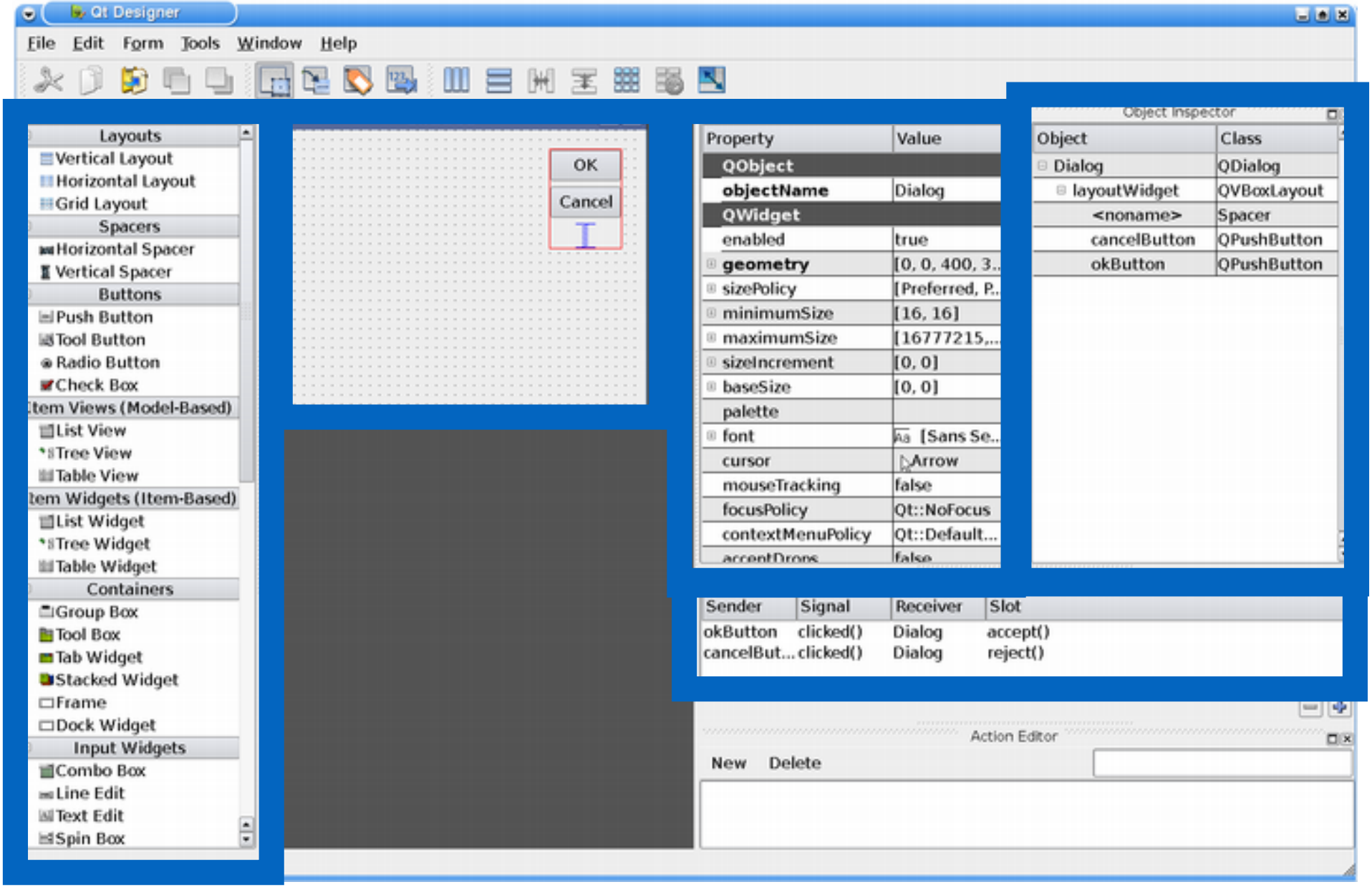
Qt Designer

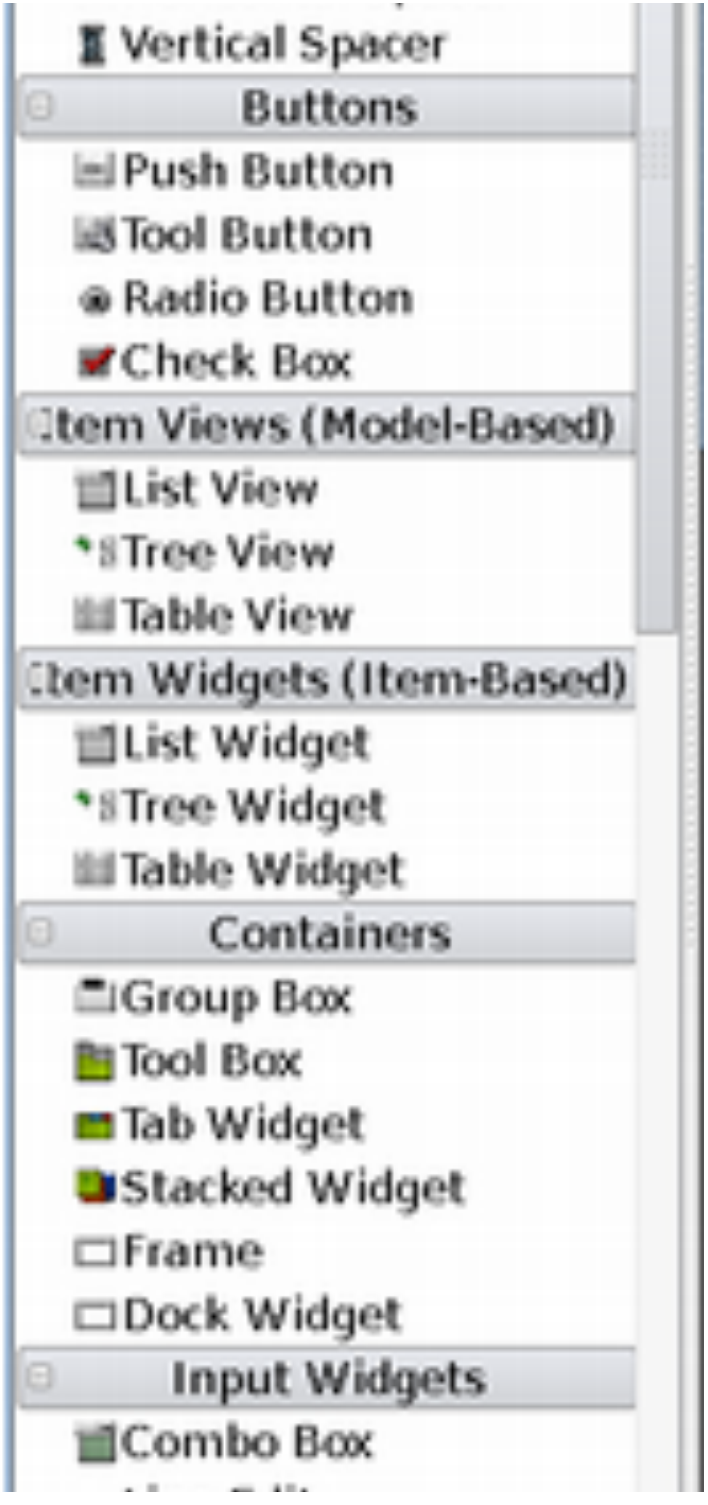

Qt Designer
File Edit Form Tools Window Help
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Item Views (Model-Based)
List View
Tree View
Table View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Tool Box
Tab Widget
Stacked Widget
Frame
Dock Widget
Input Widgets
Combo Box
Line Edit
Text Edit
Spin Box
OK
Cancel
Property Editor
Property
Value
QObject
objectName
Dialog
QWidget
enabled
true
geometry
[0, 0, 400, 3...
sizePolicy
[Preferred, P...
minimumSize
[16, 16]
maximumSize
[16777215,...
sizeIncrement
[0, 0]
baseSize
[0, 0]
palette
font
[Sans Se...
cursor
Arrow
mouseTracking
false
focusPolicy
Qt::NoFocus
contextMenuPolicy
Qt::Default...
Object Inspector
Object
Class
Dialog
QDialog
layoutWidget
QVBoxLayout
<noname>
Spacer
cancelButton
QPushButton
okButton
QPushButton
Sender
Signal
Receiver
Slot
okButton
clicked()
Dialog
accept()
cancelBut...
clicked()
Dialog
reject()
Action Editor
New
Delete

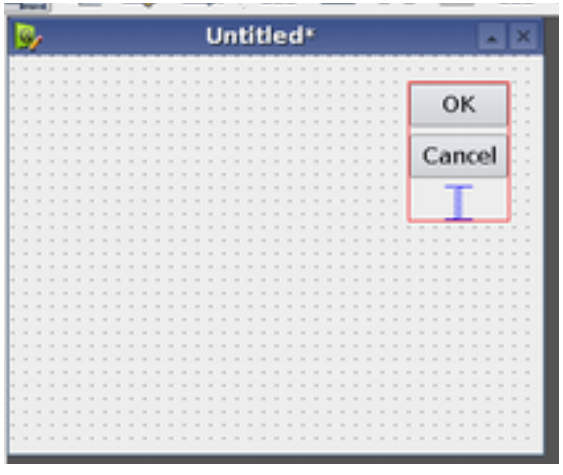
Qt Designer


Qt Designer
File Edit Form Tools Window Help
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Item Views (Model-Based)
List View
Tree View
Table View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Tool Box
Tab Widget
Stacked Widget
Frame
Dock Widget
Input Widgets
Combo Box
Line Edit
Text Edit
Spin Box
OK
Cancel
Property
Value
QObject
objectName
Dialog
QWidget
enabled
true
geometry
[0, 0, 400, 3..
sizePolicy
[Preferred, P..
minimumSize
[16, 16]
maximumSize
[16777215,..
sizeIncrement
[0, 0]
baseSize
[0, 0]
palette
font
[Sans Se..
cursor
Arrow
mouseTracking
false
focusPolicy
Qt::NoFocus
contextMenuPolicy
Qt::Default...
Object Inspector
Object
Class
Dialog
QDialog
layoutWidget
QVBoxLayout
<noname>
Spacer
cancelButton
QPushButton
okButton
QPushButton
Sender
Signal
Receiver
Slot
okButton
clicked()
Dialog
accept()
cancelBut...
clicked()
Dialog
reject()
Action Editor
New
Delete

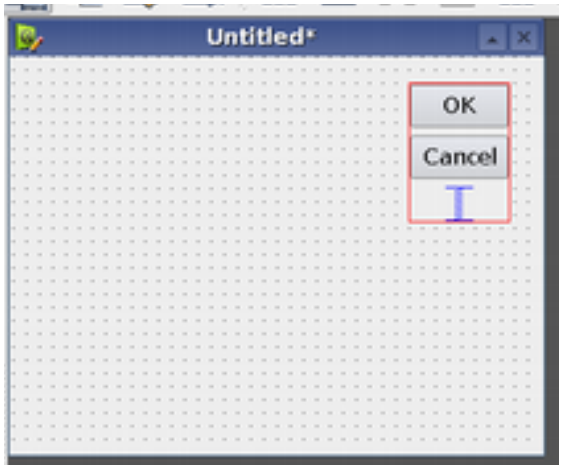
Qt Designer


Qt Designer
File Edit Form Tools Window Help
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Item Views (Model-Based)
List View
Tree View
Table View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Tool Box
Tab Widget
Stacked Widget
Frame
Dock Widget
Input Widgets
Combo Box
Line Edit
Text Edit
Spin Box
OK
Cancel
Property Value
QObject
objectName Dialog
QWidget
enabled true
geometry [0, 0, 400, 3..
sizePolicy [Preferred, P..
minimumSize [16, 16]
maximumSize [16777215,..
sizeIncrement [0, 0]
baseSize [0, 0]
palette
font [Sans Se..
cursor Arrow
mouseTracking false
focusPolicy Qt::NoFocus
contextMenuPolicy Qt::Default...
acceptDrops false
Object Inspector
Object Class
Dialog QDialog
layoutWidget QVBoxLayout
<noname> Spacer
cancelButton QPushButton
okButton QPushButton
Sender Signal Receiver Slot
okButton clicked() Dialog accept()
cancelBut... clicked() Dialog reject()
Action Editor
New Delete


Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
tem Views (Model-Based)
List View
Tree View
Table View
tem Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Tool Box
Tab Widget
Stacked Widget
Frame
Dock Widget
Input Widgets
Combo Box

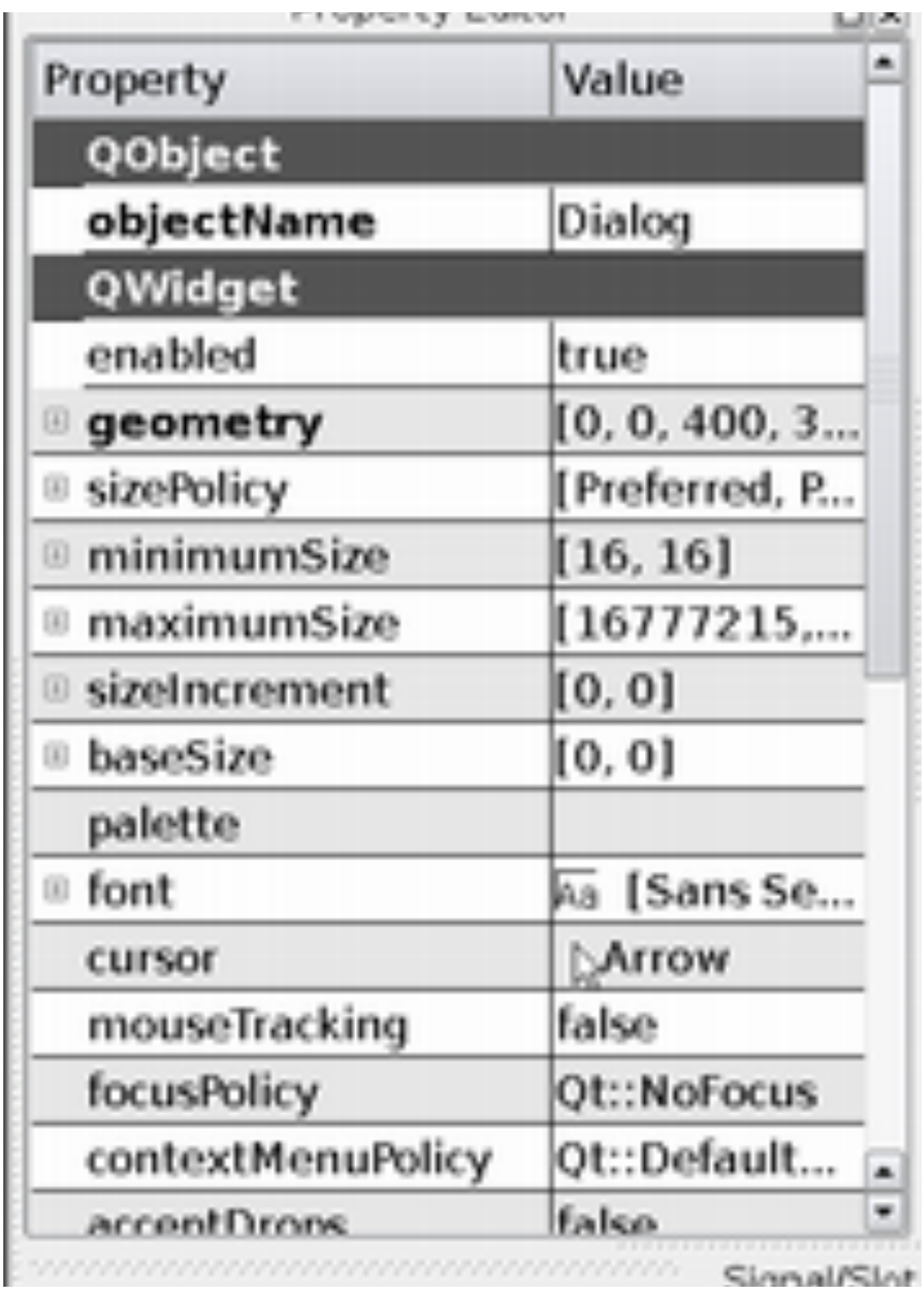

Untitled*
OK
Cancel

Qt Designer

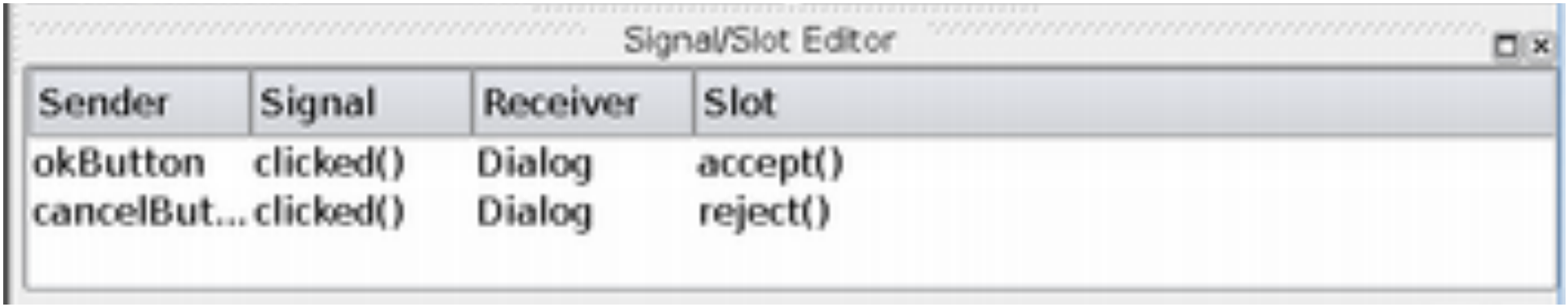


Object	Class
Dialog	QDialog
layoutWidget	QVBoxLayout
<noname>	Spacer
cancelButton	QPushButton
okButton	QPushButton

Subtitle


Property
Value
QObject
objectName
Dialog
QWidget
enabled
true
geometry
[0, 0, 400, 3...
sizePolicy
[Preferred, P...
minimumSize
[16, 16]
maximumSize
[16777215,...
sizeIncrement
[0, 0]
baseSize
[0, 0]
palette
font
[Sans Se...
cursor
Arrow
mouseTracking
false
focusPolicy
Qt::NoFocus
contextMenuPolicy
Qt::Default...
acceptDrops
false

Subtitle


Signal/Slot Editor
Sender
Signal
Receiver
Slot
okButton
clicked()
Dialog
accept()
cancelBut...
clicked()
Dialog
reject()

Qt Designer

Sauvegarder le projet en **ui_imagedialog.ui** ← fichier xml

transformer en fichier python

pyuic5 ui_imagedialog.ui > ui_imagedialog.py

Insérer la boite de dialogue dans votre code

Approche simple

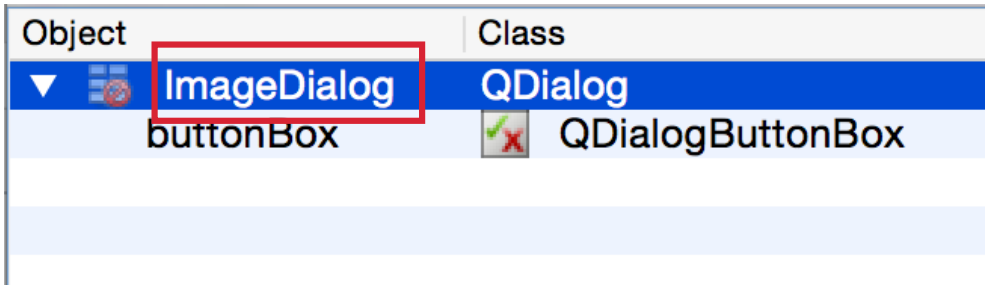
pyuic5 ui_imagedialog.ui > ui_imagedialog.py

```
import sys
from PyQt5.QtWidgets import QApplication, QDialog
from ui_imagedialog import Ui_ImageDialog

app = QApplication(sys.argv)
window = QDialog()
ui = Ui_ImageDialog()
ui.setupUi(window)

window.show()
app.exec_()
```

Approche simple

pyuic5 ui_imagedialog.ui > ui_imagedialog.py

```
import sys
from PyQt5.QtWidgets import QApplication, QDialog
from ui_imagedialog import Ui_ImageDialog

app = QApplication(sys.argv)
window = QDialog()
ui = Ui_ImageDialog()
ui.setupUi(window)

window.show()
app.exec_()
```

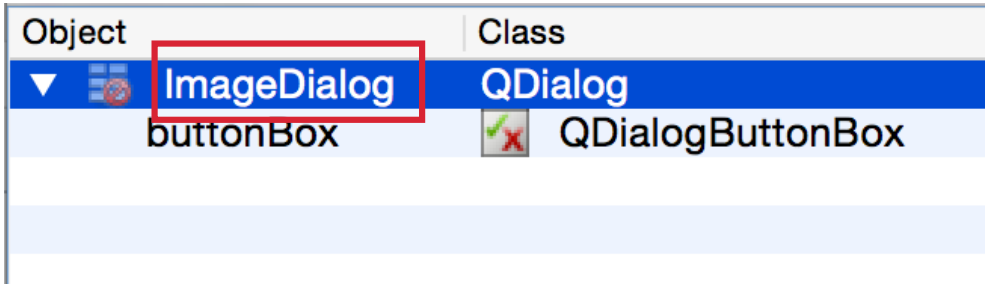


Approche simple

ui_imagedialog.py

pyuic5 ui_imagedialog.ui > ui_imagedialog.py

```
import sys
from PyQt5.QtWidgets import QApplication, QDialog
from ui_imagedialog import Ui_ImageDialog

app = QApplication(sys.argv)
window = QDialog()
ui = Ui_ImageDialog()
ui.setupUi(window)

window.show()
app.exec_()
```



Héritage simple

```
from PyQt5.QtWidgets import QDialog
from ui_imagedialog import Ui_ImageDialog

class ImageDialog(QDialog):
    def __init__(self):
        super(ImageDialog, self).__init__()

        # Set up the user interface from Designer.
        self.ui = Ui_ImageDialog()
        self.ui.setupUi(self)

        # Make some local modifications.
        self.ui.colorDepthCombo.addItem("2 colors (1 bit per
pixel)")

        # Connect up the buttons.
        self.ui.okButton.clicked.connect(self.accept)
        self.ui.cancelButton.clicked.connect(self.reject)
```

Héritage multiple

```
from PyQt5.QtWidgets import QDialog
from ui_imagedialog import Ui_ImageDialog

class ImageDialog(QDialog, Ui_ImageDialog):
    def __init__(self):
        super(ImageDialog, self).__init__()

        # Set up the user interface from Designer.
        self.setupUi(self)

        # Make some local modifications.
        self.colorDepthCombo.addItem("2 colors (1 bit
per pixel)")

        # Connect up the buttons.
        self.okButton.clicked.connect(self.accept)
        self.cancelButton.clicked.connect(self.reject)
```

The Qt Resource System

Mécanisme pour charger des fichiers binaires dans l'exécutable de l'application. Utile si

- l'application a besoin de certains fichiers (icons, fichiers de traduction)
- vous ne voulez pas perdre ces fichiers

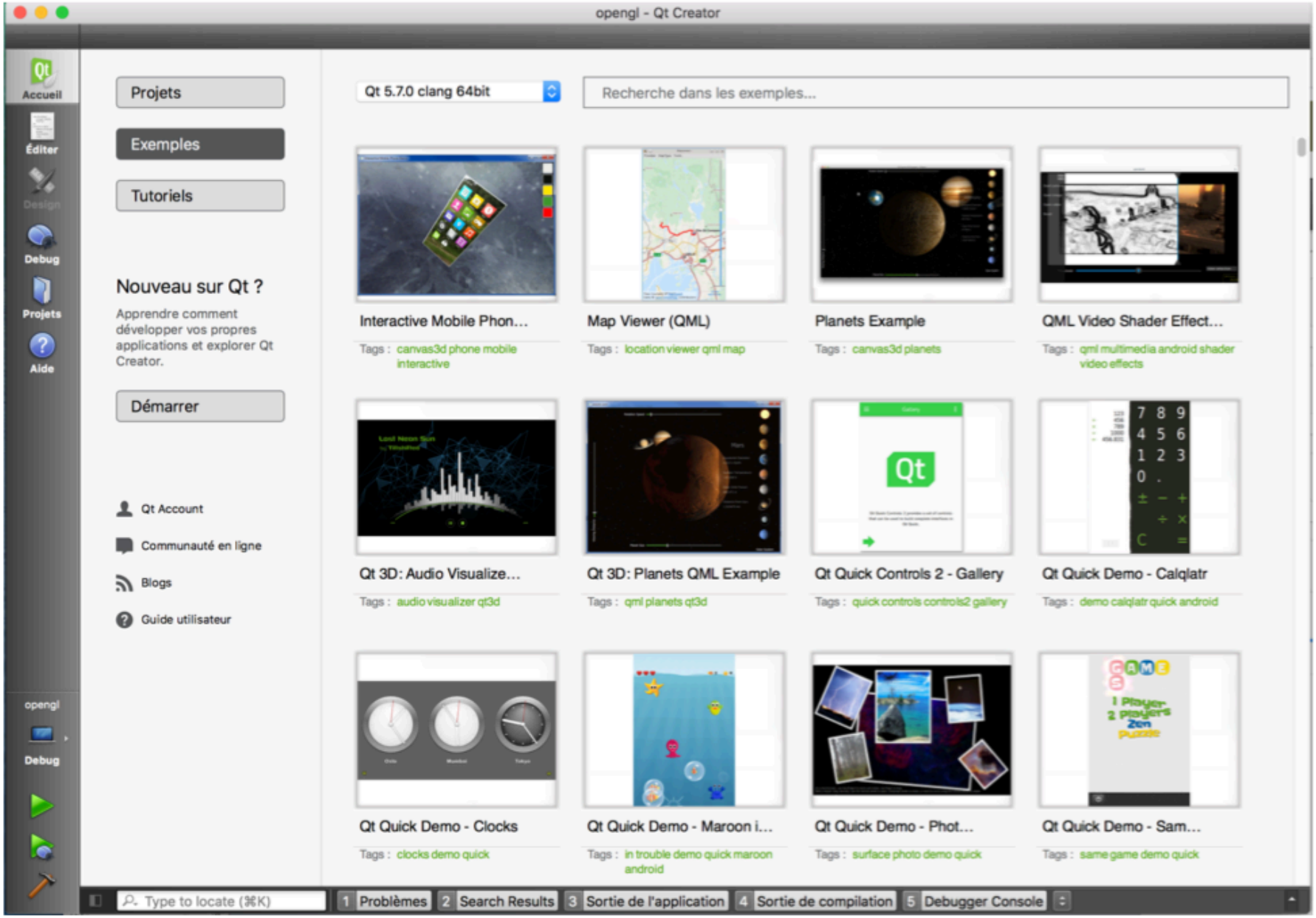
Comment l'utiliser

- créer un fichier .qrc
- pyrcc5 -o resources.py resource.qrc
- intégrer la resource dans le code. Ne pas oublier le ':' et le prefix

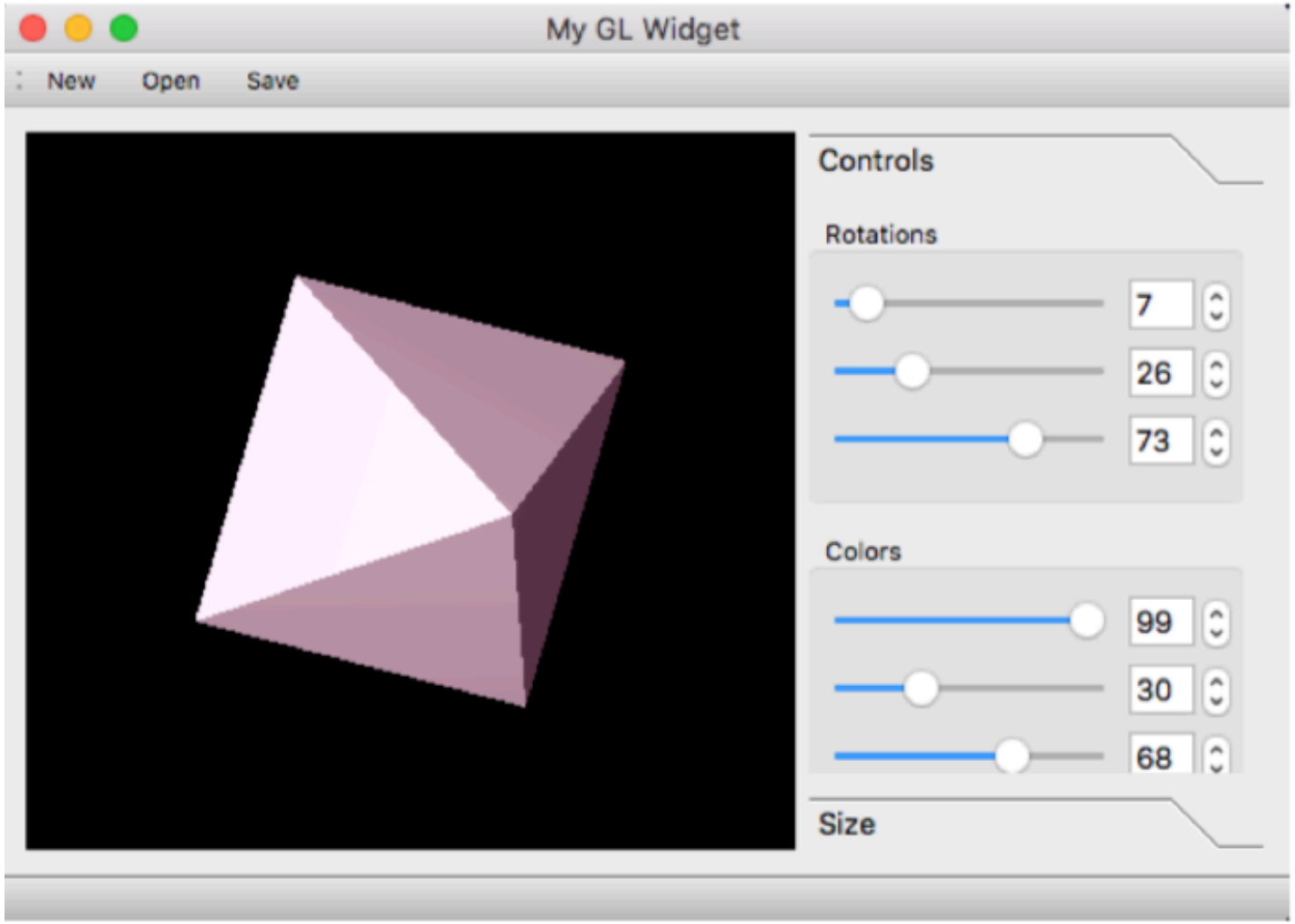
```
<RCC>
  <qresource prefix="/images">
    <file>resources/128x128/james.jpg</file>
    <file>resources/128x128/jacque.jpg</file>
    <file>resources/128x128/sandra.jpg</file>
    <file>resources/128x128/alex.jpg</file>
  </qresource>
</RCC>
```

QIcon(":/images/resources/128x128/james.jpg")

QtCreator


opengl - Qt Creator
Accueil
Éditer
Design
Debug
Projets
Aide
opengl
Debug
Projets
Exemples
Tutoriels
Nouveau sur Qt ?
Apprendre comment développer vos propres applications et explorer Qt Creator.
Démarrer
Qt Account
Communauté en ligne
Blogs
Guide utilisateur
Qt 5.7.0 clang 64bit
Recherche dans les exemples...
Interactive Mobile Phon...
Tags : canvas3d phone mobile interactive
Map Viewer (QML)
Tags : location viewer qml map
Planets Example
Tags : canvas3d planets
QML Video Shader Effect...
Tags : qml multimedia android shader video effects
Qt 3D: Audio Visualize...
Tags : audio visualizer qt3d
Qt 3D: Planets QML Example
Tags : qml planets qt3d
Qt Quick Controls 2 - Gallery
Tags : quick controls controls2 gallery
Qt Quick Demo - Calqlatr
Tags : demo calqlatr quick android
Qt Quick Demo - Clocks
Tags : clocks demo quick
Qt Quick Demo - Maroon i...
Tags : in trouble demo quick maroon android
Qt Quick Demo - Phot...
Tags : surface photo demo quick
Qt Quick Demo - Sam...
Tags : same game demo quick
Type to locate (⌘K)
1 Problèmes
2 Search Results
3 Sortie de l'application
4 Sortie de compilation
5 Debugger Console

QtOpenGl


My GL Widget
New
Open
Save
Controls
Rotations
7
26
73
Colors
99
30
68
Size

QtChart


Theme:
Light
Animation:
No Animations
Legend:
No Legend
Anti-aliasing
Area chart
10.0
7.5
5.0
2.5
0.0
0.0
1.5
3.0
4.5
6.0
Pie chart
Slice 0:0
Line chart
7.0
5.3
3.5
1.8
0.0
0.0
2.5
5.0
7.5
10.0
Bar chart
20.0
15.0
10.0
5.0
0.0
1
2
3
4
5
6
7
Spline chart
7.0
5.3
3.5
1.8
0.0
0.0
2.5
5.0
7.5
10.0
Scatter chart
7.0
5.3
3.5
1.8
0.0
0.0
2.5
5.0
7.5
10.0