Introduction to PyQt

Sylvain Malacria

http://www.malacria.com/ mailto:

sylvain.malacria@inria.fr

Slides inspired Gilles Bailly

Documentation

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

What is Qt?

written in C ++ graphics library by the company TrollTech

- Mechanism to interact:
  - with the user (button, drop-down list, ..)
  - with the system (OpenGL, XML, SQL, sockets, plug ...)

Multi-Platform

- Windows | Mac | Linux
- Android | iOS | WinRT | BlackBerry * | Sailfish
- Embedded Linux | Embedded Android | Windows Embedded

Free (GPL), but also has a commercial license

Approach: Write once, compile anywhere

Historical

Haavard & Eirik had the idea of creating a graphics library

Qt 0.9 First public distribution X11 / Linux

Qt 1.0 (business licenses and open source)

Qt 2.0 Open Source (QPL License)

Qt 3.0 Mac Support *Qt Designer*

Qt 4.0 (GPL 2.0 license for every platform)

Nokia acquires Trolltech (Qt's parent company) and its 250 employees with Qt 4.5

Distribution *QtCreator*

Qt 4.5 Qt 4.6 animation; GraphicScene; state machine; Qt gestures was

bought by Digia (target *Android, iOS and Windows 8)*

Qt 5.0 Qt Quick (creation of dynamic interfaces) 20th anniversary of

the first public distribution Qt Support Mobile (iOS, Android, 1990 1995 1996

1999 2001 2005 2008 2009 2008 2010 2011 2012 2015 2016

Why Qt?

Performance (C ++) Relatively

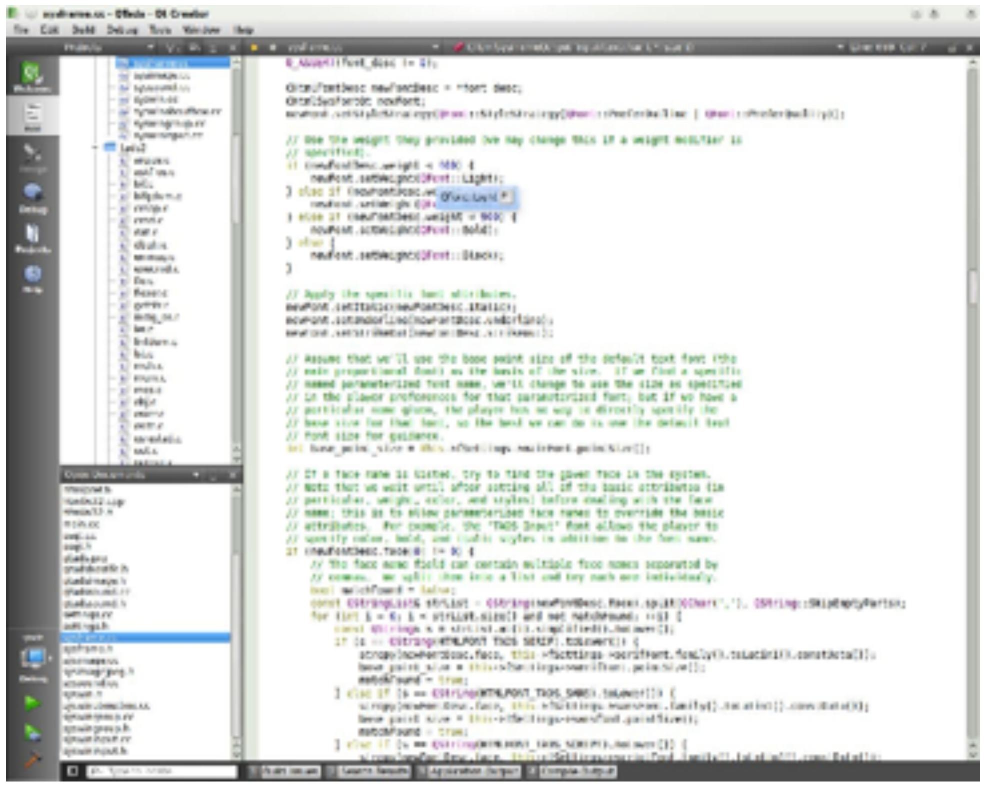
Simple Free (GPL) and source

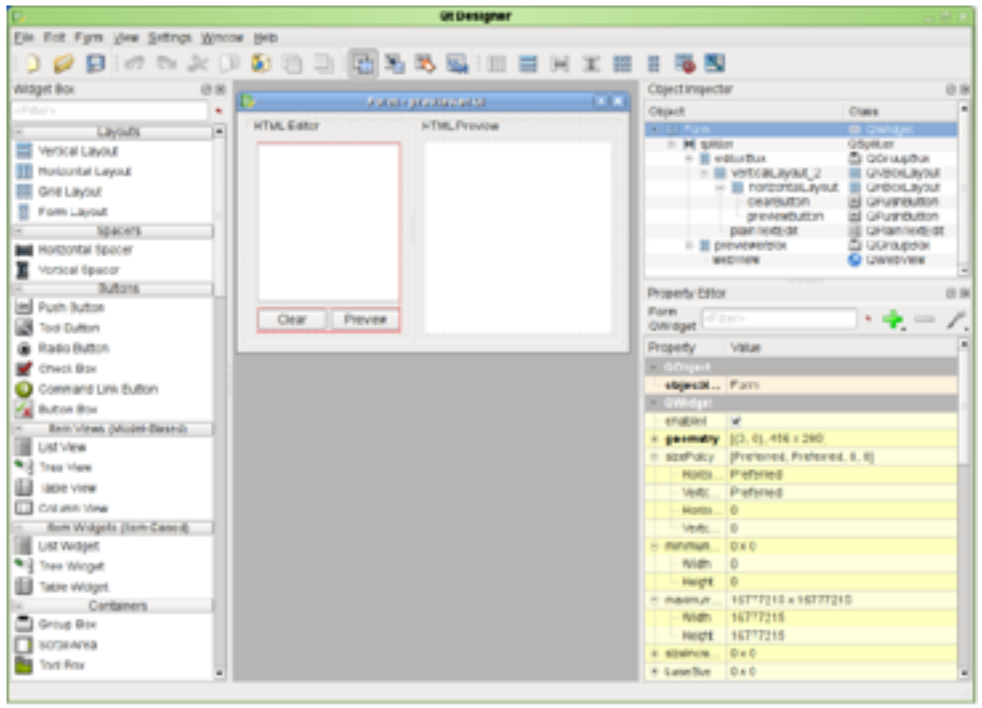
code Many tools

- Interface Builder: Qt Designer
- Internationalization Qt Linguist
- Documentation: Qt Assistant
- Examples: Qt Templates
- Programming: Qt Creator (eclipse)

Multi-Platform

- Look and feel simulated


Creator

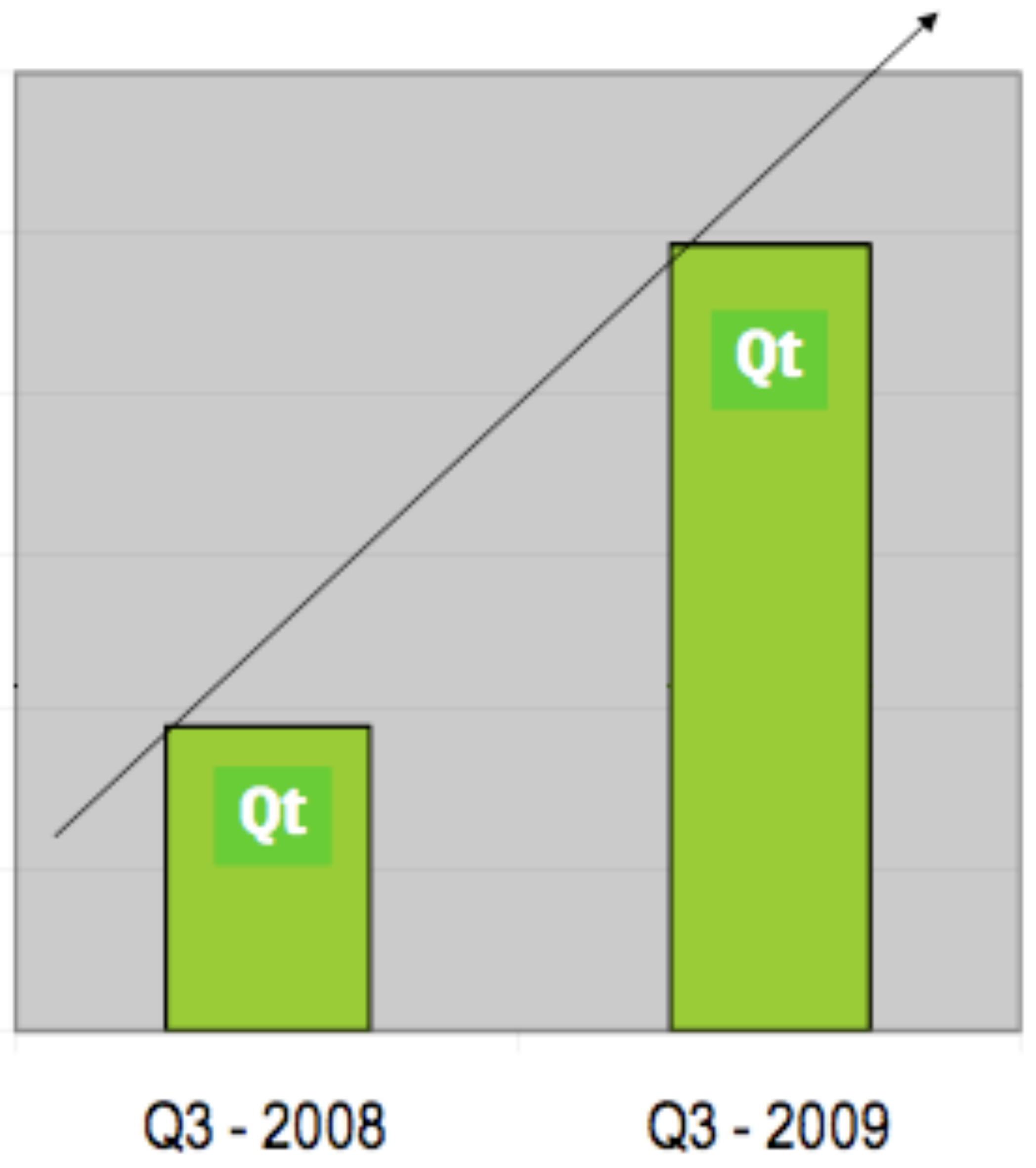


Designer Qt

Today

250% increase of the GPL downloads

Qt Downloads from qt.nokia.com



Qt 5: 10 000 per day download

Today

Qt users:

- ESA, Nokia, NASA, Adobe, Motorola, Google, ...

... and many, many more!

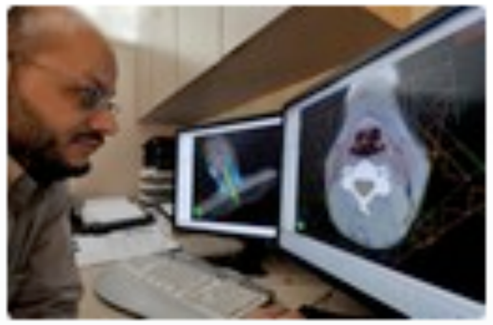
Bindings (java, **python,** c #)

Qt in Automotive Infotainment

Qt in Aerospace

Qt in Home Media

Qt in IP Communication


Qt in Medical

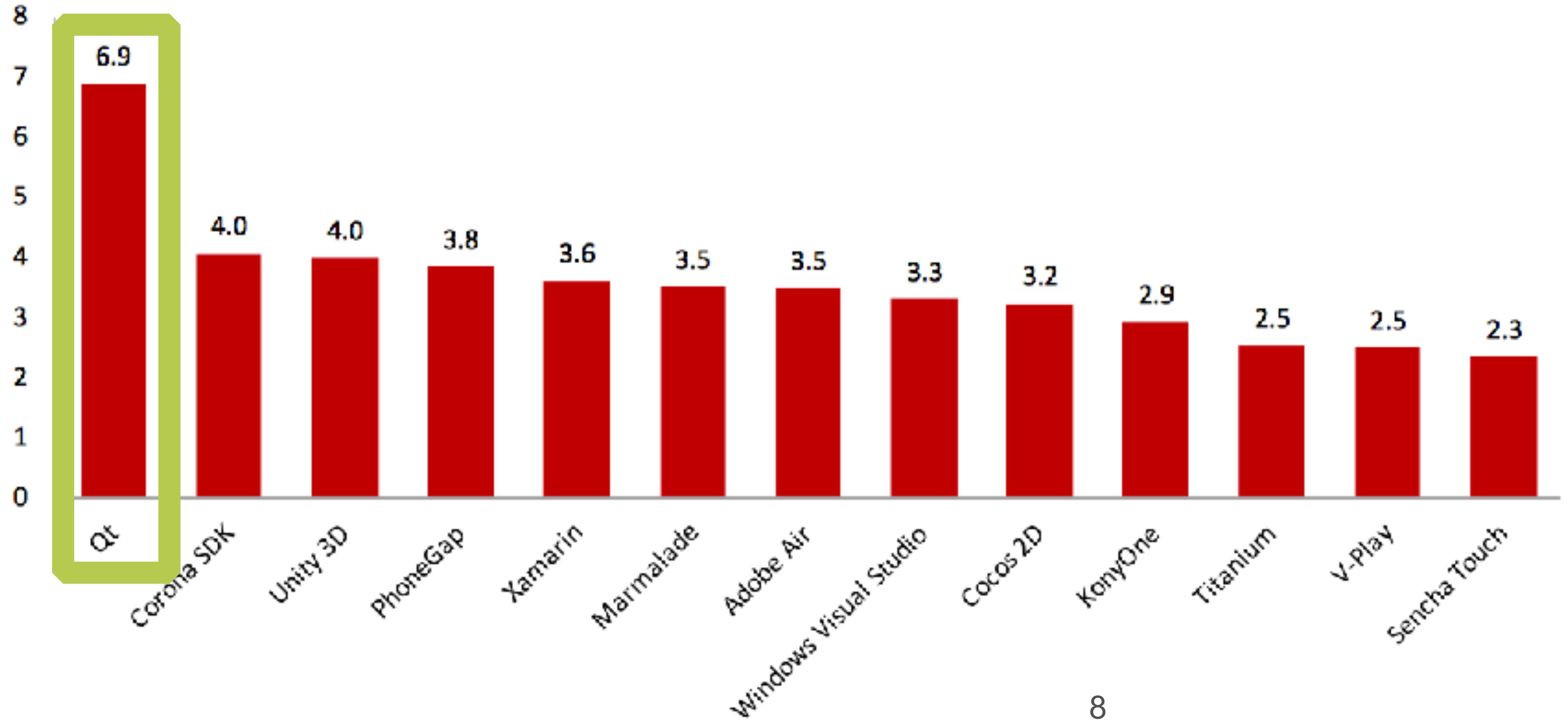
Qt in Oil & Gas

Qt in Visual Effects

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

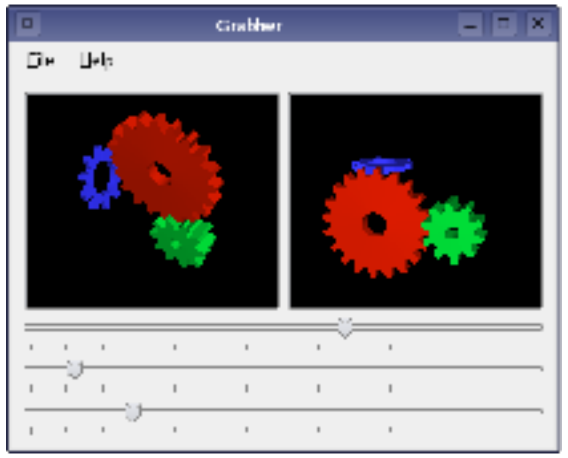


research2guidance 39: Top 10 Cost performance-ratio

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%**	

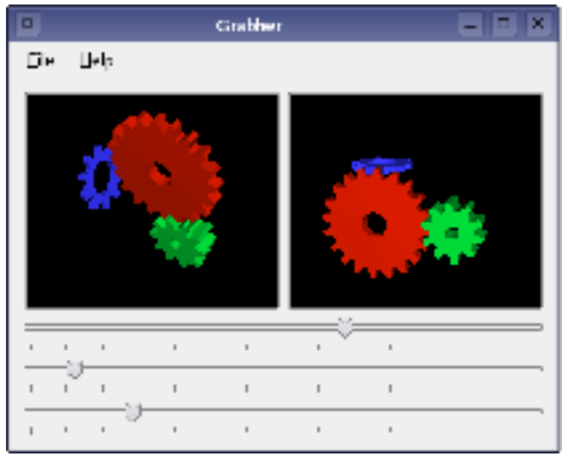
Research2guidance, CPT Benchmarking 2014

Today

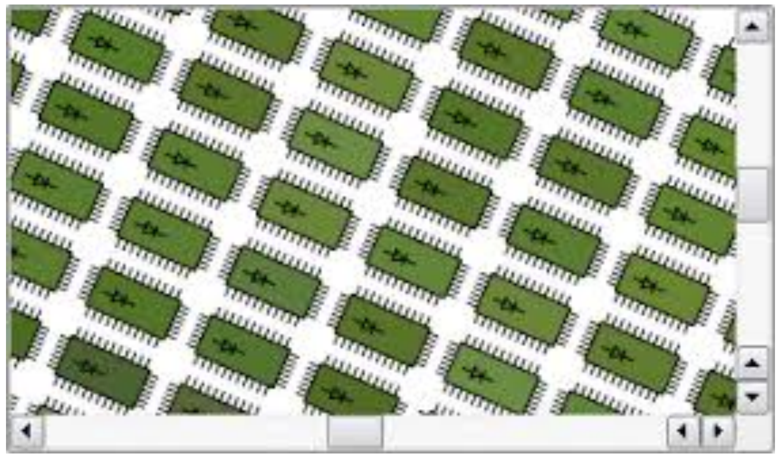


Qt Widgets

Today



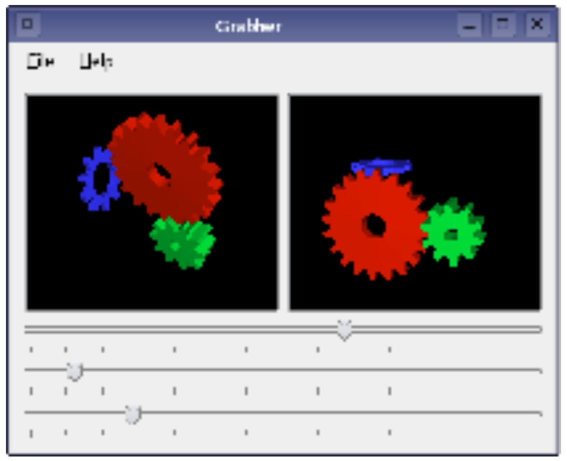
Qt Widgets



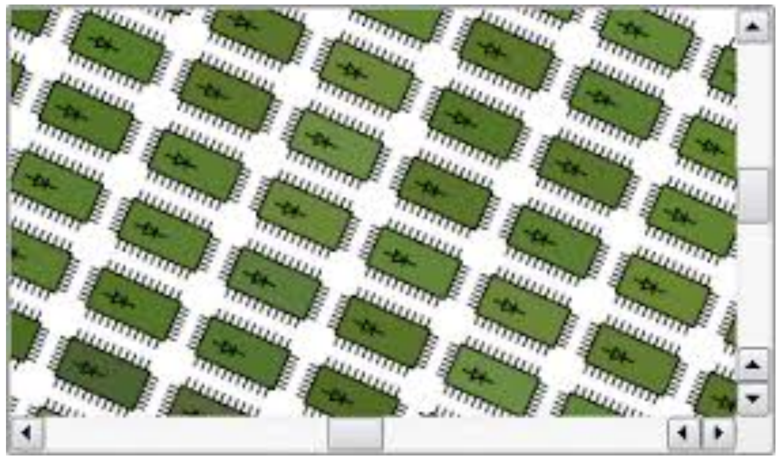
Qt Graphics View

Qt Quick / QML

Today



vs

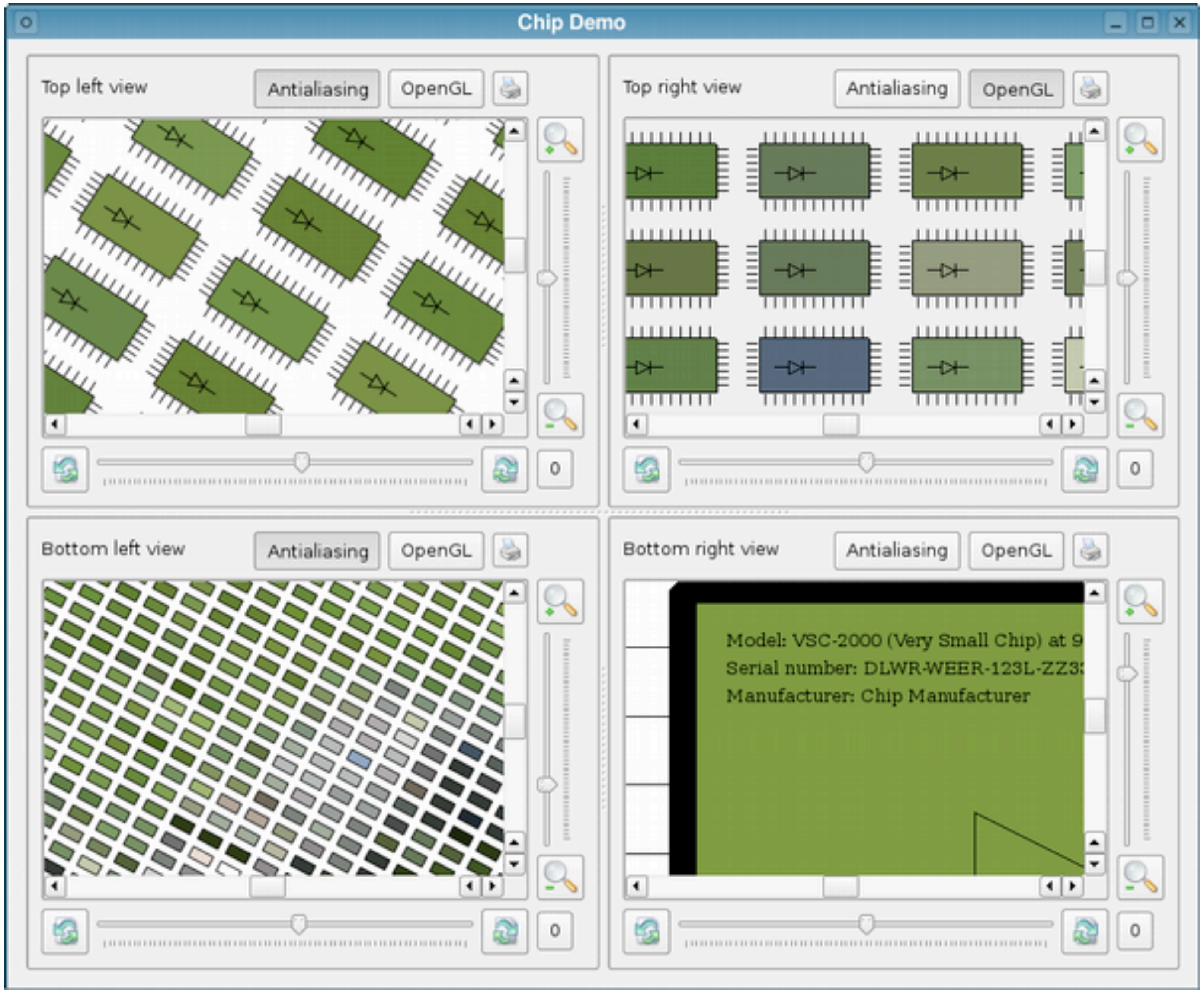


Qt Widgets

Qt Graphics View

- Widgets can not be *processed*
- Widgets use coordinates *pixels;* GraphicsItems into logical units (int double vs)

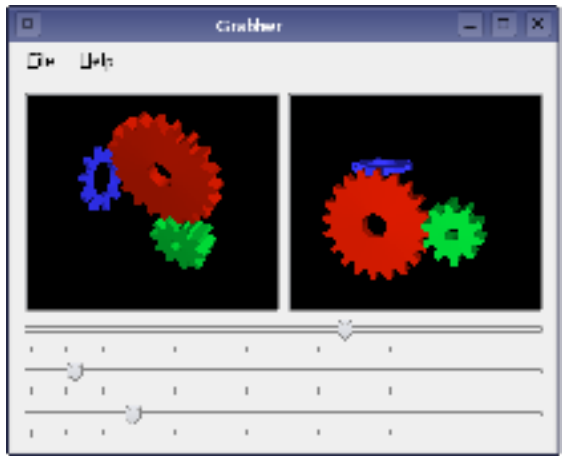
- Widgets can be used in layouts
- 4000000 row widgets, but 4000000 items work fine



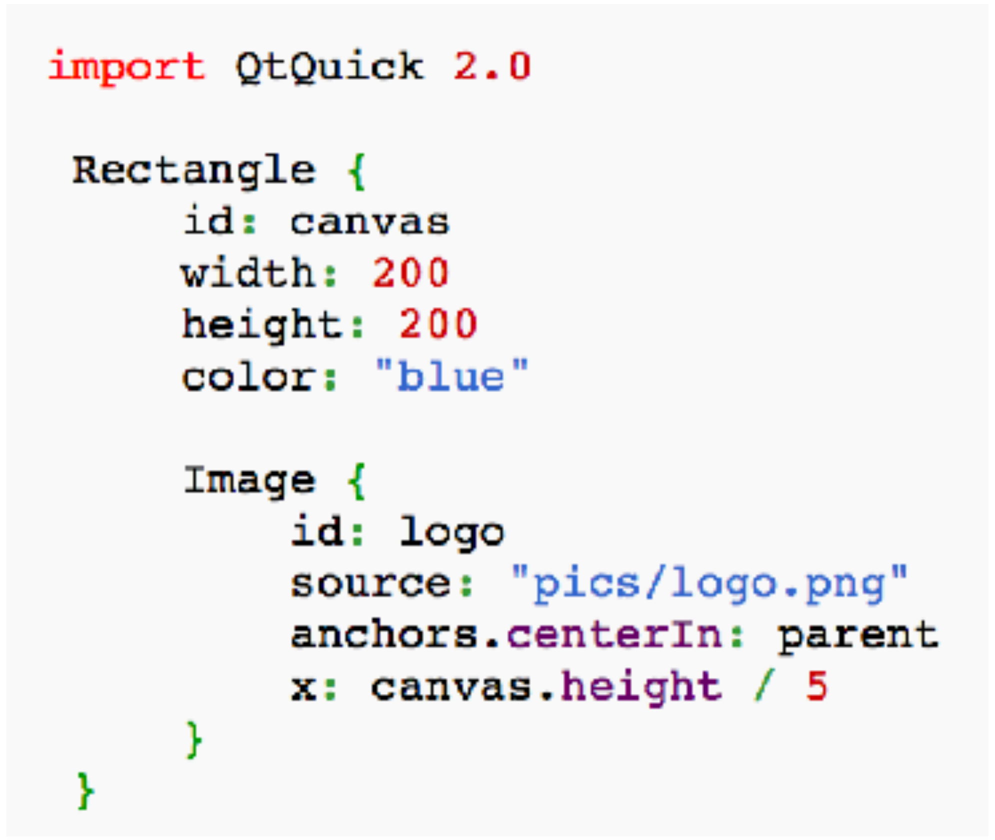
Model: VSC-2000 (Very Small Chip) at 9
Serial number: DLWR-WEER-123L-ZZ3
Manufacturer: Chip Manufacturer

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

Today



vs



```
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
    }
}
```

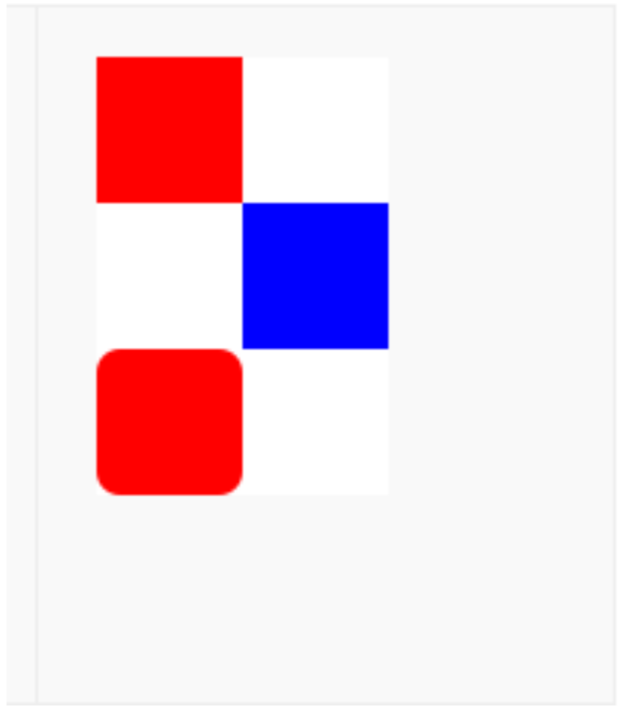
Qt Widgets

Qt Quick / QML

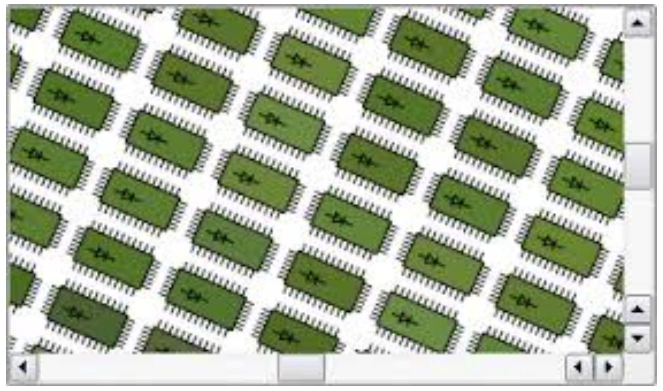
Save declarative

- QML is based on JSON (Language); QtQuick (library)
- QWidgets are more mature, flexible and have more features
- Qt Quick focuses on animation and transition
- Qt Quick is (so far) rather mobile devices
- Qt Quick will (maybe) one day replace QWidgets
- Qt Quick is (perhaps) better for designers (non-computer)

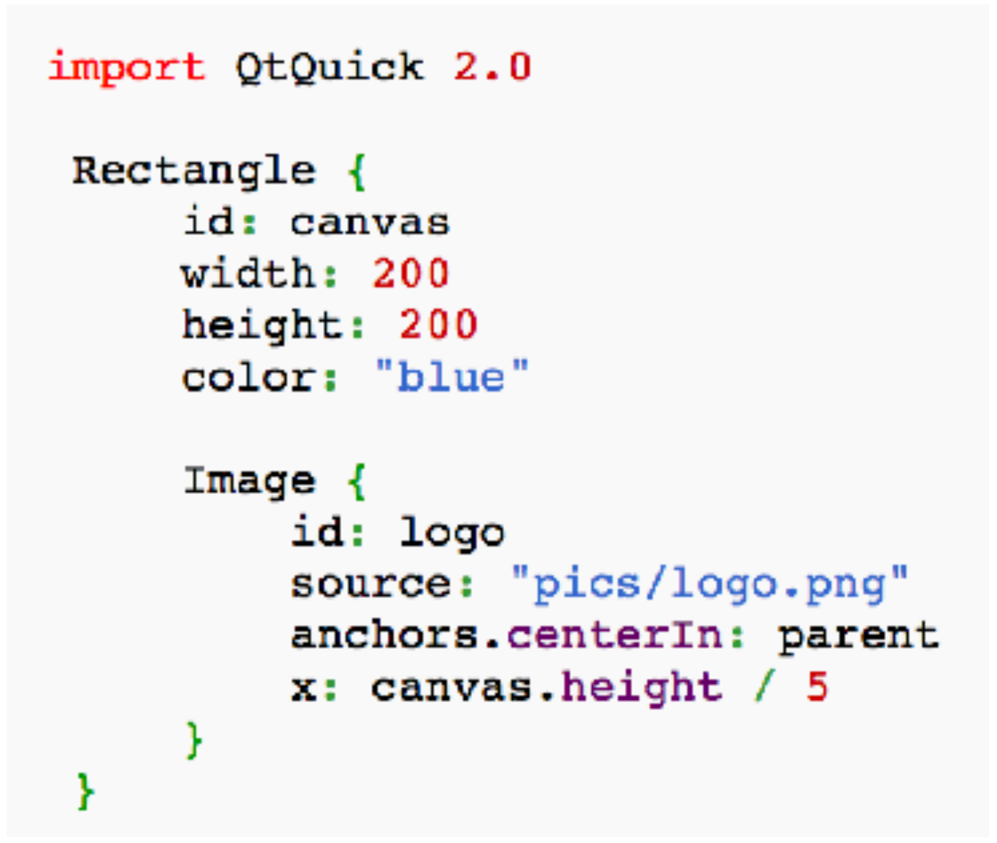
```
// application.qml
import QtQuick 2.3

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



Today



vs



```
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

Save declarative

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

PyQt

Python bindings for Qt v2 and v3

Developed by Riverbank Computing Limited Is for the

same platforms that Qt Binding most popular with *PySide*

PyQt5 for **QT5,** PyQt4 for Qt4

di ff erent licenses Qt (GNU GPL 3 and commercial license) PyQt can

generate python code from Qt Designer Ability to add messages widgets

PyQt Qt Designer

PyQt is not THAT for programming GUI !!!

Objectives of the course

reminders programming **Python**

Getting started with **PyQt**

Introduction to **signals** and **slots** Qt Overview of

the main Qt classes

Python

Programming language object dynamic

typing strong Placed under a free license

Verry much Quick libraries used

Many extensions for numerical calculation

Syntax

syntax C	Python syntax

```
int factorial ( int not) {
        yew ( not <2 ) {
                return 1 ; } else {

                return not * factorial (n - 1 );
        }}
```
	```
	def factorial (not):
	        yew not <2 :
	                return 1
	        else : return not * factorial (n - 1 )
	```

Syntax

syntax C	Python syntax
int factorial (int not) {	def factorial (not):
yew (not <2) {	yew not <2 :
return 1 ; } else {	return 1
	else : return not * factorial (n - 1)
return not * factorial (n - 1);	
}}	

Beware tabs !!!

Dynamic typing strong

~~int at = 4~~

```
at = 4
type (at)                        <Class 'int'>
```

```
at = 4.1
type (at)                        <Class 'fl oat'>
```

Some basic types

boolean

digital

- int
- long
- fl oat
- complex

collections

- list
- tuple
- set
- dict
- etc.

Some basic types

boolean

digital

- int
- long
- fl oat
- complex

collections

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

```
list1 = [ 'Physics' , 'Chemistry' , 1200 ] Print (list1 [ 0 ])

        >>> 'Physics'

list1 . append ( "Blah" ) Print (list1 [ 3
])

        >>> 'Bla'

print (list1 [ 1: 3 ])

        >>>  [ 'Chemistry', 1200]
```

Some basic types

boolean

digital

- int
- long
- fl oat
- complex

collections

- list
- tuple
- set
- dict
- etc.

```
list1 = [ 'Physics' , 'Chemistry' , 1200 ] Print (list1 [ 0 ])

        > > > 'Physics'

list1 . append ( "Blah" ) Print (list1 [ 3
])

        > > > 'Bla'

print (list1 [ 1: 3 ])

        > > >  [ 'Chemistry', 1200]
```

Basic operations on collections (List, Tuple)

```
alphabetT = ( 'at' , 'B' , 'C' , 'D' , 'E' , 'F' , 'G' , 'H' , 'I' , 'J' , 'K' ) alphabetL = [ 'at' , 'B' , 'C' , 'D' , 'E' , 'F' , 'G' , 'H' , 'I' , 'J' , 'K' ] alphabetT2 = 'at' , 'B' ,
, 'D' ( 'E' , 'É' , 'È' ) 'F' , 'G' , 'H' , 'I' , 'J' , 'K'
```

➡ tuple
➡ list

```
len (AlphabetT)
                >>> 11
```

➡ Number of Items

```
len (AlphabetT2 [ 4 ])
                >>> 3
```

```
alphabetT [ -2 ]
                >>> 'J'
```

➡ Access *since the end*

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

➡ Loop *Each for*

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

➡ Loop *iterative*

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

Input / Output fi le

```
file = open ( "Text.txt" , "R" ) text = file . read ()

file . write ( "BLA bla \ not " )
file . close ()
```

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

classes

```
class Car (vehicle) :                                ➡ Declaration of a class that inherits Car Vehicle
    # comment
    nbRoues = 4

    def __ init__ ( self , Brand, color): super () .__ init __ ()        ➡ Declaring a constructor
                                                    ➡ Call the constructor of the superclass
        self . color = color                        ➡ Declaring an instance variable
        self . Mark = Mark                          ➡ Declaring an instance variable
```

```
                    ➡ Conventional declaration of main ()
def hand ():
    audirouge = Car( 'Audi' , 'red' )
    print ( audirouge . color)

                                                    ➡ Automatic call Hand
yew __ name__ == "__hand__" :
     hand( sys.argv )
```

documentation PyQt

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

The main modules

QtCore

QtWidgets

QtGui

QtBluetooth

QtOpenGL

QtSript / QtScriptTools QtSql

QtSvg QtWebKit

QtXml / QtXmlPatterns

QtMultimedia QtSensors

Module QtCore

QObject

Base Type:

- QChar, QDate, **QString,** QStringList, Qtime ... File

systems:

- QDir, QFile, ...

Container:

- QList, QMap, Qpair, QSet, QVector ...

Graphic:

- QLine, QPoint, QRect, QSize ...

Thread:

- QThread, QMutex ...

Other:

- QTimer ...

QString

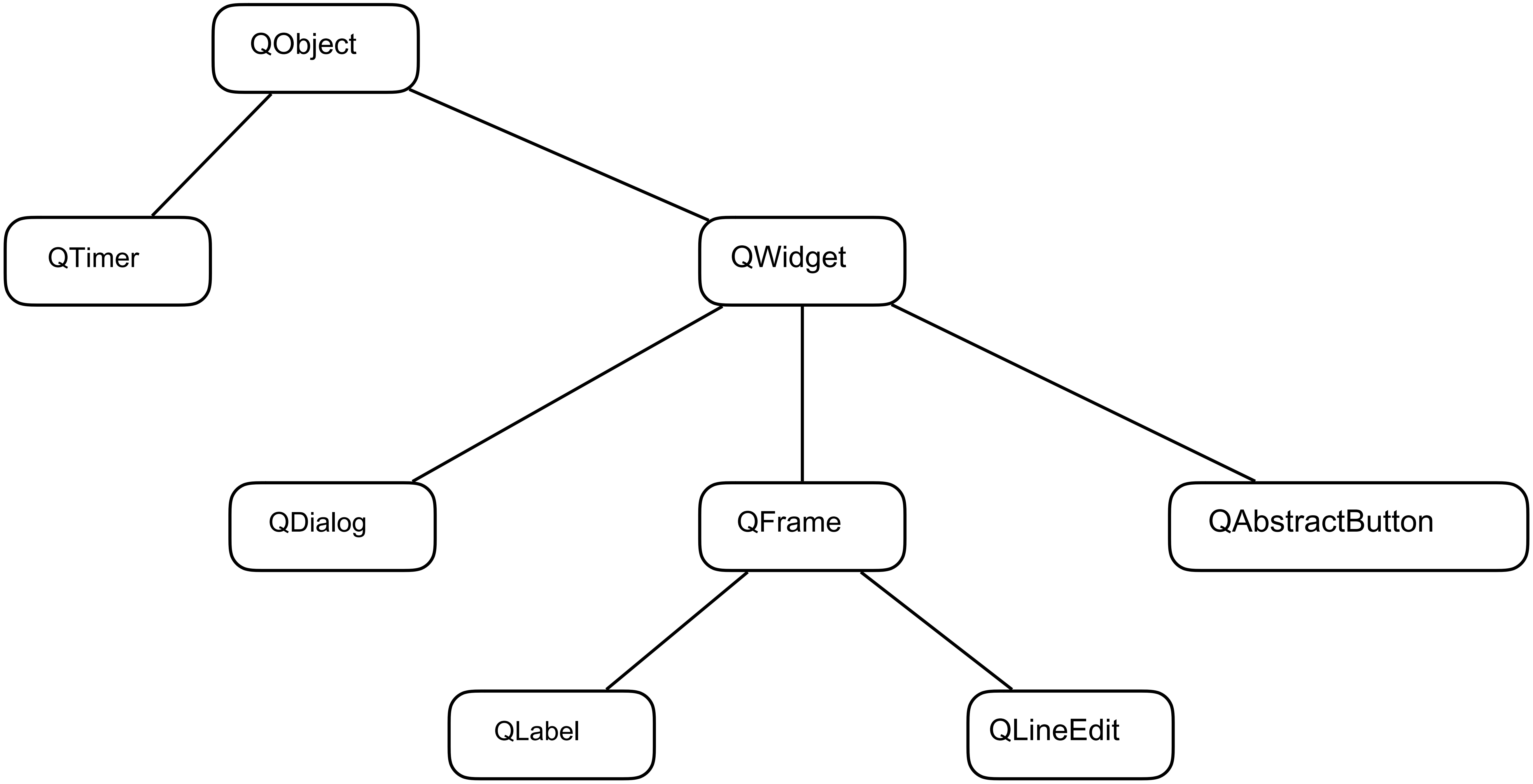
16-bit Unicode encoding

- Following QChar s
- 1 = 1 character QChar 16 bits (usually)
- Character 1 = 2 QChar s of 16 bits (for values> 65535)

conversions a QString :

- **ToASCII ( ):** 8-bit ASCII
- **toLatin1 ( ):** Latin-1 (ISO 8859-1) 8 bits
- **toUtf8 ( ):** Unicode UTF-8 multibyte (1 character 1 = 4 bytes)
- **toLocal8Bit ( ):** local 8-bit encoding

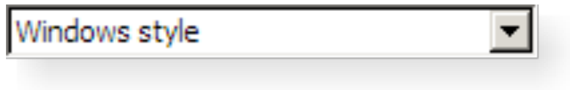
main widgets



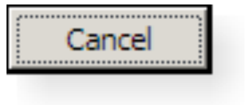
Module QtWidgets

Qwidget

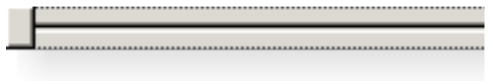
QComboBox



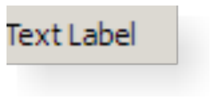
QPushButton



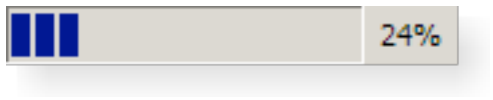
QSlider



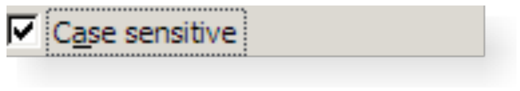
QLabel



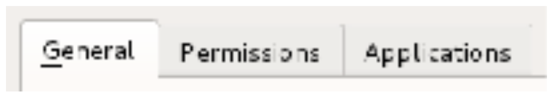
QProgressBar



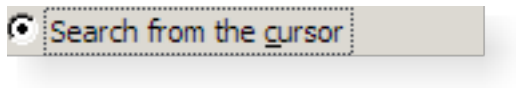
QCheckBox



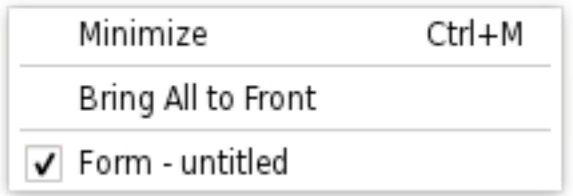
QTabBar



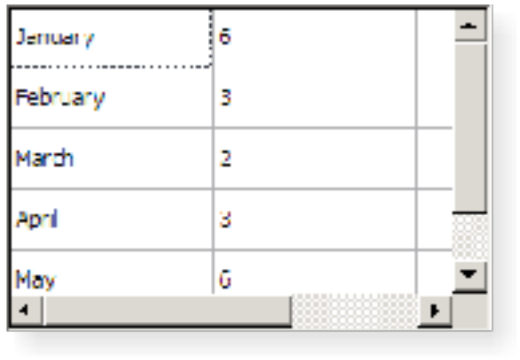
QRadioButton



QMenu

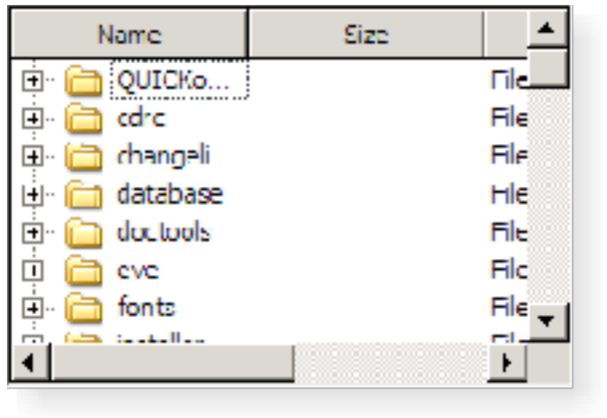


QTableView

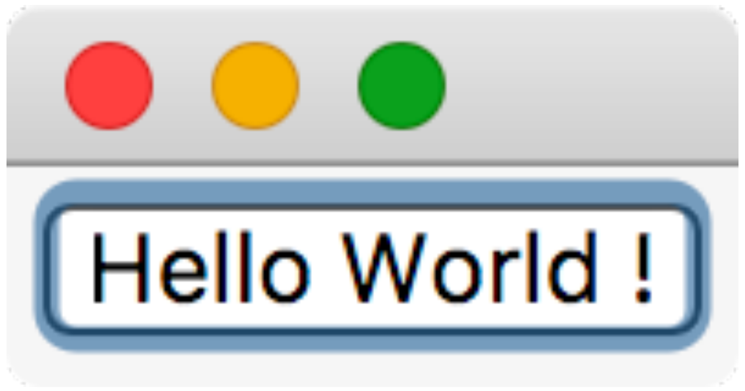


January	6
February	3
March	2
April	3
May	6

QTreeView

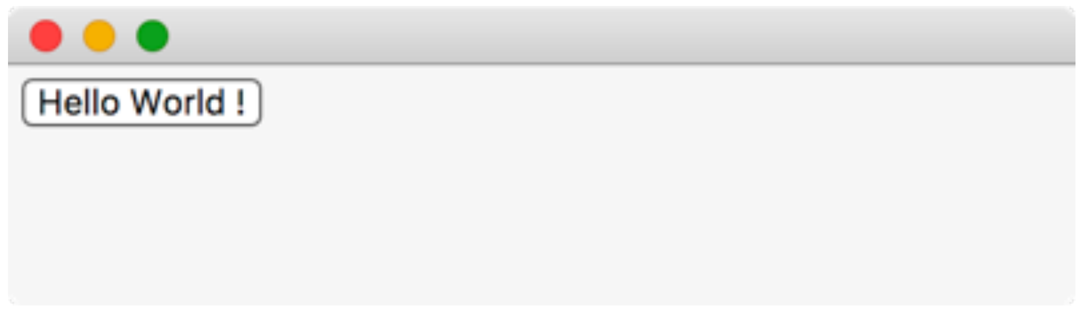


single window



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


def hand (Args):
        app = QApplication (args) button = QPushButton ( "Bonjour Monde !" , None
        ) button . resize ( 100 , 30 ) button . show () app . exec_ ()




yew __ name__ == "__hand__" :
          hand (sys . argv)
```

Single window with button in widget



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


def hand (Args):
       app = QApplication (args) = QWidget widget ( None ) Widget.resize ( 400 , 90 )
       button = QPushButton ( "Bonjour Monde !" , widget ) button . resize ( 100 , 30 )
       Widget () . show () app . exec_ ()


yew __ name__ == "__hand__" :
         hand (sys . argv)
```

Signals and slots

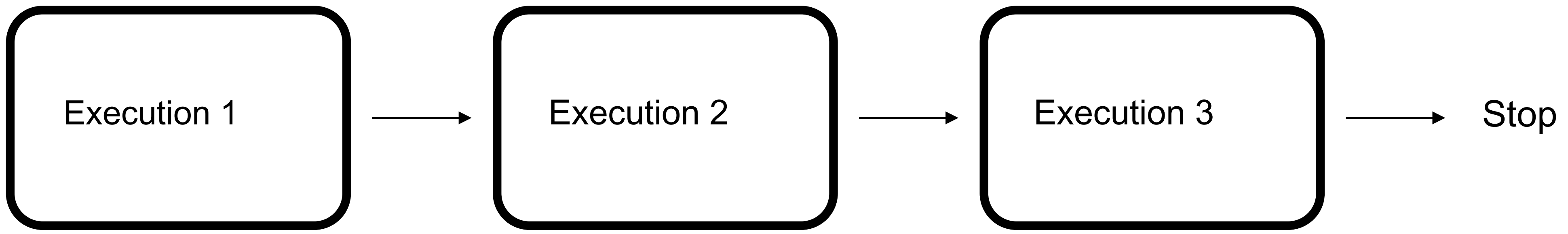
How, from a "click of a button," I can run the part corresponding to the logic of my application? (Eg close the application) ??

solutions:

- MFC (introduced language over C ++)
- Java (using listeners)
- **Qt (mainly uses signals and slots)**

algorithmic Application

Using procedures (functions) sequentially called Sequence of steps to be made in a certain order



Inputs - Outputs user

"Classic" programming:

- Main program initializes and calls functions in a pre-determined order
- Potential users events are "requested" (Pause program)

programming " **event** ":

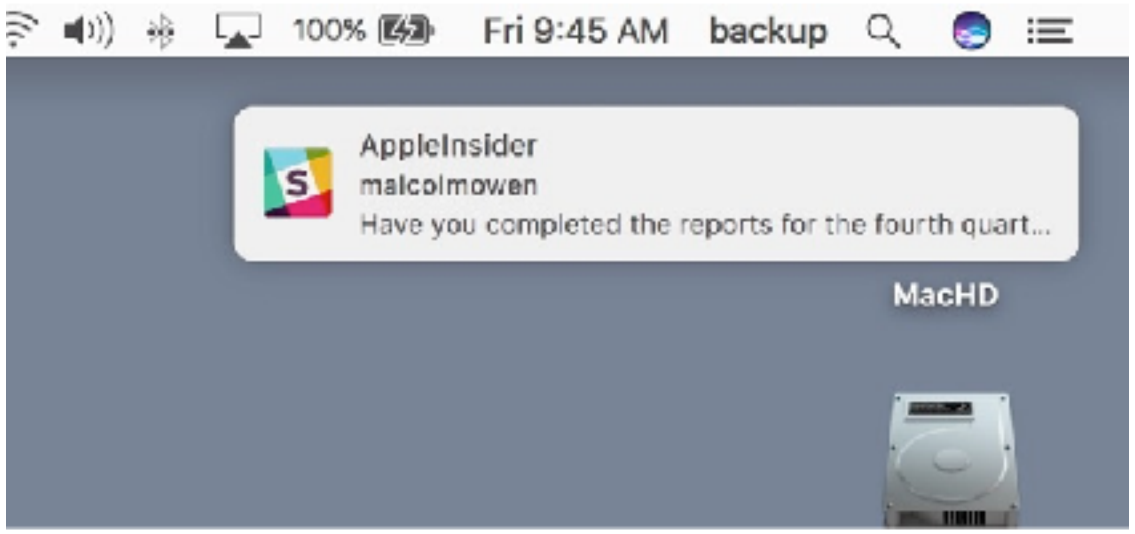
- main program **initializes** variables and functions **react to events**
- The sequence is controlled by the occurrence of events
  (Including user actions)
- main loop that processes events (buried in the library)

What events?

user Actions

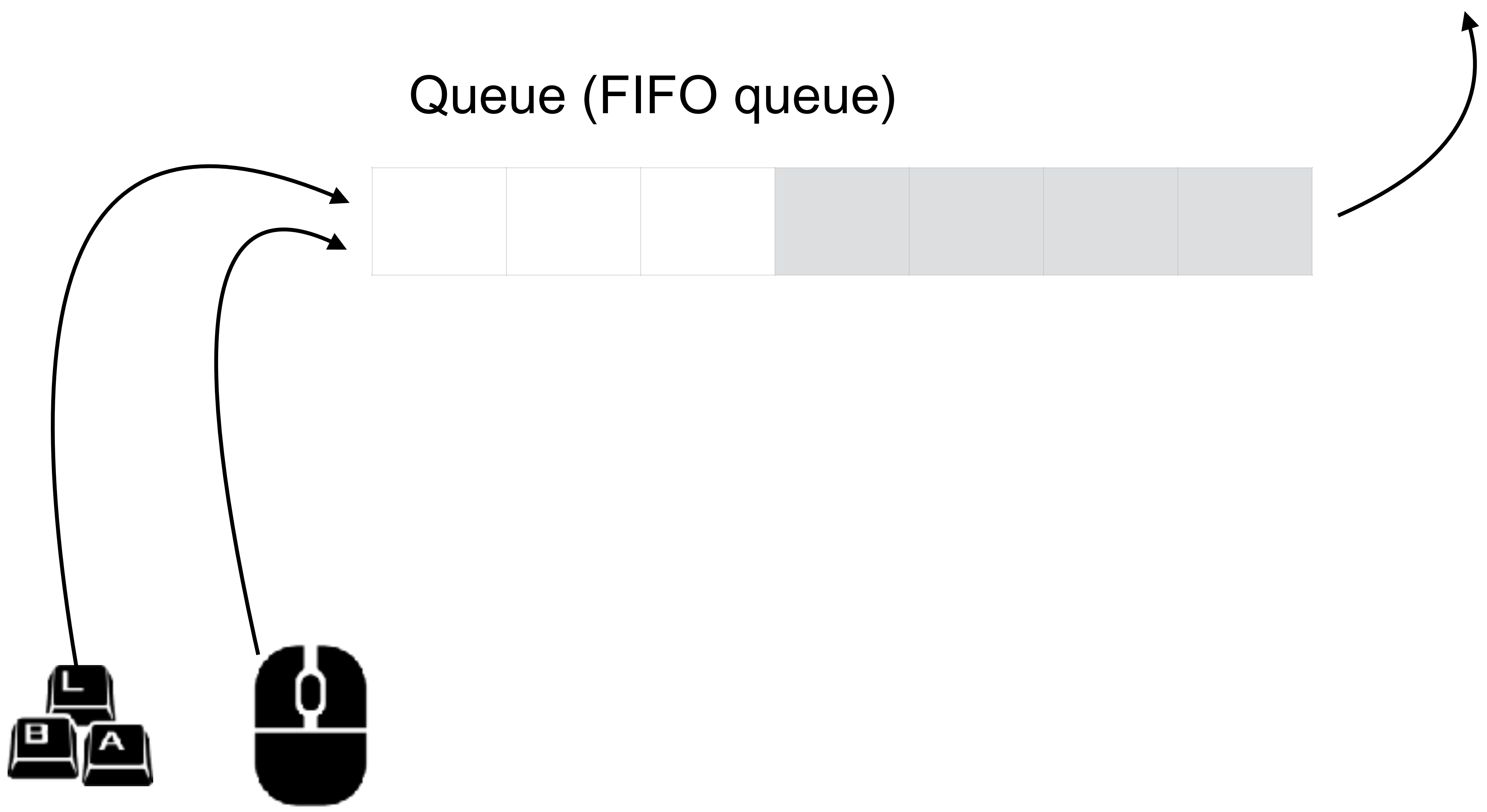
Noti fi cation process (applications, OS, MAJ) sensory

sensors (ubiquitous info)

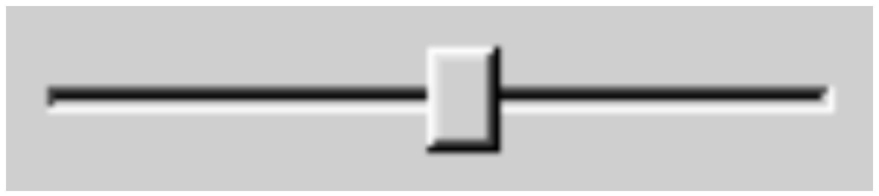


AppleInsider
malcolmowen
Have you completed the reports for the fourth quart...

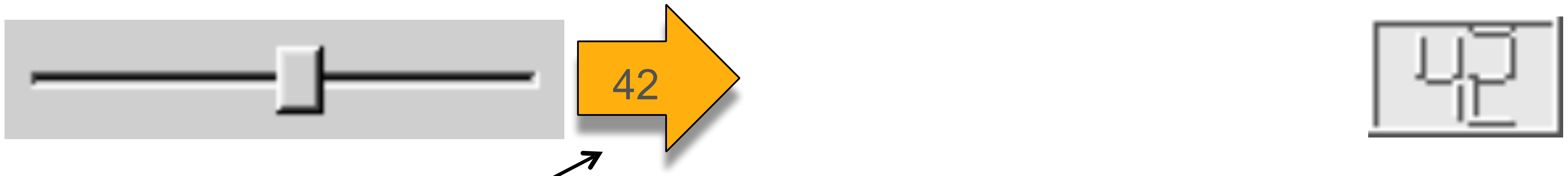
```
while ( true ) {
        yew (! Queue.isEmpty ()) {
                event = queue.nextEvent (); source = findSourceForEvent
            (event); source.processEvent (event); }}
```



Connect signals and slots

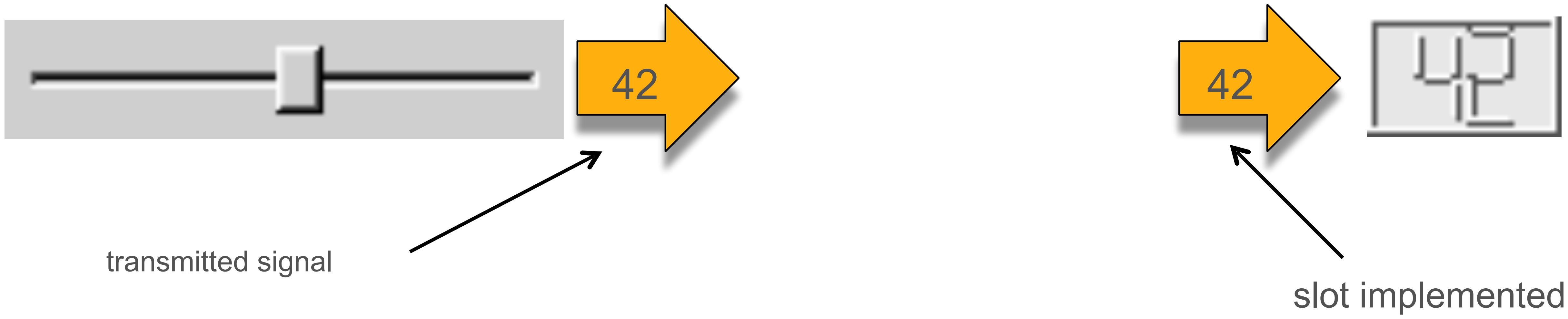


Connect signals and slots

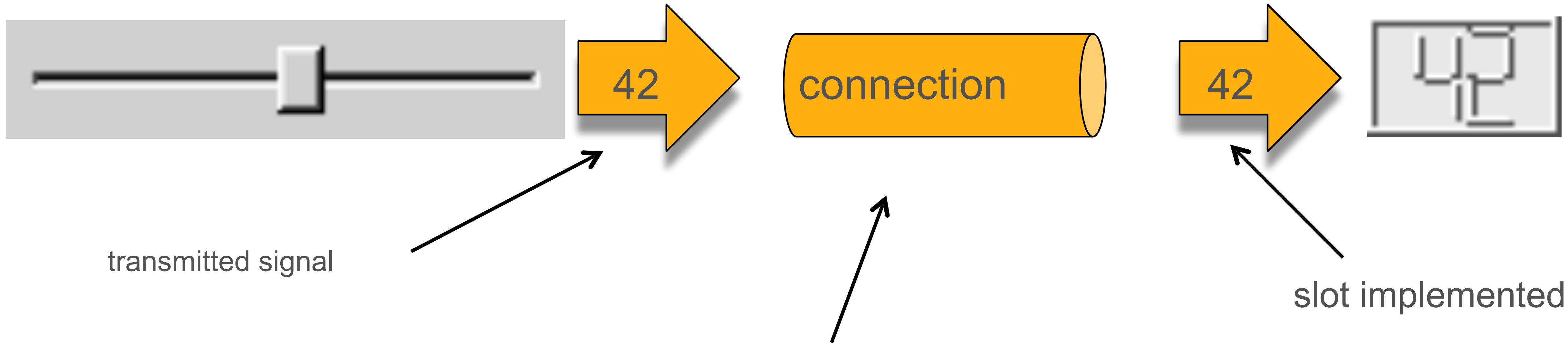


transmitted signal

Connect signals and slots



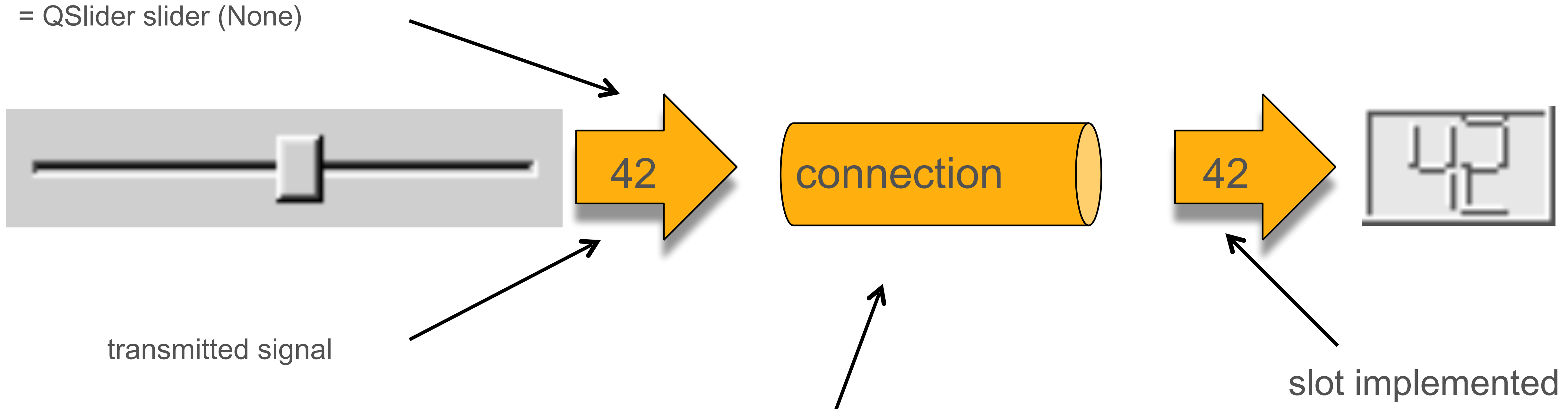
Connect signals and slots



slider.valueChanged.connect (lcd.display)

Connect signals and slots

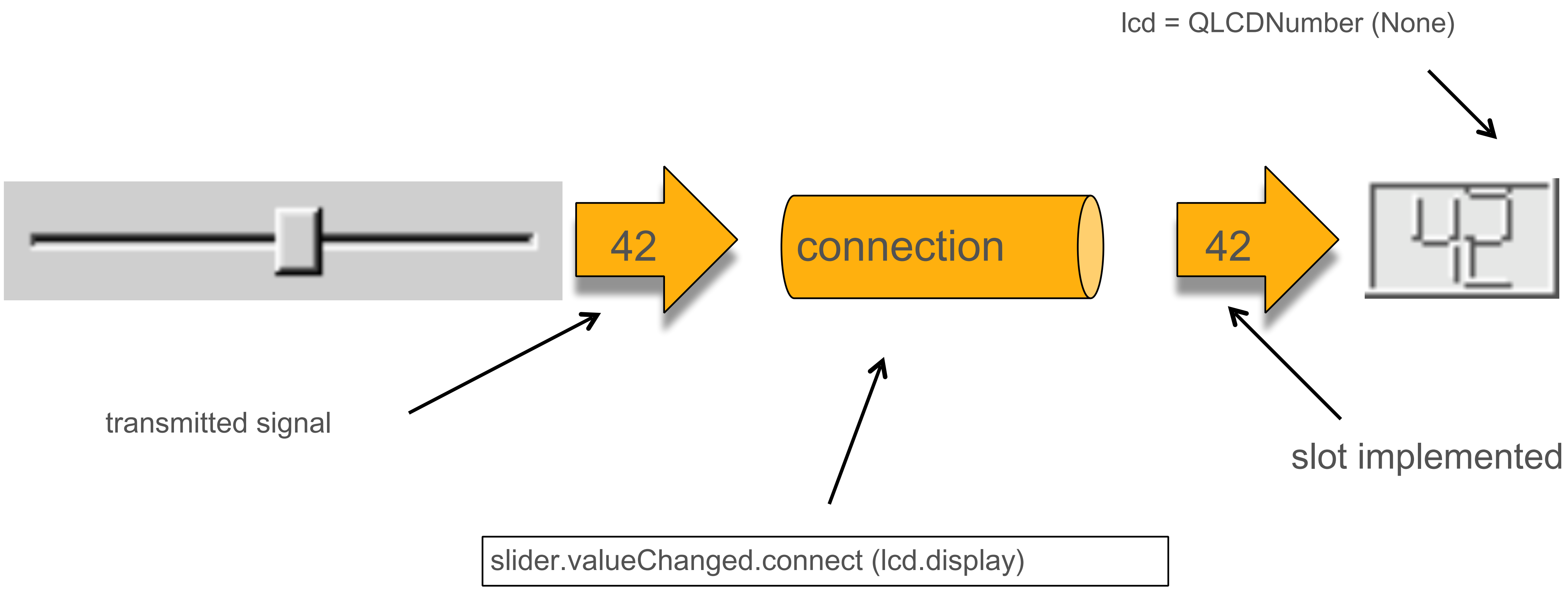
```
class QSlider (QObject)

    ....

    def mousePressEvent (self)

        self.valueChanged.emit (value) ...
```



```
slider.valueChanged.connect (lcd.display)
```

Connect signals and slots

```
class QLCDNumber (QObject)


    def display (num)
        M-value = num; ...
```



```
lcd = QLCDNumber (None)
```

```
slider.valueChanged.connect (lcd.display)
```

A class with signals and slots

```
class MyClass ( QObject ):

        mySignal = pyqtSignal ( int)

        void mySlot (self, num):
                BLA bla
```

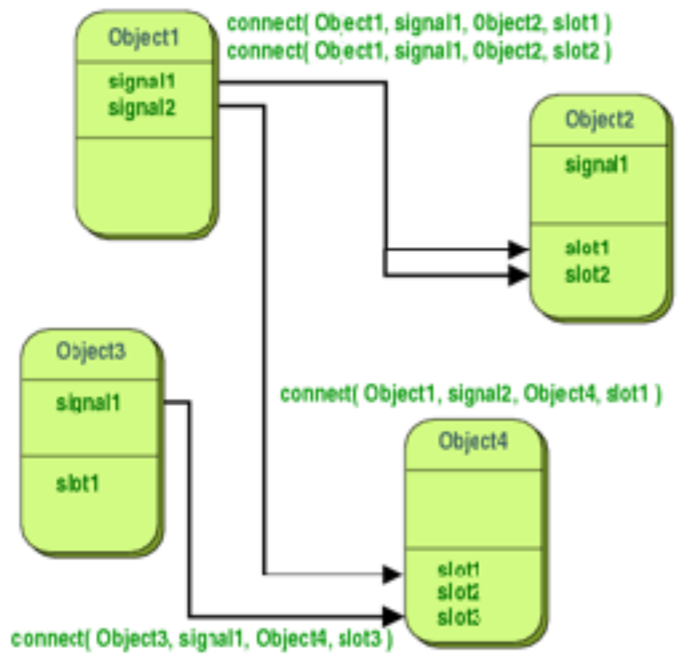
- Sub class of **QObject**
- The **signals** are not implemented
- The **slots** beings are implemented

A class with signals and slots

```
class MyClass ( QObject ):

      mySignal = pyqtSignal ( int)

      def __ init __ (self, parent = None):
            great (MyClass, self) .__ init __ (parent)

      @pyqtSlot ( int)
      void mySlot (num):                    ←   sometimes necessary
            BLA bla
```

- Sub class of **QObject**
- The **signals** are not implemented
- The **slots** beings are implemented

Signals and slots


connect(Obect1, signal1, Object2, slot1)
connect(Obect1, signal1, Object2, slot2)
Object1
signal1
signal2
Object2
signal1
slot1
slot2
Object3
signal1
slot1
connect(Object1, signal2, Object4, slot1)
Object4
slot1
slot2
slot3
connect(Object3, signal1, Object4, slot3)

Modularity, flexibility

- log on **many** signals **a** slot
- log on **a** signal **many** slots

Philosophy

- The transmitter does not need to know (s) receiver (s)
- The sender does not know whether the signal was received
- The receiver does not know the sender
- component by programming (independent, reusable)

Security, strong typing

- The types of settings must be the same
- A slot can have **less** parameters a signal

Example: transfer of money between banks

```
class PunchingBag ( QObject)
        punched = pyqtSignal ()                              Signal

        def __ init__ ( self ):
                #   Initialize the PunchingBag have a QObject
                QObject . __init __ ( self )

        def punch ( self ):                                  slot
                self . punched . emit ()


@pyqtSlot ()
def say_punched ():
        print ( ' Bag Was punched. ' )


def hand (Args):
        bag = PunchingBag ()
        #   Connect punched the bag's signal to the slot say_punched
        bag . punched . connect (say_punched)                        Log in

        #   Punch the bag 10 times
        for i in tidy ( 10 ): Bag . punch()


yew __ name__ == "__hand__" :
        hand (sys . argv)
```

Questions

How to connect a signal to a slot?

- EmetteurObj. <NameSignal>. **connect (** Receiver. <NameSlot>)

What code to declare / implement a slot?

- nothing special (but you can add @pyqtSlot ()

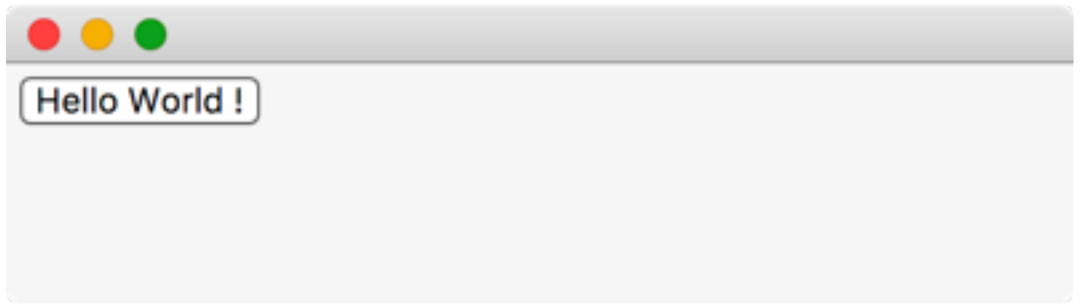
Is that a slot can return a value?

- Yes

What code to declare / implement a signal?

- mySignal = pyqtSignal ()

Single window with button in widget



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

def hand (Args):
     app = QApplication (args) = QWidget widget ( None ) Widget.resize ( 400 , 90 )
     button = QPushButton ( "Bonjour Monde !" , widget ) button . resize ( 100 , 30 )
     Button.clicked.connect ( app.quit ) Widget () . show () app . exec_ ()
```

```
yew __ name__ == "__hand__" :
      hand (sys . argv)
```

The main modules

QtCore

QtWidgets

QtGui

QtBluetooth

QtOpenGL

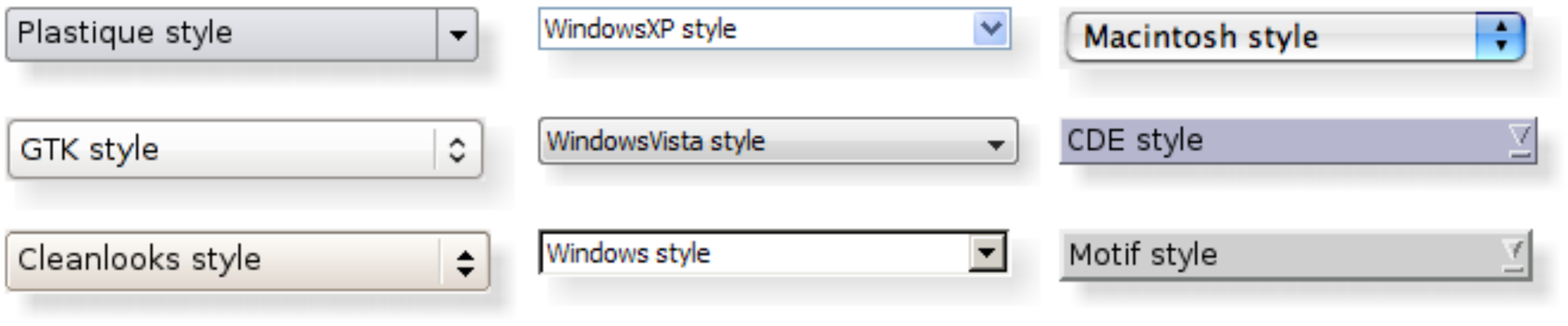
QtSript / QtScriptTools QtSql

QtSvg QtWebKit

QtXml / QtXmlPatterns

QtMultimedia QtSensors

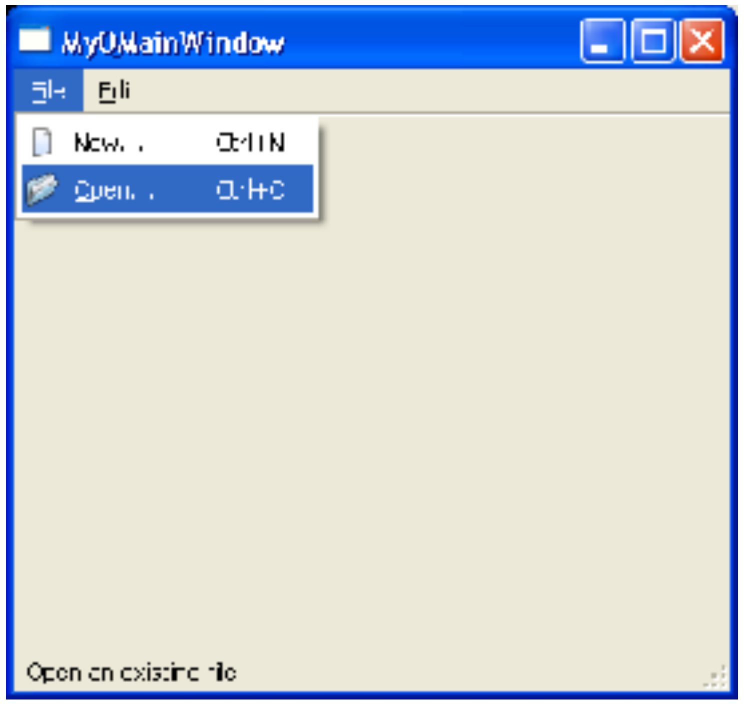
QStyle

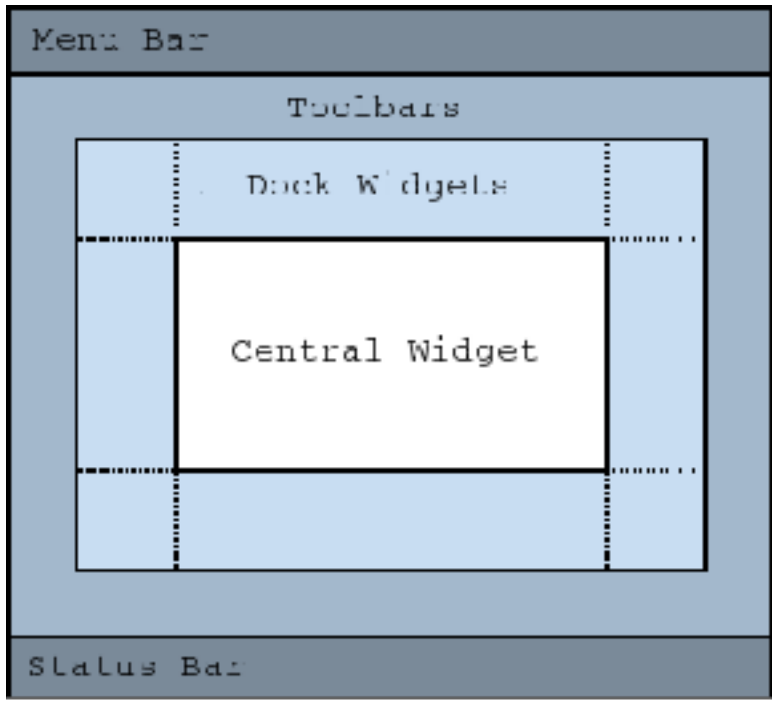


Can be passed as argument to the execution of the program

Ex: python3 test.py -style Windows

QMainWindow





~~Method 1: cre~~ate an instance of QMainWindow

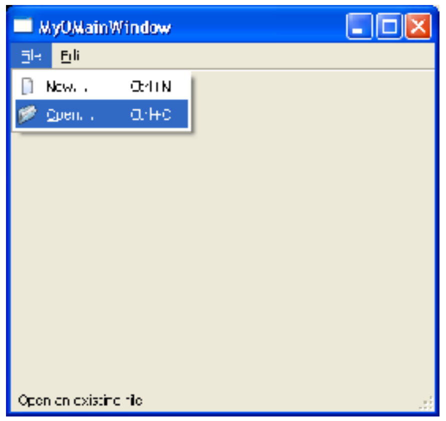
```
win = QMainWindow () Win . resize (
200 , 300 )
```

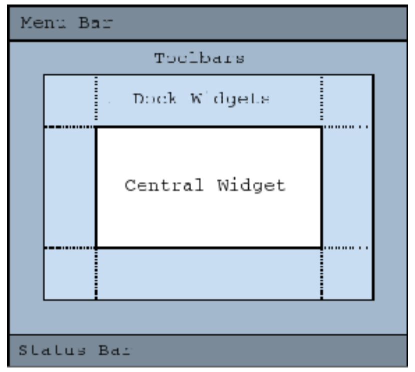
Method 2: Create a subclass QMainWindow

```
class Win ( QMainWindow ): Def __init __ ( self ):
    Self . resize ( 200 , 300 )
```

QMainWindow

menus





bar = self. **menuBar ()**

if subclass (method 2)

otherwise win.menuBar () (method 1)

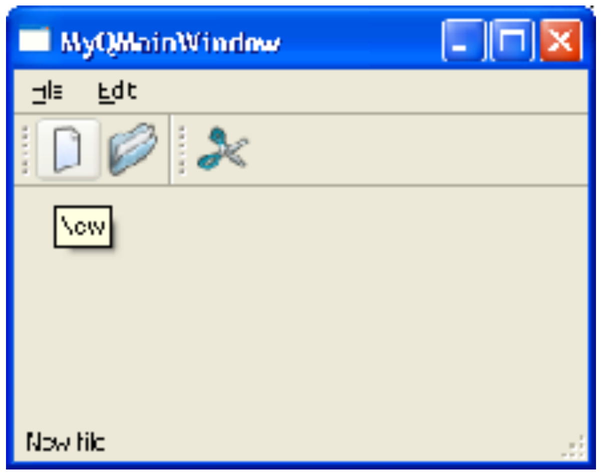
FileMenu = bar. **addMenu ("** File ")

NewACT = **QAction (QIcon (" path /** images / new.png ")," New ... ", None) NewACT. **SetShortCut (** "Ctrl + N") NewACT. **setToolTip (** tr ("New File")) NewACT. **setStatusTip (** tr ("New file"))

filemenu. **addAction (** NewACT)

newAct.triggered.connect (self.open)

QMainWindow



QMenuBar, QMenu, QAction

QToolBar

- fi = leToolBar QToolBar ("File")
- fi leToolBar. **addAction (**NewACT)
- NewACT. **setEnabled (**false)

inaccessible (grayed out) in the control menus and toolbar

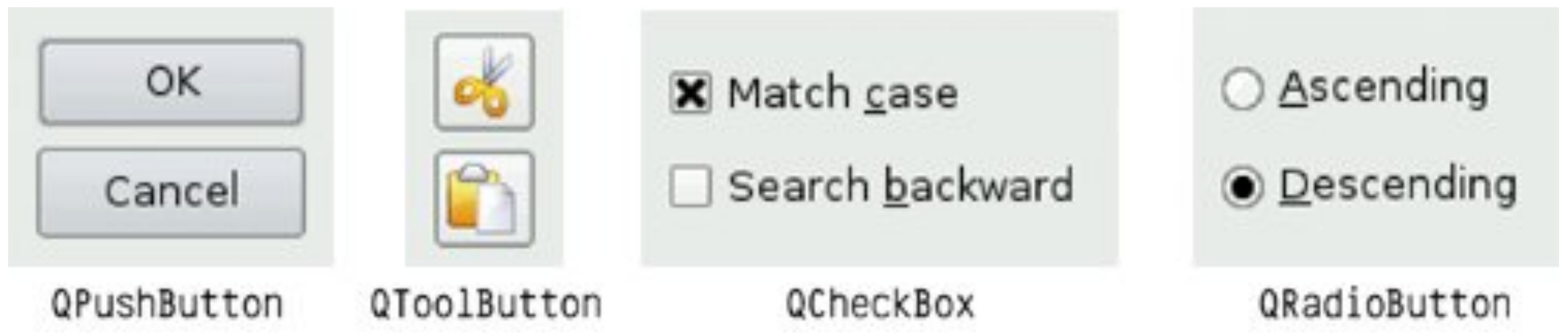
QToolTip, QWhatsThis

central component

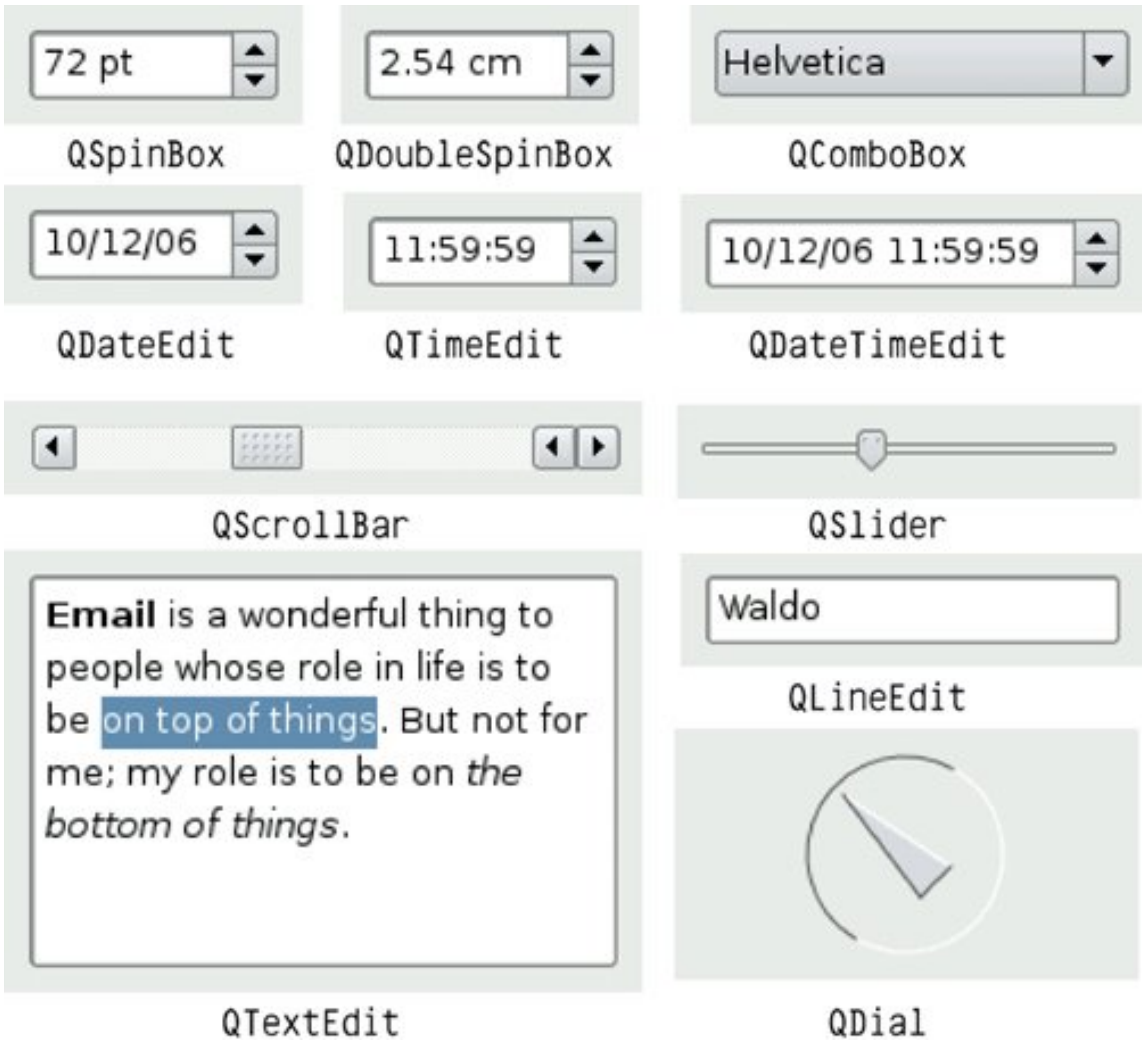
textEdit = TextEdit (self); self. **setCentralWidget**

(textEdit);

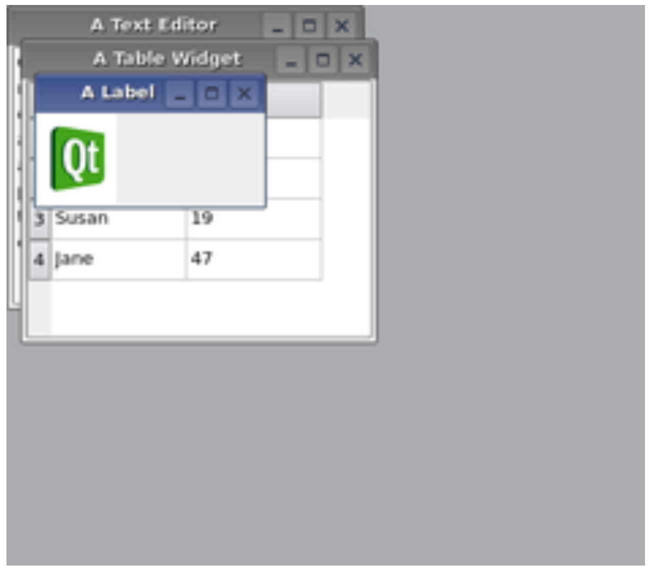
Buttons


OK
Cancel
QPushButton
QToolButton
Match case
Search backward
QCheckBox
Ascending
Descending
QRadioButton

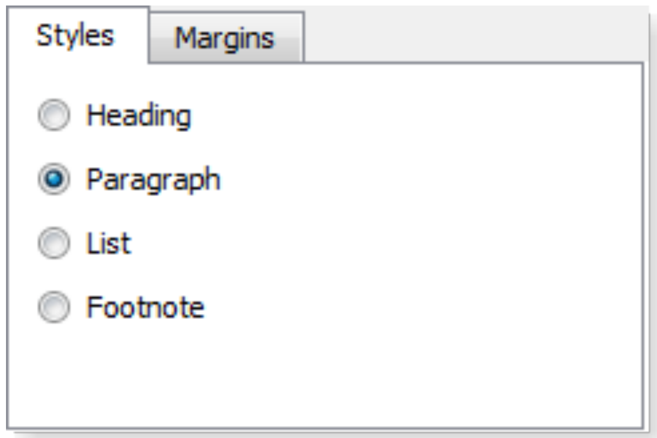
Input Widgets



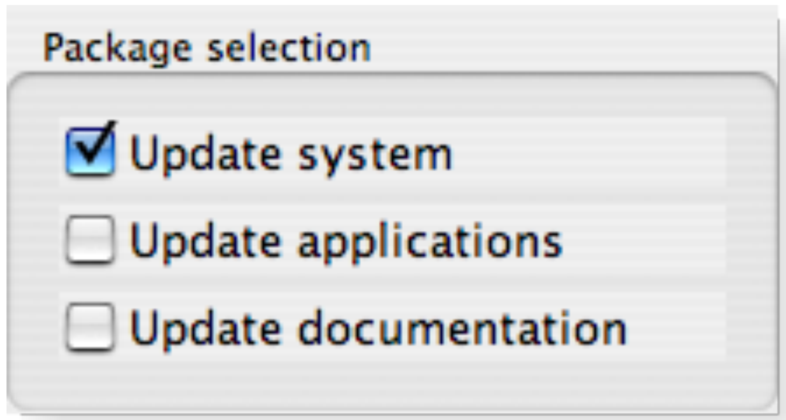
containers



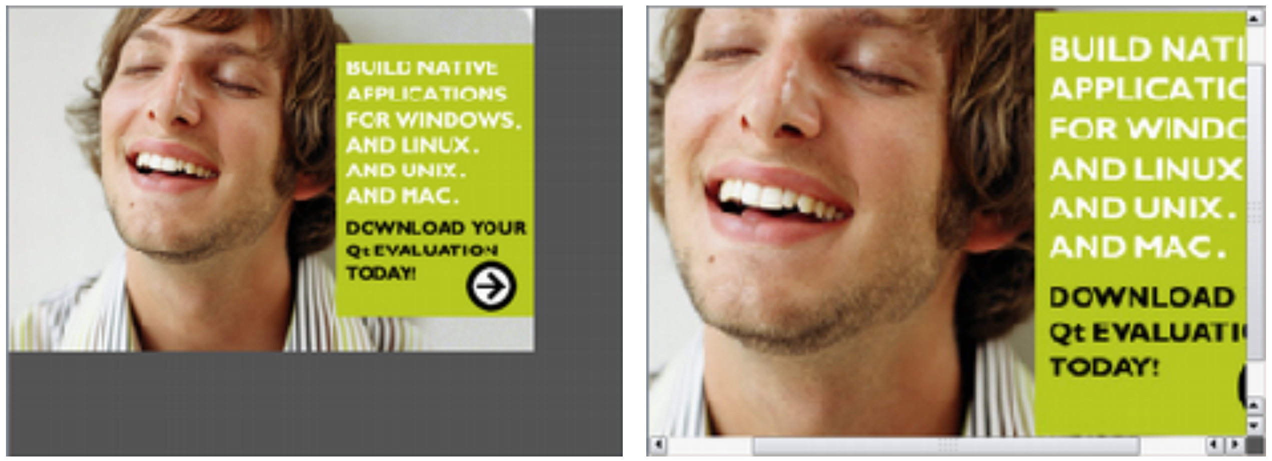
QMidArea



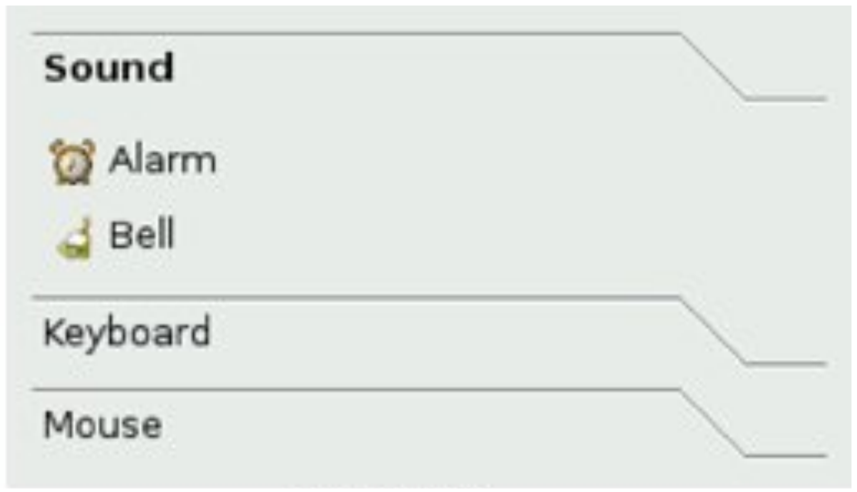
QTabWidget



QGroupBox


BUILD NATIVE APPLICATIONS FOR WINDOWS, AND LINUX, AND UNIX, AND MAC.
DOWNLOAD YOUR Qt EVALUATION TODAY!

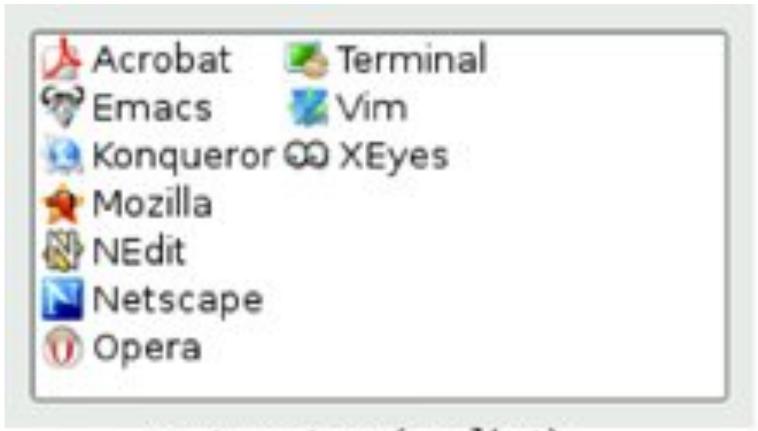
QScrollArea



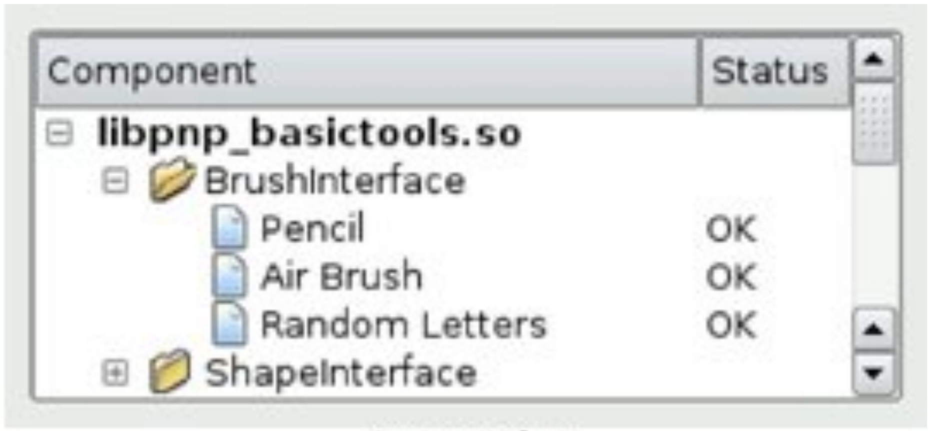
QToolBox

QWidget ; QFrame; QDockWidget; QStackedWidget

Views

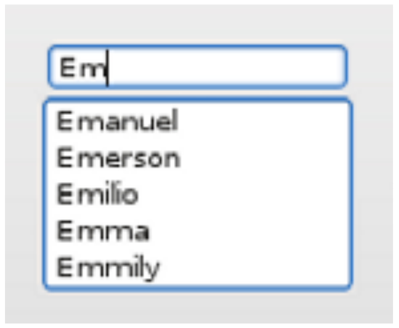


QListView (as list)

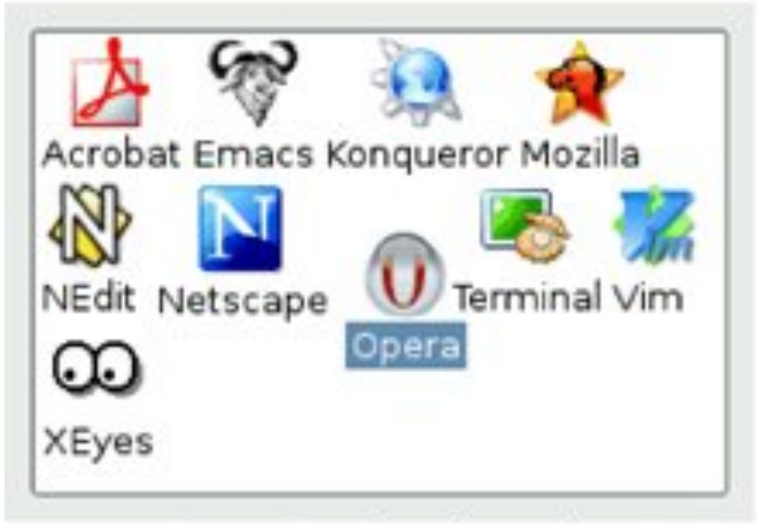


Component	Status
libpnp_basictools.so	
BrushInterface	
Pencil	OK
Air Brush	OK
Random Letters	OK
ShapeInterface	

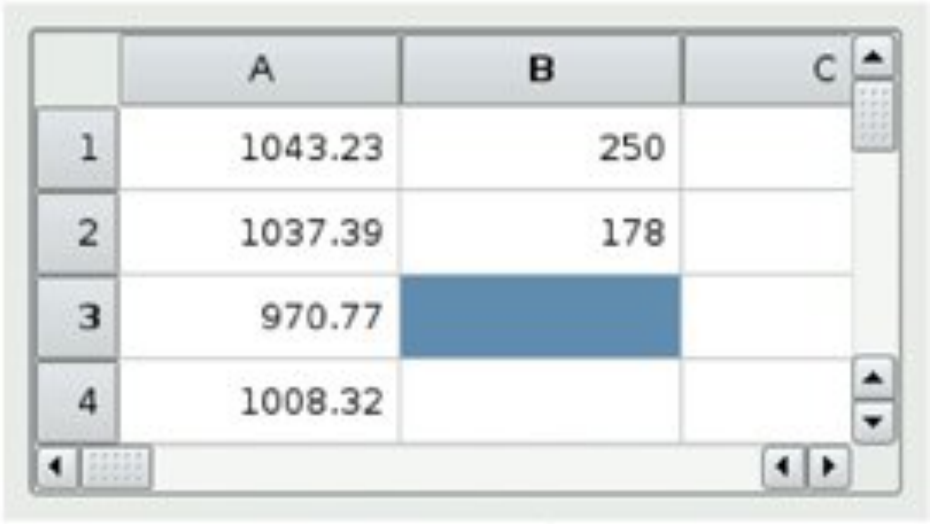
QTreeView



QCompleter



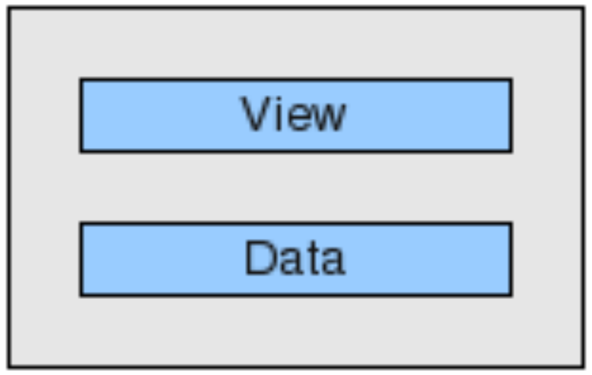
QListView (as icons)



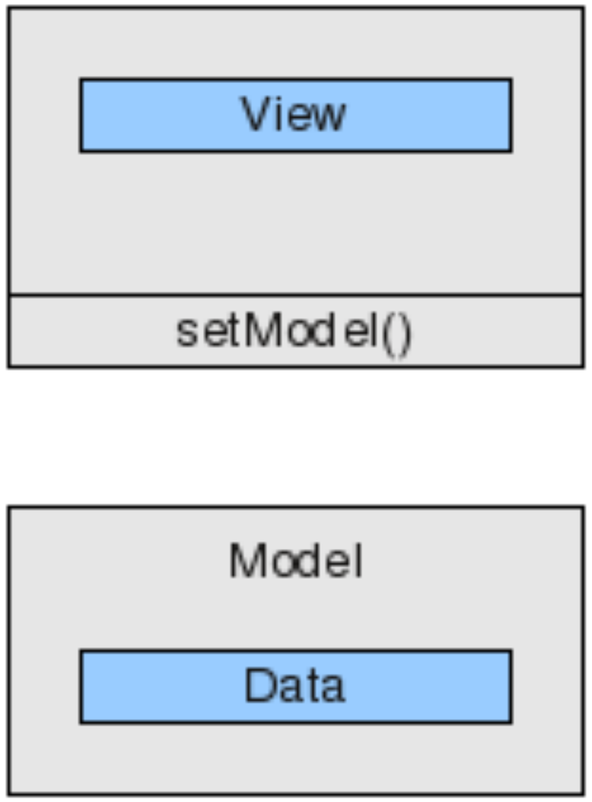
	A	B	C
1	1043.23	250	
2	1037.39	178	
3	970.77		
4	1008.32		

QTableView

Standard widgets use data That Is of the share widget



View classes we operate external data (the model)

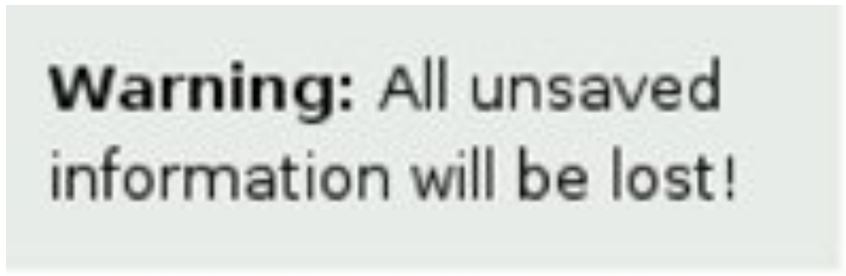


```
def hand( args ):
       app = QApplication (args) tableView =
       QTableView () = myModel MyModel ()

       tableView.setModel (myModel) tableView.show ()
       app.exec ()
```

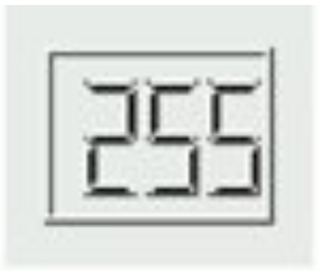
```
class MyModel ( QAbstractTableModel ):
    def __init __ (self):
        QAbstractTableModel .__ init __ (self) self.myData =
        <dataBase>


    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 hand( args ):
    app = QApplication (args) tableView =
    QTableView () = myModel MyModel ()

    tableView.setModel (myModel) tableView.show ()
    app.exec ()
```

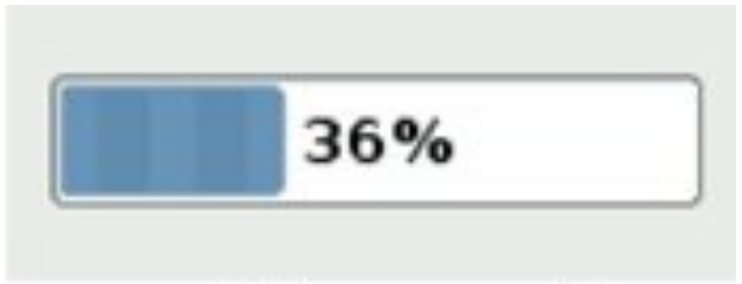
Display Widgets


Warning: All unsaved information will be lost!

QLabel (text)

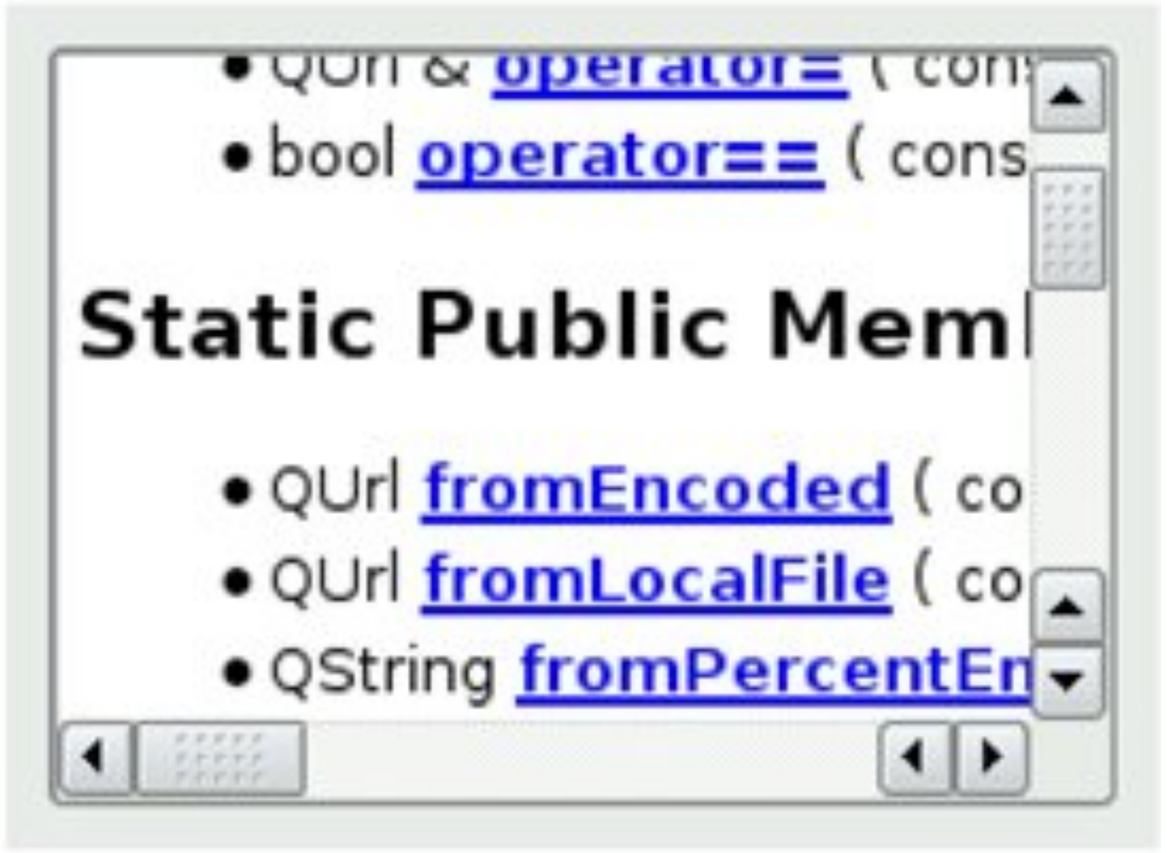


QLCDNumber



QProgressBar

QLabel (image)

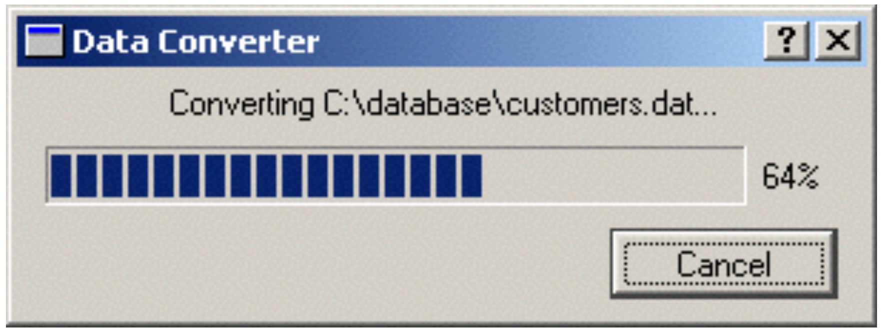

• QUrl & operator= (cons
• bool operator== (cons

Static Public Mem

• QUrl fromEncoded (co
• QUrl fromLocalFile (co
• QString fromPercentEn

QTextBrowser

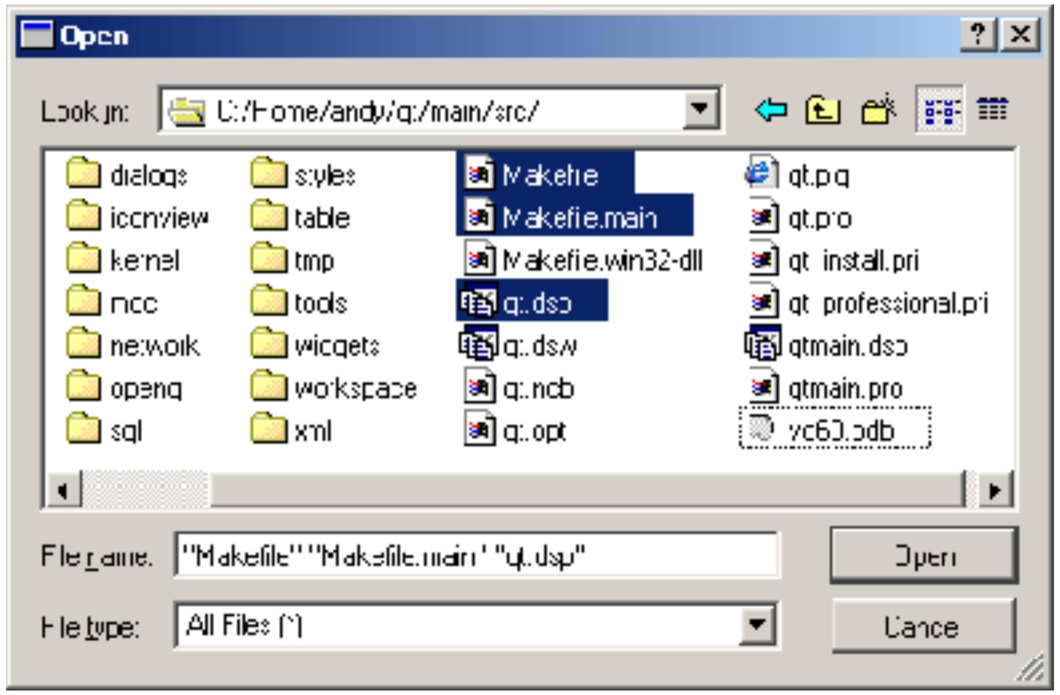
Dialog Boxes



Converting C:\database\customers.dat...

64%

QProgressDialog



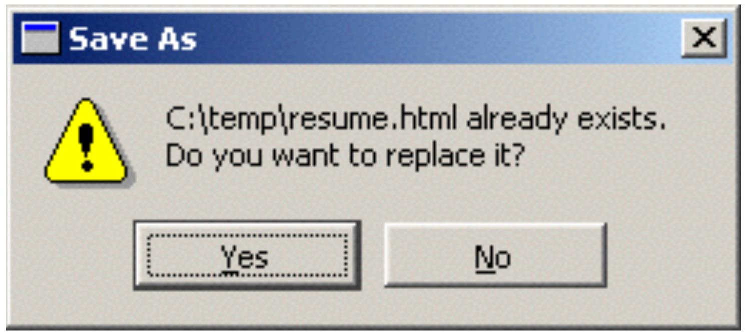
Look in: C:/home/andy/qt/main/src/

dialogs, iconview, kernel, moc, network, opengl, sql, styles, table, tmp, tools, widgets, workspace, xml, Makefile, Makefile.main, Makefile.win32-dll, qt.dsp, qt.dsw, qt.ncb, qt.opt, qt.pro, qt.pro, qt_install.pri, qt_professional.pri, qtmain.dsp, qtmain.pro, vc60.pdb

File name: "Makefile" "Makefile.main" "qt.dsp"

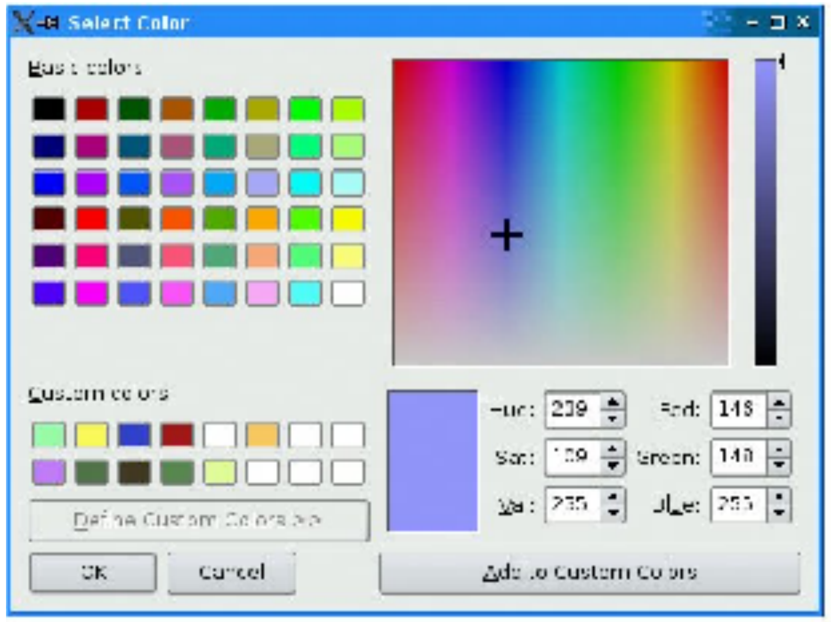
File type: All Files (*)

QFileDialog



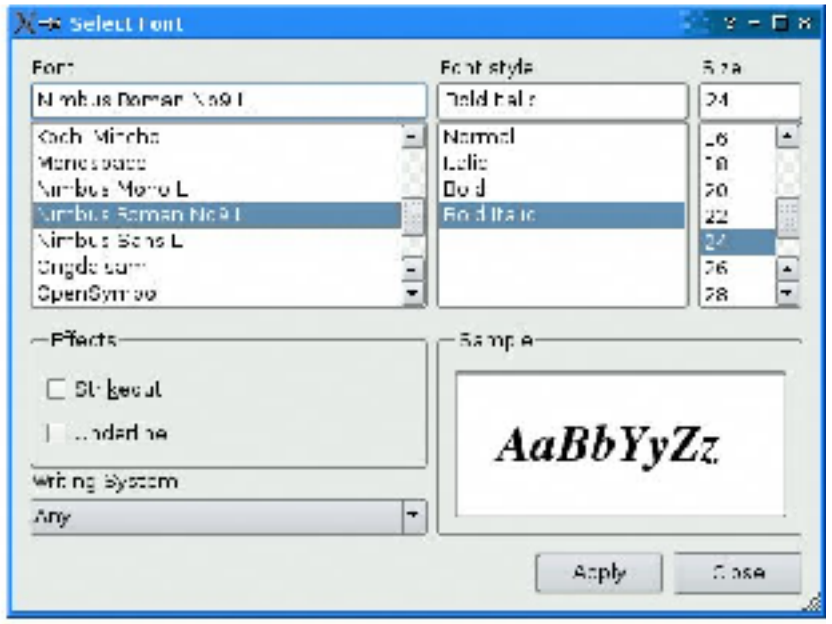
C:\temp\resume.html already exists.
Do you want to replace it?

QMessageBox



Hue: 239 Red: 148
Sat: 109 Green: 148
Val: 255 Blue: 255

QColorDialog



Font: Nimbus Roman No9 L
Font style: Bold Italic
Size: 24

Kochi Mincho, Monospace, Nimbus Mono L, Nimbus Roman No9 L, Nimbus Sans L, Lingdo sum, OpenSymbol

Normal, Italic, Bold, Bold Italic

16, 18, 20, 22, 24, 26, 28

Strikeout
Underline

Writing System: Any

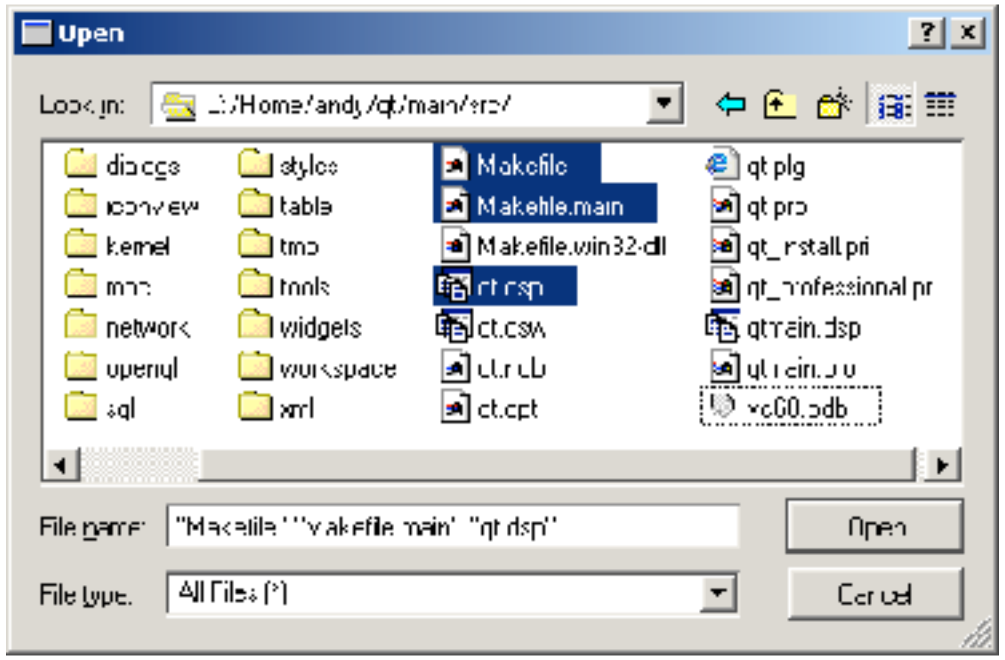
AaBbYyZz

QFontDialog

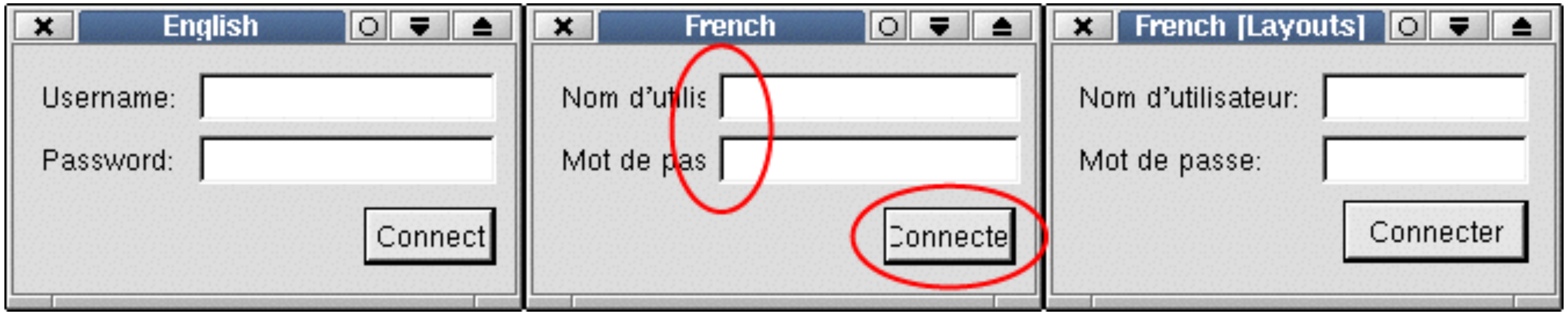
modal dialog box

simplified solution

```
fileName = QFileDialog. getOpenFileName ( self,                    // Parent
                                          "Open Image"            // title
                                          "/ Home / jana"         // initial directory
                                          "* .txt")               // filter

fileName = QFileDialog. getSaveFileName ( ...)
```



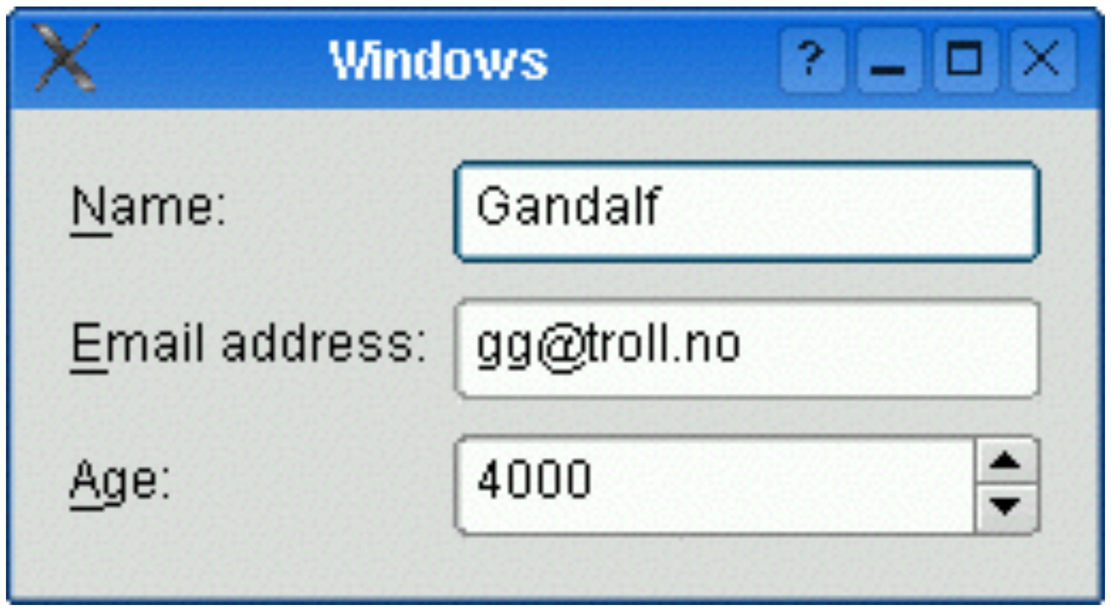
layout



Problems

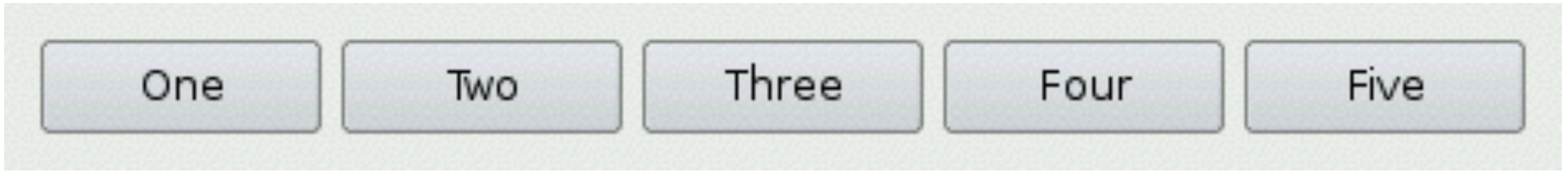
- internationalization
- resizing
- code complexity

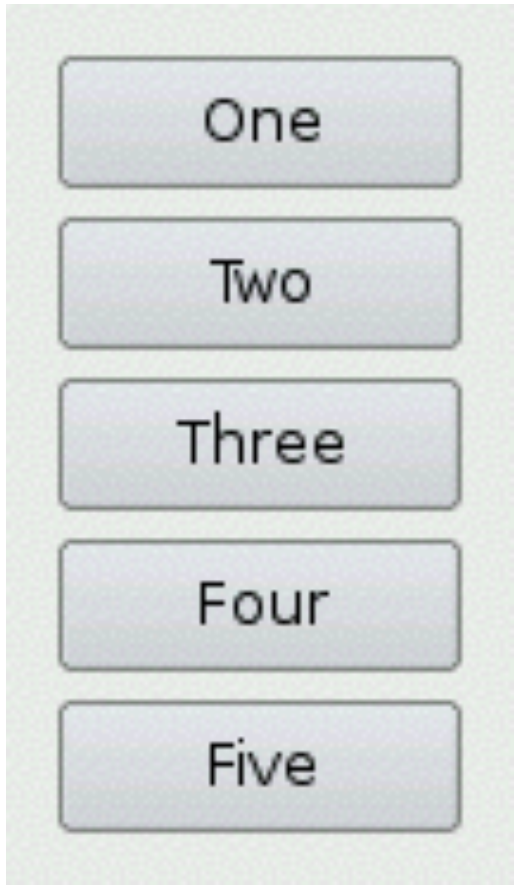
layout

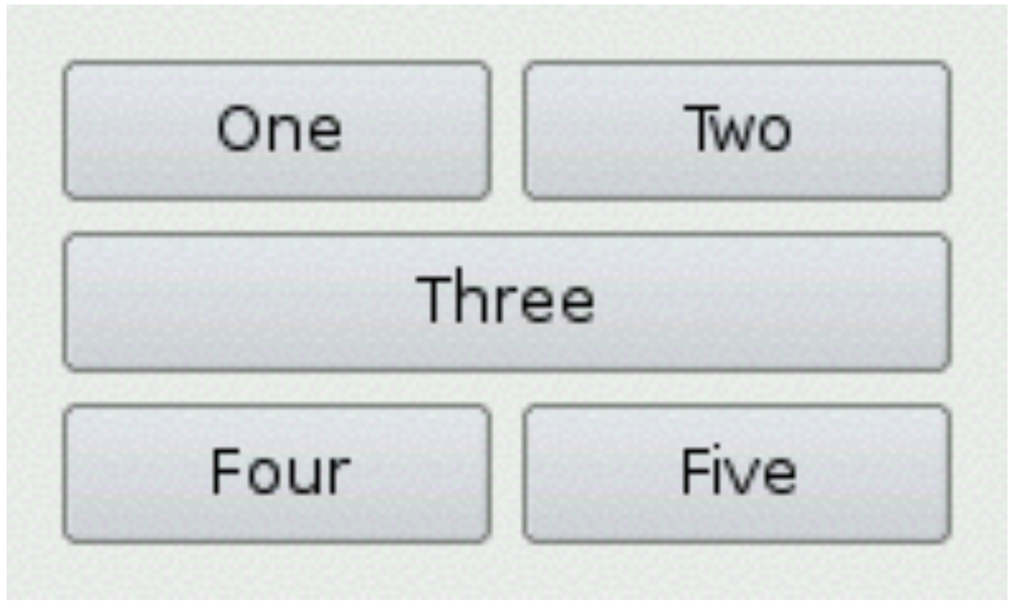


QFormLayout

QHBoxLayout







QVBoxLayout

QGridLayout

layout

Example

v_layout = QVBoxLayout () V_layout. addWidget (QPushButton ("OK"))
v_layout. addWidget (QPushButton ("Cancel")) v_layout. **addStretch ( )**

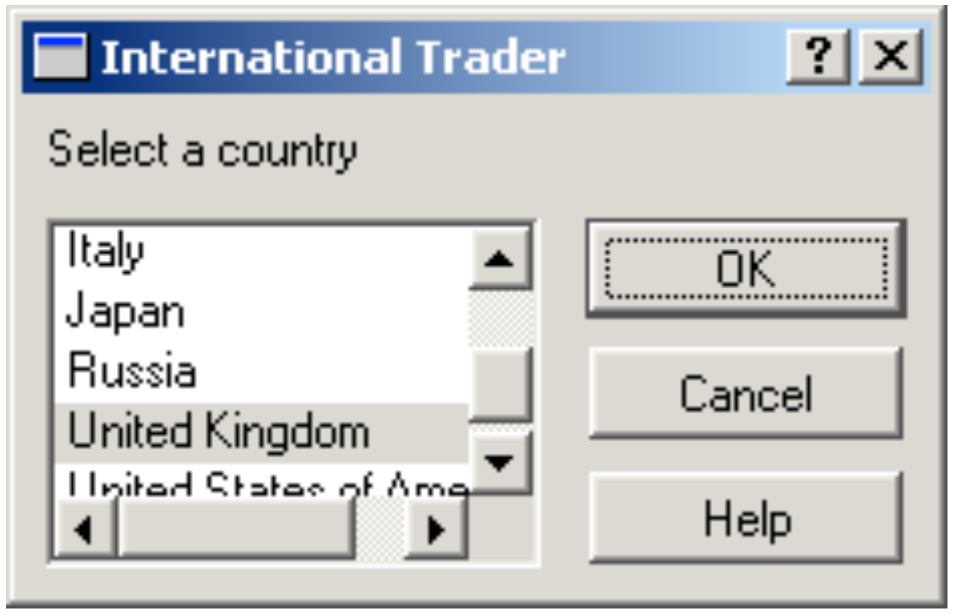
v_layout. addWidget (QPushButton ("Help"))

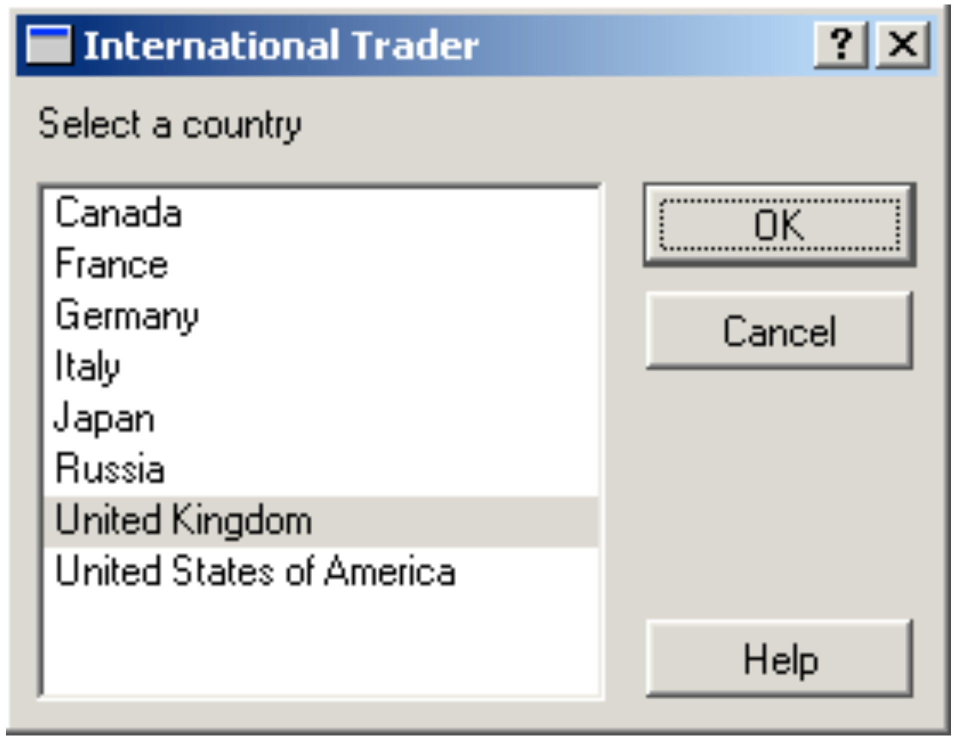
COUNTRY_LIST QListBox = (); countryList.insertItem (
"Canada");
. . . etc ...

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 layouts:

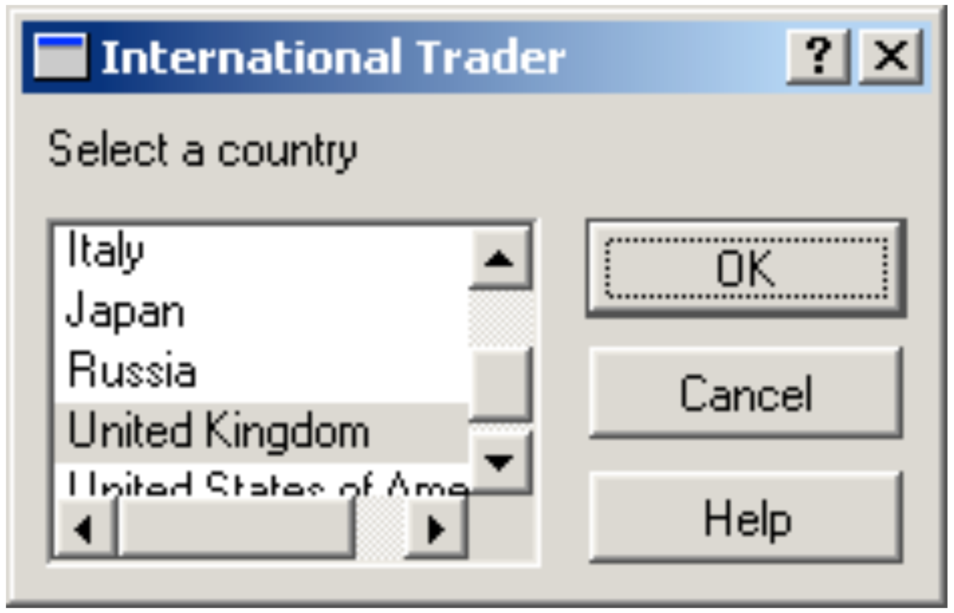
- can be nested
- not linked to a hierarchy of containers such as Java
- cf. the "stretch"

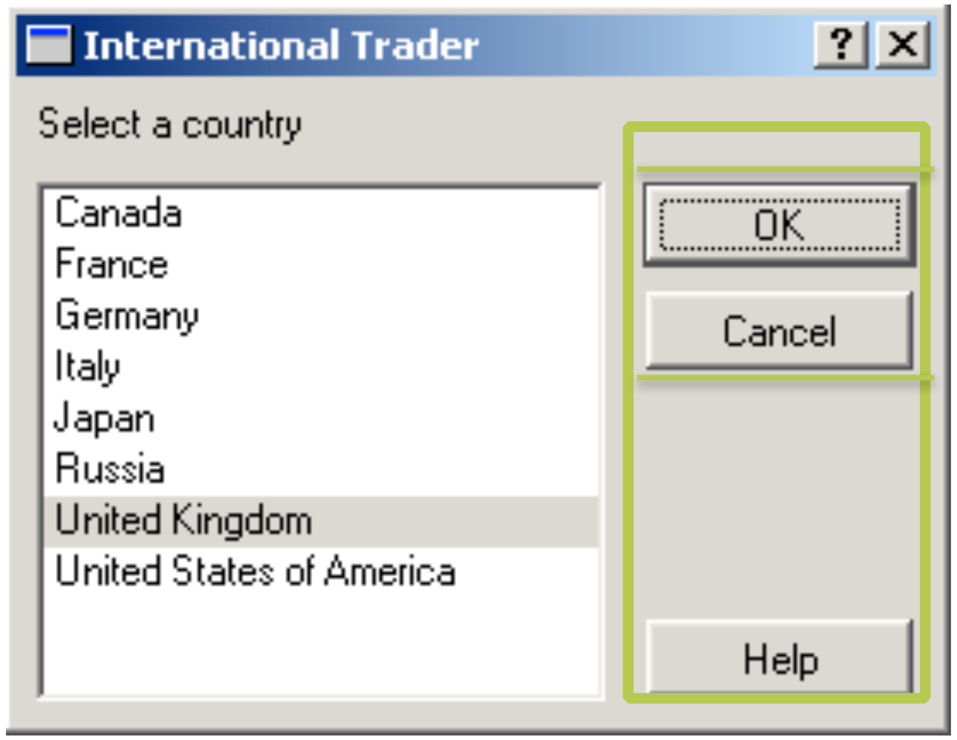
layout

Example

```
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 layouts:

- can be nested
- not linked to a hierarchy of containers such as Java
- cf. the "stretch"

layout

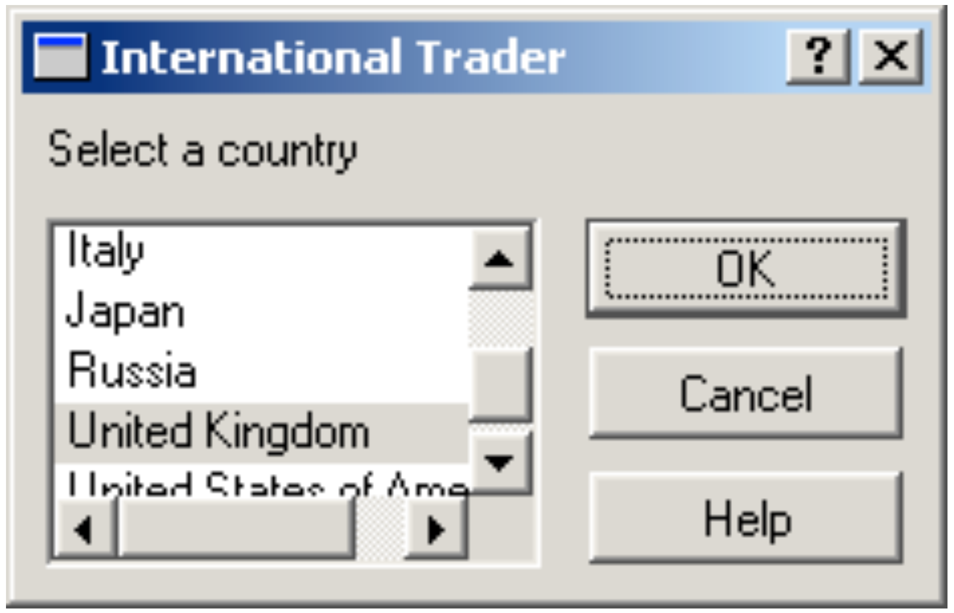
Example

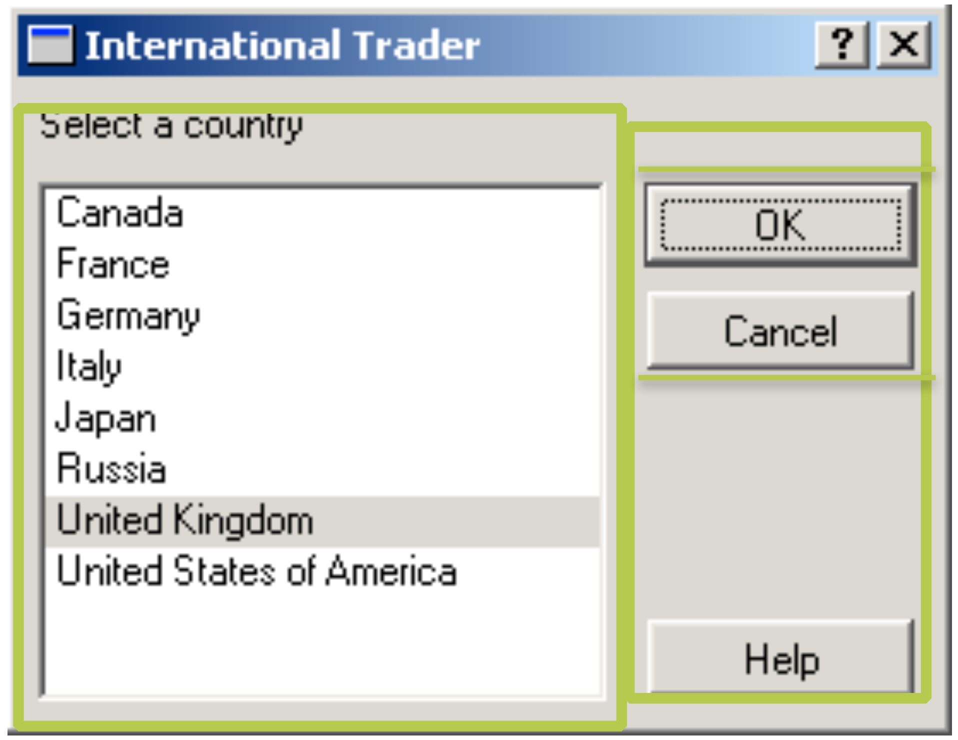
```
v_layout = QVBoxLayout () V_layout. addWidget (QPushButton ( "OK"))
v_layout. addWidget (QPushButton ( "Cancel")) v_layout. addStretch ( )


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 ()
```





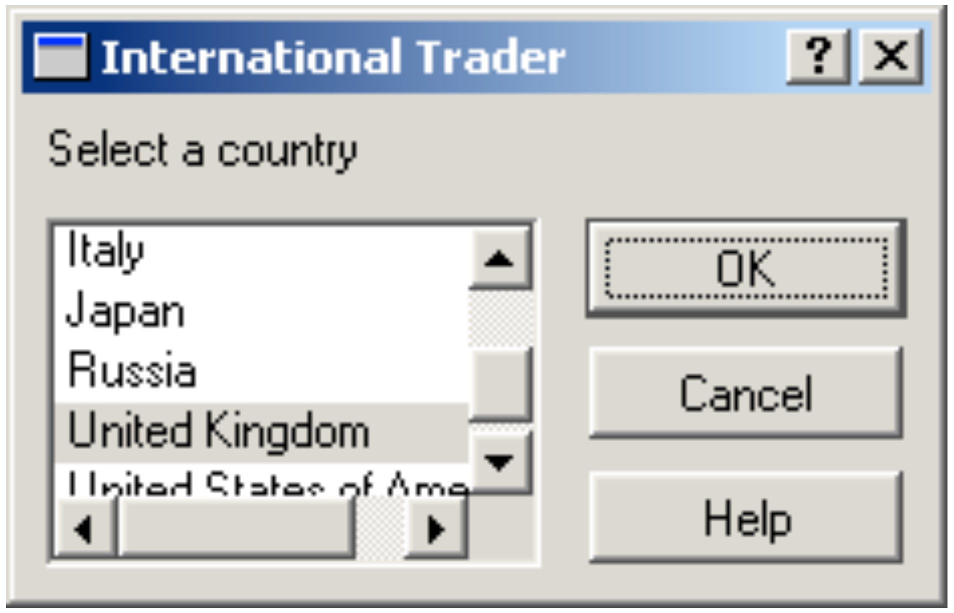
Notes layouts:

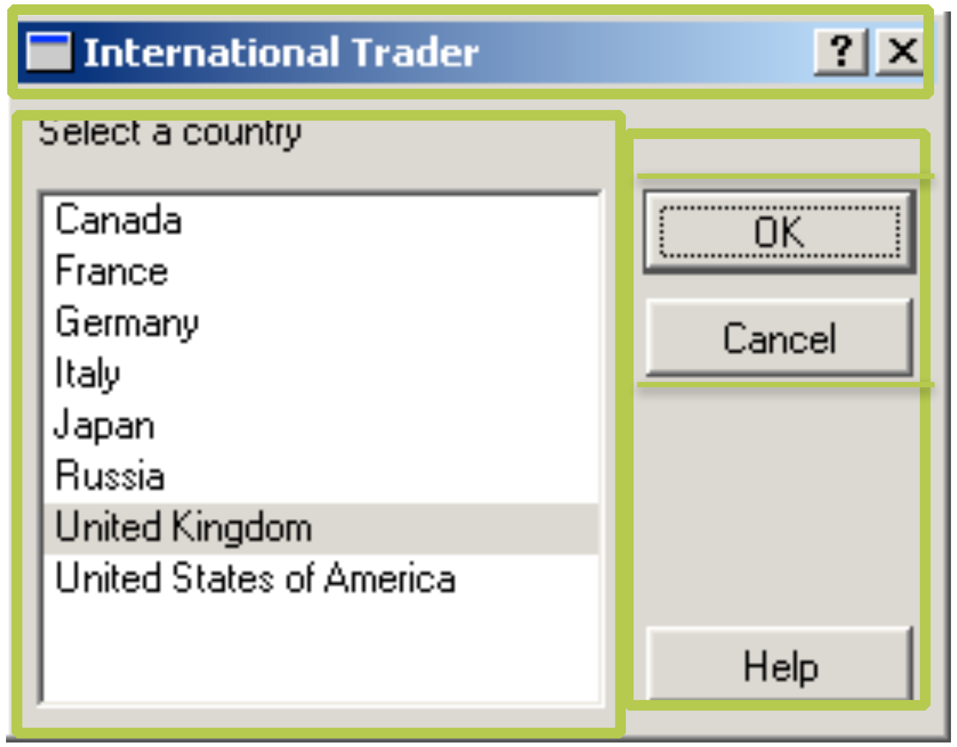
- can be nested
- not linked to a hierarchy of containers such as Java
- cf. the "stretch"

layout

Example

```
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 layouts:

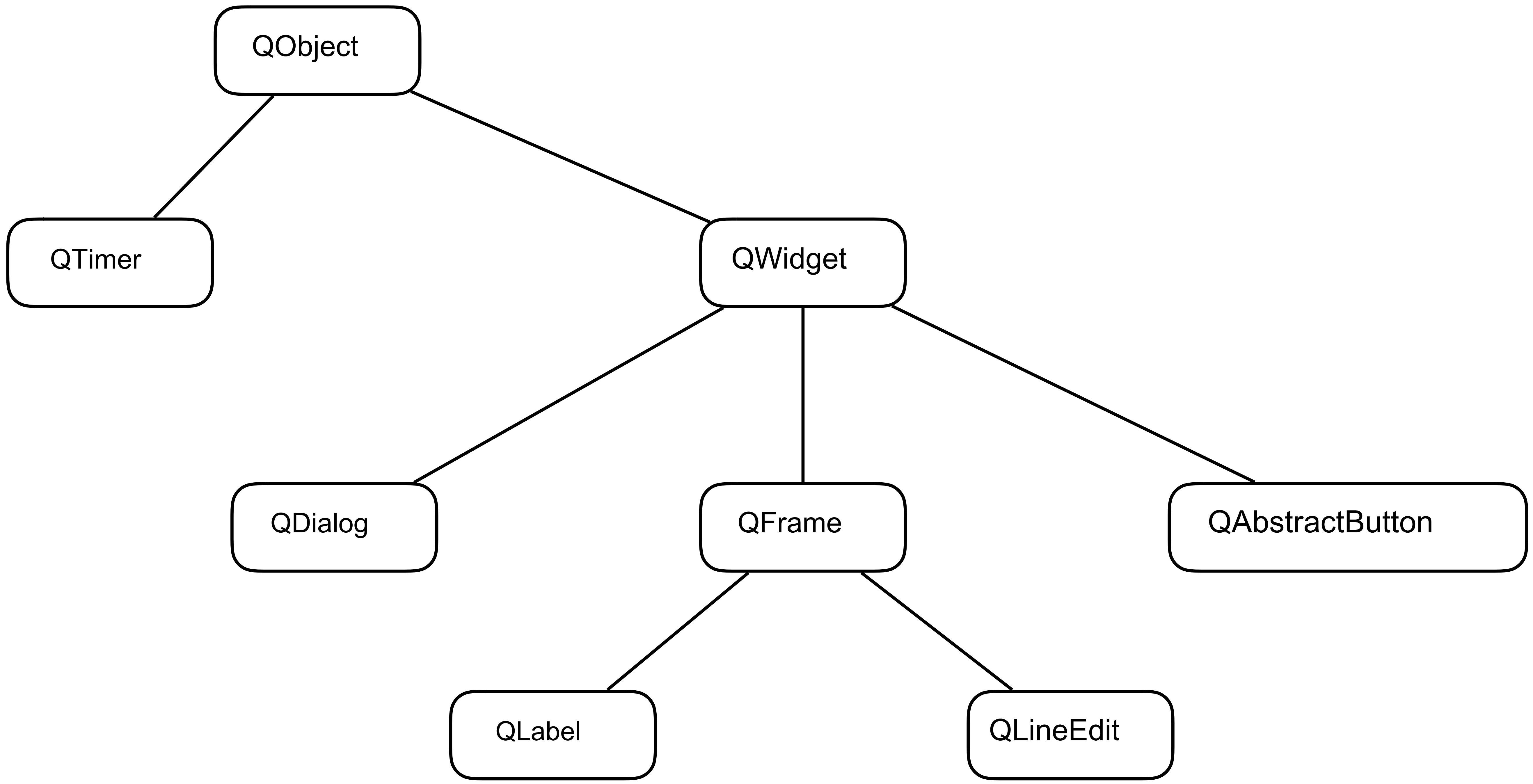
- can be nested
- not linked to a hierarchy of containers such as Java
- cf. the "stretch"

inheritance tree

vs.

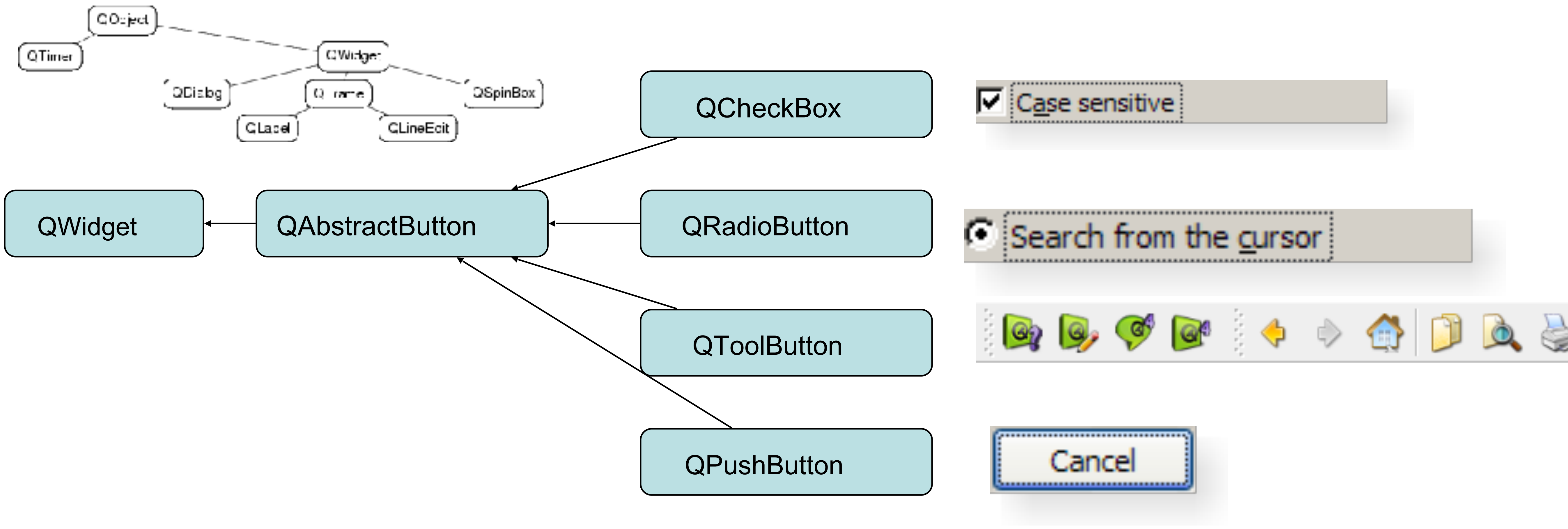
shaft instantiation

Trees heritage



main widgets

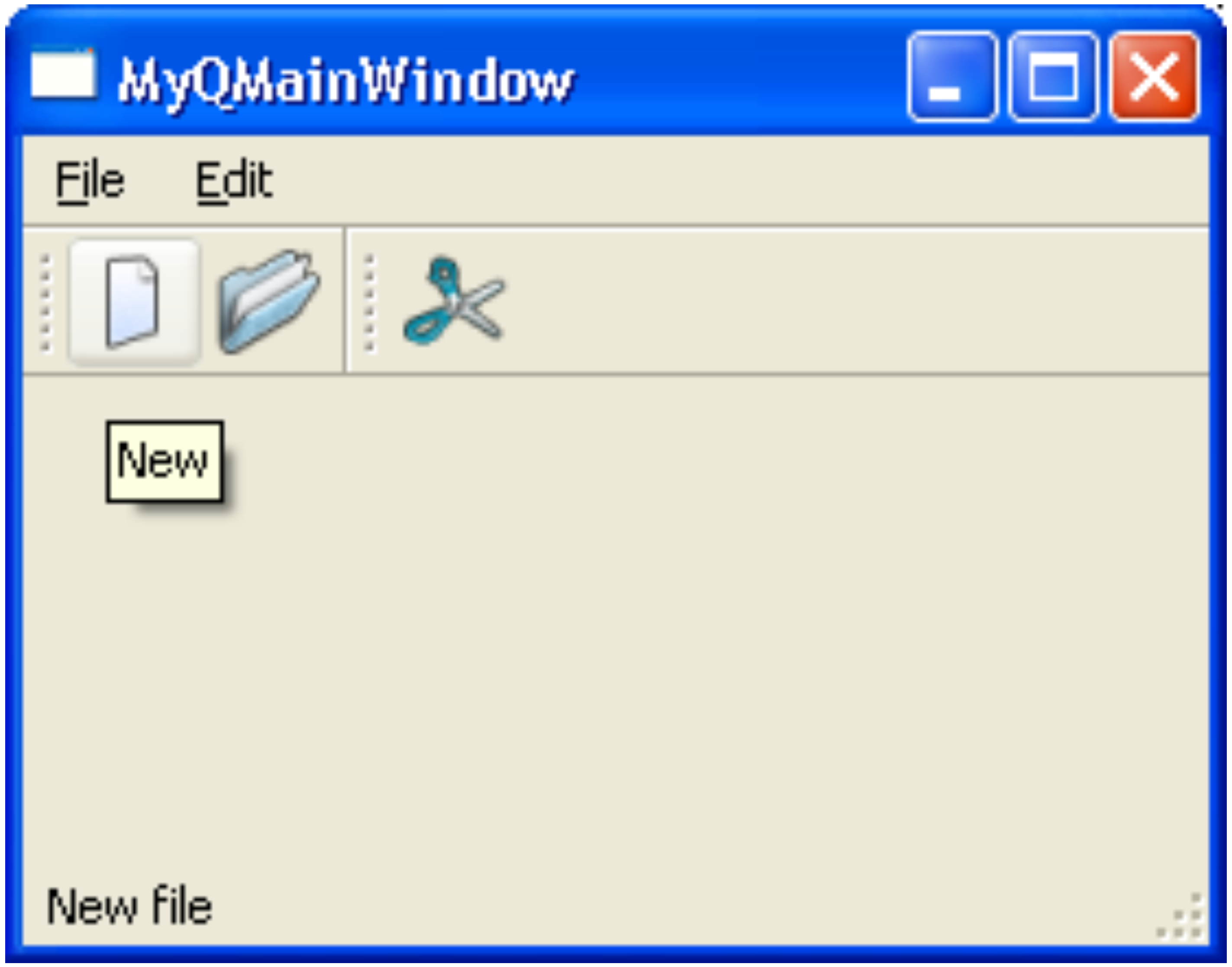
Tree heritage



Tree instantiation

Instance Hierarchy (= objects)

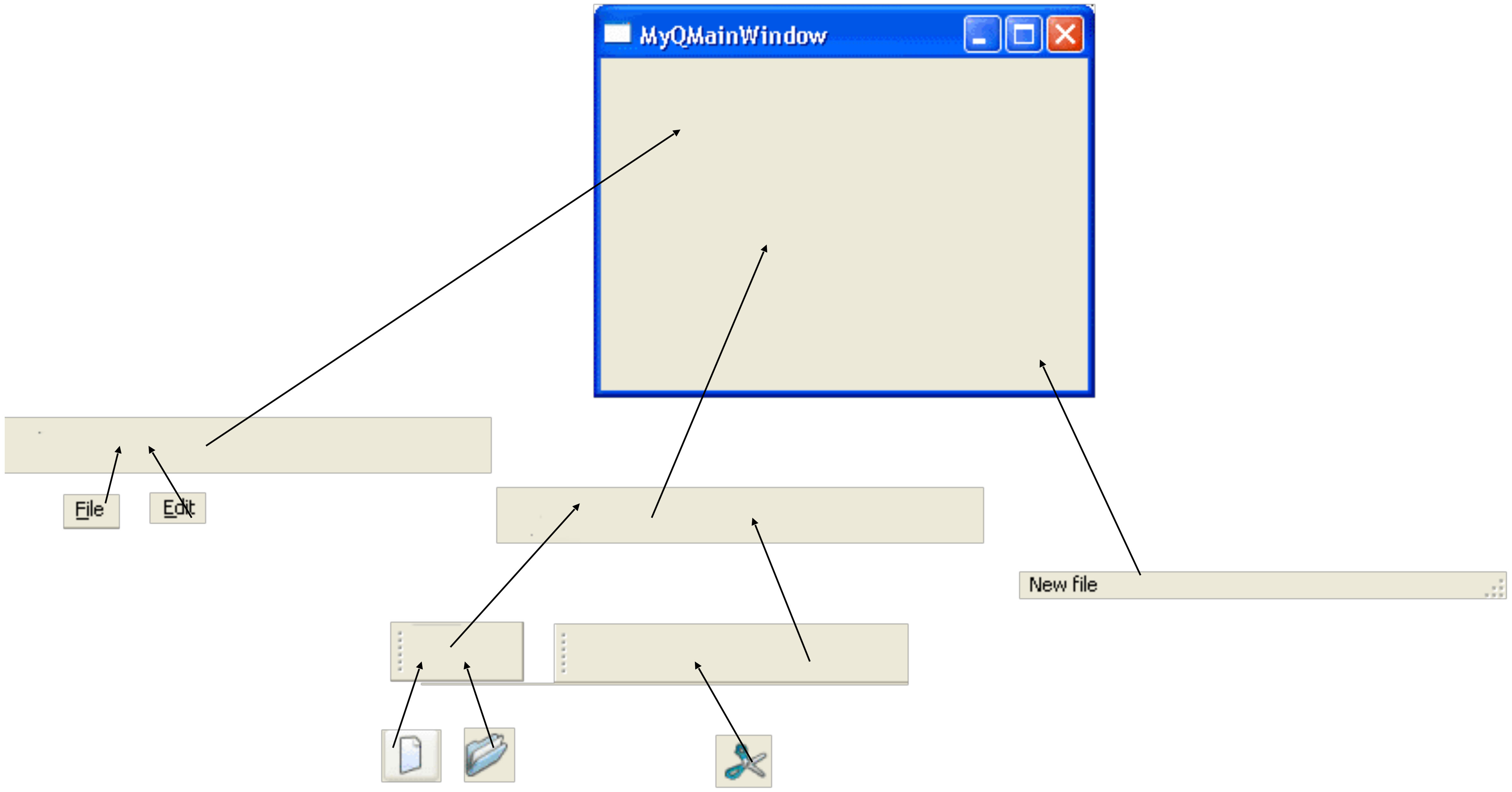
- Tree fi liation objects



Tree instantiation

Instance Hierarchy (= objects)

- Tree fi liation objects



Tree insanciation

The children say with its parent (≠ java)

- label = QLabel ("Hello", parent);
- exceptions
  - QFile, QApplication ...

If the parent of a Widget is zero, the widget is a window (Window). What do

parents do?

- They have a child list
- They automatically destroy the children when they are destroyed
- Enable / disable children when they enable / disable them same
- Same for Show / Hide

Tree instantiation

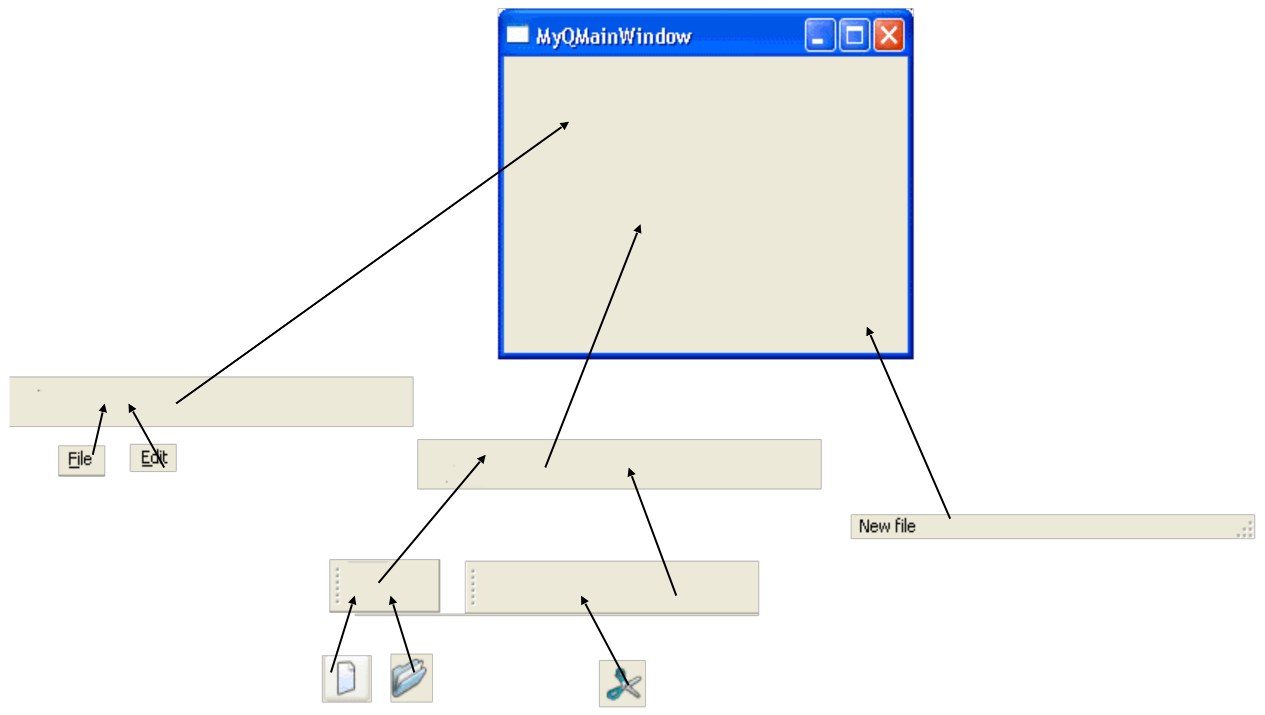
Instance Hierarchy (= objects)

- Tree fi liation objects

Each object contains its children

- Clipping: Children included in parents'
- Overlay: children over parents

An object has only one parent

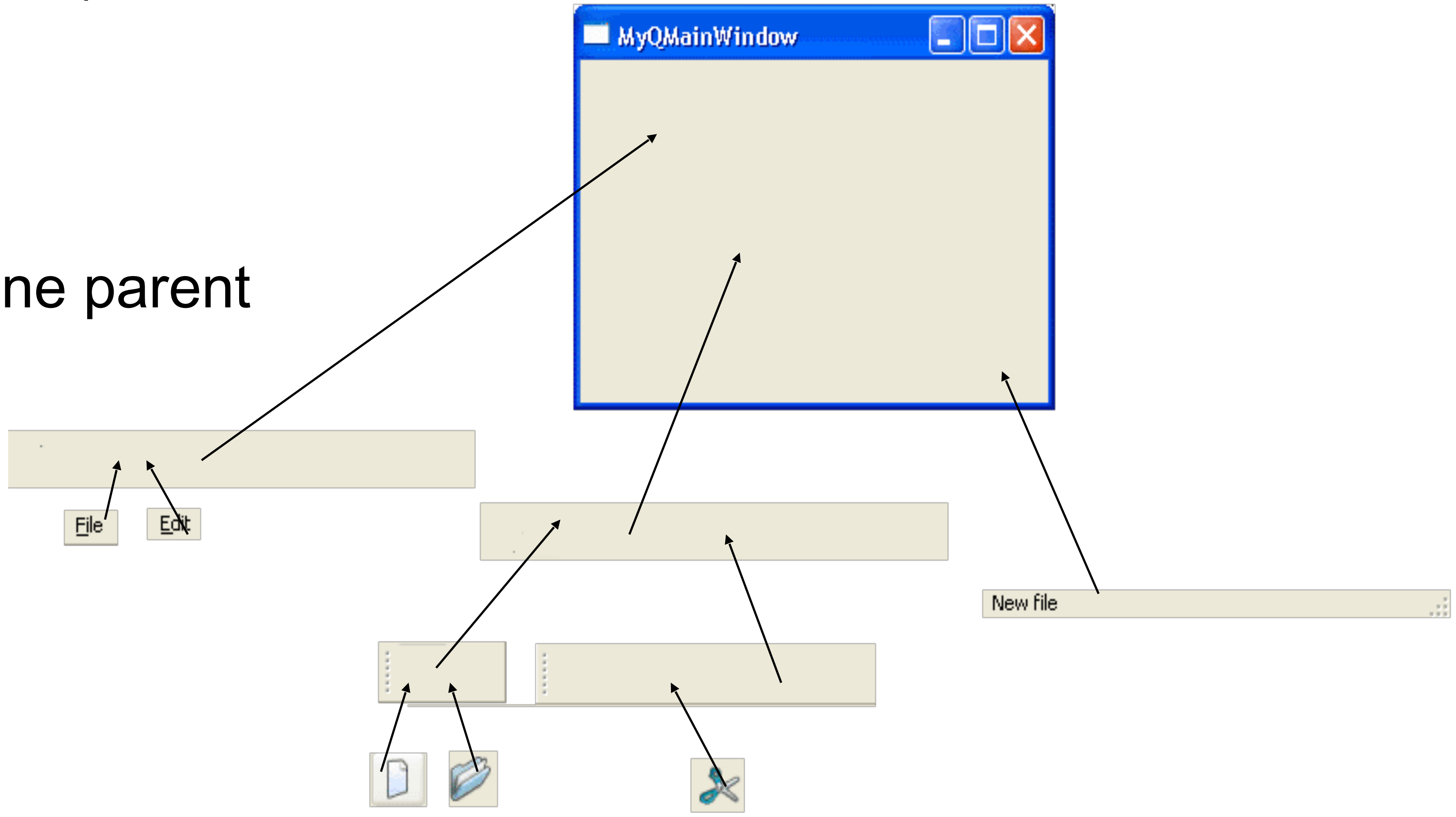


Tree instantiation

- The children say with its parent (≠ java)
  - **label = QLabel ("Hello", parent);**
  - execptions
    - QFile, QApplication ...
- If the parent of a Widget is zero, the widget is a window (Window).
- What do parents do?
  - They have a child list
  - They automatically destroy the children when they are destroyed
  - Enable / disable children when they enable / disable them same
  - Same for Show / Hide

Tree instantiation

- Instance Hierarchy (= objects)
  - parentage tree objects
- Each object contains its children
  - Clipping: Children included in parents'
  - Overlay: children over parents
- An object has only one parent


MyQMainWindow
File
Edit
New file