



UNIVERSITY OF
LIVERPOOL

Application Development with C++ (ELEC362)

Lecture 15: Qt Designer

mihasan@liverpool.ac.uk

Previous lecture

- The concept of developing a GUI application from classes in a library is discussed.
- The concept of widgets, parent and child relation were introduced.
- The first application has been built.
- The object-object connection and the user interface-object connections were discussed.
- Classes introduced: `QWidget`, `QDialog`, `QMainWindow`, `QDebug`, `QLineEdit`, `QPushButton`, `QPlainTextEdit`, `QString`

This lecture

- What is covered in this lecture?
 1. Using Qt designer to improve the application's user interface and behaviour.
 2. Signal-Slot mechanism
- Why it is covered?
 1. An application with a very good user interface have much higher chances of commercial success.
 2. The signal-slot mechanism is very efficient method for object-object communications.
- How are topics covered in this lecture: Live demonstration

What is Qt Designer

- Qt Designer is the tool used to design and generate user interface (ui). It can be launched by clicking on any of the files with .ui extension under forms of the project tree.
- Reference: <https://doc.qt.io/qt-5/qtdesigner-manual.html>
- In addition to its use as a source for widgets in a Ui, it can also be used for:
 1. Controlling the layout of the application (how it looks on different screens and at different sizes).
 2. Controlling the stylesheet of the widgets (their colour, appearance, style, etc..).
 3. Quick implementation of the signal-slot mechanism.
 4. Adding external resources to the project.


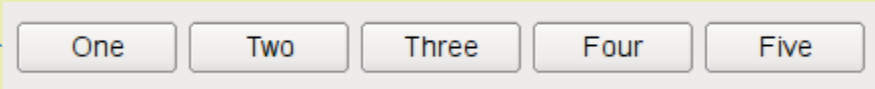

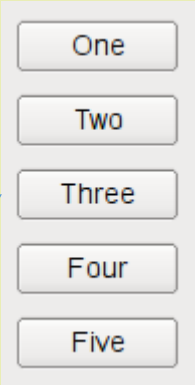


Classes interacting with Qt Designer

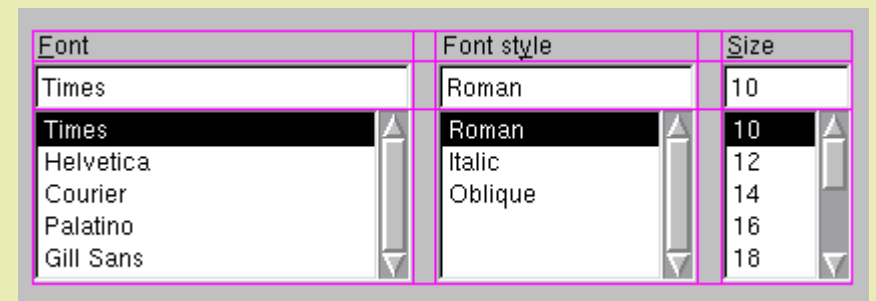
- Qt Designer walkthrough (demonstration).
- By setting options through Qt Designer, we are implicitly manipulating many classes related to the look and the style of the application, including:

Class	What it does
QLayout	Controls the arrangement of the child widgets in the parent widget and how they change when size/window change.
QStyle	Controls the appearance of individual widgets.

- Fortunately, Qt Designer sets all the options for these classes automatically!
- Anything done in Qt Designer is transformed into a code that is hidden from the developer.

QLayout Class

- This class is responsible for managing how the shape of the widgets change when the window is resized, or is drawn on a different size screen.
- Using Qt Designer we can set all the parameters for the layout without coding!!
- The class **QLayout** is not used on its own, instead its subclasses are used:
- **QHBoxLayout** : is used to construct horizontal box of widgets.

- **QVBoxLayout** : is used to construct vertical box of widgets.

- **QGridLayout** : divides the parent widget into rows and columns, and puts each widget it manages into the correct cell. 




Size control of widgets

- The layout distributes the space on the parent widget among child widgets based on their size policies.
- The size policy of a widget can be changed from this menu (bottom right of the screen).
- Every widget has a “sizeHint()” property, which holds the ideal size of the widget.
- Reference: <https://doc.qt.io/qt-5/qsizepolicy.html>

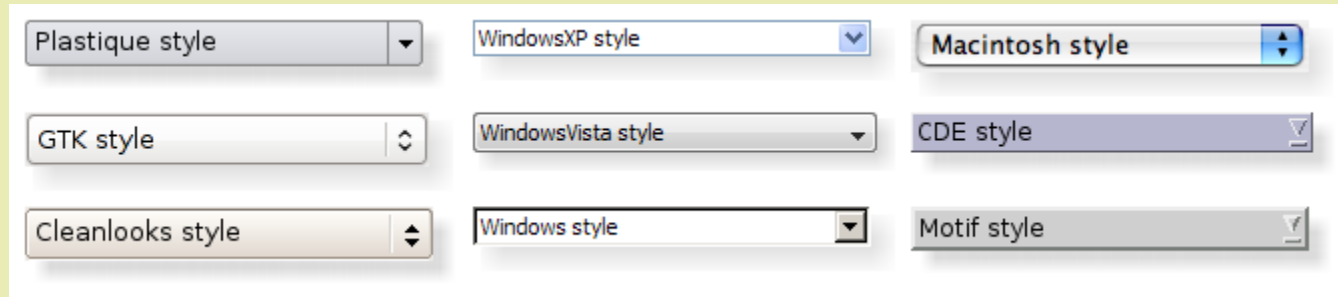
Good practice note:

After finalising a GUI make sure the size policy is set for all widgets.

QWidget	
enabled	<input checked="" type="checkbox"/>
> geometry	[(10, 70), 371 x 121]
> sizePolicy	[Expanding, Expanding, 0, 0]
Horizontal Policy	Expanding
Vertical Policy	Expanding
Horizontal Stretch	0
Vertical Stretch	0
> minimumSize	0 x 0
> maximumSize	16777215 x 16777215
> sizeIncrement	0 x 0
> baseSize	0 x 0
palette	Inherited
> font	A [MS Shell Dlg 2, 8]
cursor	 Arrow
mouseTracking	<input type="checkbox"/>
tabletTracking	<input type="checkbox"/>
focusPolicy	NoFocus
contextMenuPolicy	DefaultContextMenu
acceptDrops	<input type="checkbox"/>
> tooltip	

QStyle Class

- Qt Designer automatically chooses the Style of the application to make it look native to the platform it is operating on.



- Another example: <https://doc.qt.io/qt-5/gallery.html>
- How to change the style is here: <https://doc.qt.io/archives/qt-4.8/style-reference.html>

Good practice note:

A good application must be consistent with its native style

Style sheet of a widget

- Style sheets are appearance properties that each widget has. Changing the style sheet allows the developer to control the look of a widget.
- Every widget has a list of properties that can be customised, in addition to the allowed values can be found here:

<https://doc.qt.io/qt-5/stylesheets-reference.html#>

- To get you started quickly, here is a list of examples by widget:

<https://doc.qt.io/qt-5/stylesheets-examples.html>

Adding resource files to an application

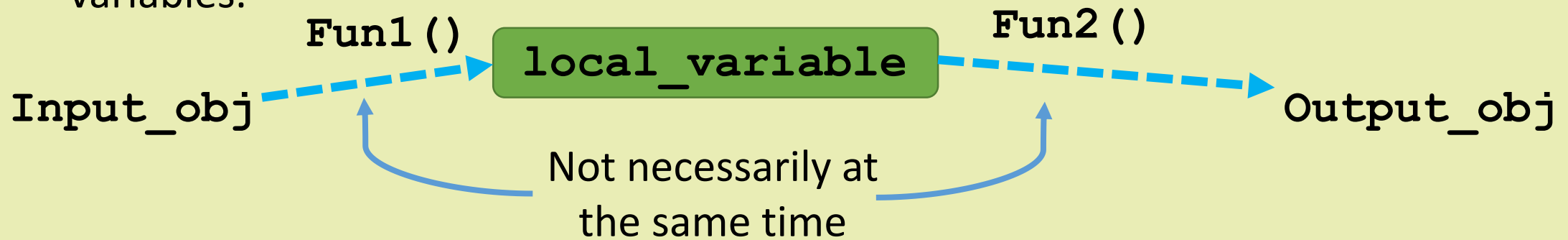
- What if you want more customisation options than Qt can offer?
- A resource file is data file that can be integrated into the executable file of the application and is accessible during runtime.
- Resources files can be pictures, colours, fonts, etc. Which can be imported into a Qt application to customise it.

Good practice note:

Make sure you use transparent pictures/icons as resources. If available use vector graphics (extension of .svg or .eps).

Signal - Slot mechanism

- In the previous lectures we discussed object-object communications using local variables.



- The signal-slot mechanism provides an instantaneous direct link between the two objects.




Signal - Slot mechanism

- A Signal is a function which emits/ transmits its output to all Slots of objects it is connected to automatically.
- A Slot is a function that takes the emitted signal as an input and is automatically executed when the signal is received.
- Every object in Qt has predefined signals and slots suitable for its purpose.
- The developer can define their own signals and slots as well:

```
// Somewhere in a header file:  
public slots:    // Can be public, private, or protected  
    //normal function prototype  
signals:        // has no access specifier  
    //normal function prototype
```

Signal - Slot connection

- A Slot function of an object can be dealt with as a normal function.
- The signals and slots of an object are separated from other functions in the documentation, example: <https://doc.qt.io/qt-5/qabstractbutton.html>
- To connect signals and slots from Qt Designer, press on this icon  on the top left
- Connecting a single to slot can be done in the code as well:

```
connect(sender_obj, SIGNAL(signal_fun), receiver_obj, SLOT(slot_fun))
```

- Further reading: <https://doc.qt.io/qt-5/signalsandslots.html>
- Search suggestions example (demonstration).

Summary

- In this lecture, further uses of Qt Designer were discussed.
- Formatting the lay out of the application and controlling its sizing were discussed.
- Controlling the style of a Widget was discussed.
- Adding resource files to an application was explained.
- The Signal-Slot mechanism was discussed and it was contrasted with connecting objects using variables.
- Classes introduced: `QLayout`, `QStyle`, `QHBoxLayout`, `QVBoxLayout`, `QGridLayout`