Introduction to Python GUI Programming with Qt

WEEK 2

Instructor: Ph.D. Oleg Tymchuk

Overview

Introduction to Qt

Qt modules and QObject

PyQt

Introduction to Qt

- Qt is a cross-platform development framework written in C++
- Can be used in several programming languages through bindings
  Ruby, Java, Perl, Python → PyQt
- Use the standard native tools to build Qt apps (IDE, debugger etc.)
- Qt provides a platform-independent encapsulation of the local window system and operating system
- The Qt API is identical on every platform, applications are compiled to native executables
- Qt is more than a GUI toolkit. It includes abstractions of network sockets, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, a fully functional web browser, a help system, a multimedia framework, as well as a rich collection of GUI widgets

Result: write once, compile everywhere

Introduction to Qt. Supported Platforms

Using Qt, you can write GUI applications once and deploy them across desktop, mobile and embedded operating systems without rewriting the source code

- Windows
- Linux/X11
- macOS
- Embedded Linux, QNX
- Android, iOS, Windows RT

Qt is supported on a variety of 32-bit and 64-bit platforms

Introduction to Qt. Development Tools

Tool	Description
Qt Creator	Qt tool which provides a cross-platform, complete integrated development environment
Qt Designer	Qt tool for designing and building graphical user interfaces (GUIs) with Qt Widgets
qmake	Qt tool which helps simplify the build process for development projects across different platforms
Qt Linguist	Qt tool for localizing applications
Qt Assistant	Qt tool for viewing on-line documentation in Qt help file format

Introduction to Qt. Customers

- Qt is used for a wide variety of applications: mass-market and custom software, internal apps, research, modeling, visualization, etc..

Qt modules and QObject. Qt modules

Module	Description
Qt Core	Core non-graphical classes used by other modules
Qt GUI	Base classes for graphical user interface (GUI) components. Includes OpenGL
Qt Multimedia	Classes for audio, video, radio and camera functionality
Qt Multimedia Widgets	Widget-based classes for implementing multimedia functionality
Qt Network	Classes to make network programming easier and more portable
Qt QML	Classes for QML and JavaScript languages
Qt Quick	A declarative framework for building highly dynamic applications with custom user interfaces
Qt Quick Controls	Reusable Qt Quick based UI controls to create classic desktop-style user interfaces
Qt Quick Dialogs	Types for creating and interacting with system dialogs from a Qt Quick application
Qt Quick Layouts	Layouts are items that are used to arrange Qt Quick 2 based items in the user interface
Qt SQL	Classes for database integration using SQL
Qt Test	Classes for unit testing Qt applications and libraries
Qt Widgets	Classes to extend Qt GUI with C++ widgets

Qt modules and QObject. QObject

The QObject class forms the foundation of Qt's object model and is the parent class of many Qt classes

Include these features:

- a very powerful mechanism for seamless object communication called signals and slots
- queryable and designable object properties
- hierarchical and queryable object trees that organize
- object ownership in a natural way with guarded pointers (QPointer)
- a dynamic cast that works across library boundaries

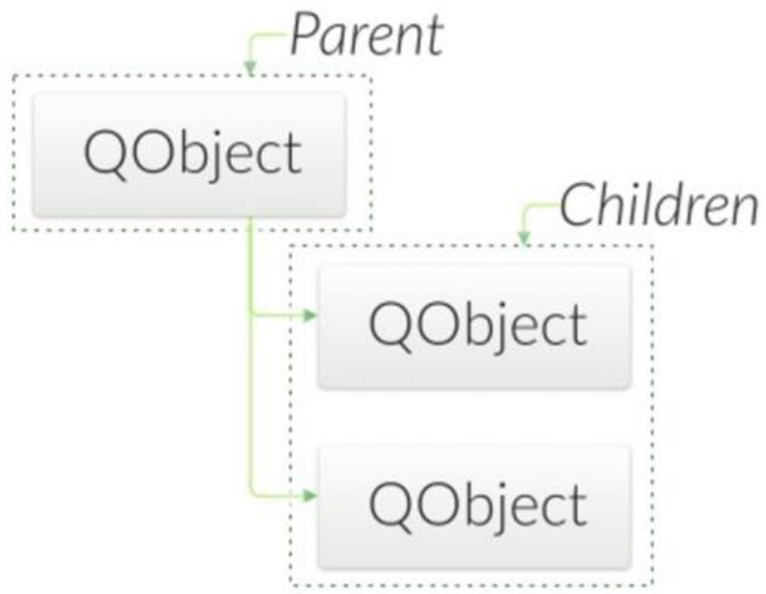
QObject has no visual representation

Qt modules and core types. Object Trees & Ownership

- QObjects organize themselves in object trees
- When you create a QObject with another object as parent, it's added to the parent's children() list, and is deleted when the parent is

  QObject(QObject *_parent_ = Q_NULLPTR)

- When any QObject in the tree is deleted, if the object has a parent, the destructor automatically removes the object from its parent



Parent-child relationship IS NOT inheritance!

PyQt

PyQt is a set of Python v2 and v3 bindings for The Qt Company's Qt application framework

- bindings are implemented as a set of Python modules and contain over 1,000 classes
- almost the entire Qt library is available
- PyQt5 supports Qt v5 (the Qt Company no longer supports Qt v4)
- PyQt is dual licensed on all supported platforms under the GNU GPL v3 and the Riverbank Commercial License
- PyQt does not include a copy of Qt (you must obtain a correctly licensed copy of Qt yourself)

PyQt

- PyQt brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python

- PyQt is able to generate Python code from Qt Designer
  - It is also possible to add new GUI controls written in Python to Qt Designer

- Python is a simple but powerful object-orientated language
  - Python's set of extension modules providing a wide variety of functions including HTTP servers, XML parsers, database access, data compression tools and, of course, graphical user interfaces

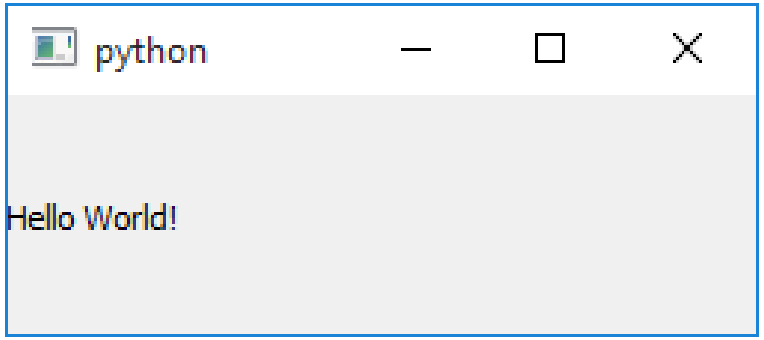
- PyQt combines all the advantages of Qt and Python
  - A programmer has all the power of Qt, but is able to exploit it with the simplicity of Python

PyQt. “Hello World!”

```
import sys
from PyQt5.QtWidgets import QApplication, QLabel

if __name__ == '__main__':

    app = QApplication(sys.argv)
    label = QLabel('Hello World!')
    label.show()
    sys.exit(app.exec_())
```



PyQt. “Hello World!”

`import sys`

- system-specific parameters and functions

`from PyQt5.QtWidgets import QApplication, Qlabel`

- the QApplication class manages the GUI application's control flow and main settings
- the QLabel widget provides a text or image display

`app = QApplication(sys.argv)`

- for any GUI application using Qt, there is precisely one QApplication object

`label = QLabel('Hello World!')`

- new instance of a Qlabel

`label.show()`

- shows the widget and its child widgets

`sys.exit(app.exec())`

- enters the main event loop and waits until exit() is called, then returns the value that was set to exit()
- It is necessary to call exec() to start event handling