Introduction to Qt

- Create powerful Graphics User Interfaces (GUI)
- Multi-platform : Windows, MacOs, Linux, Symbian, Android…
- Complete framework : 2D and 3D graphics, network, XML, SQL, …
- Widely used : Autodesk Maya, Adobe Photoshop Elements, Skype, VLC Media Player, VirtualBox, Mathematica, KDE…

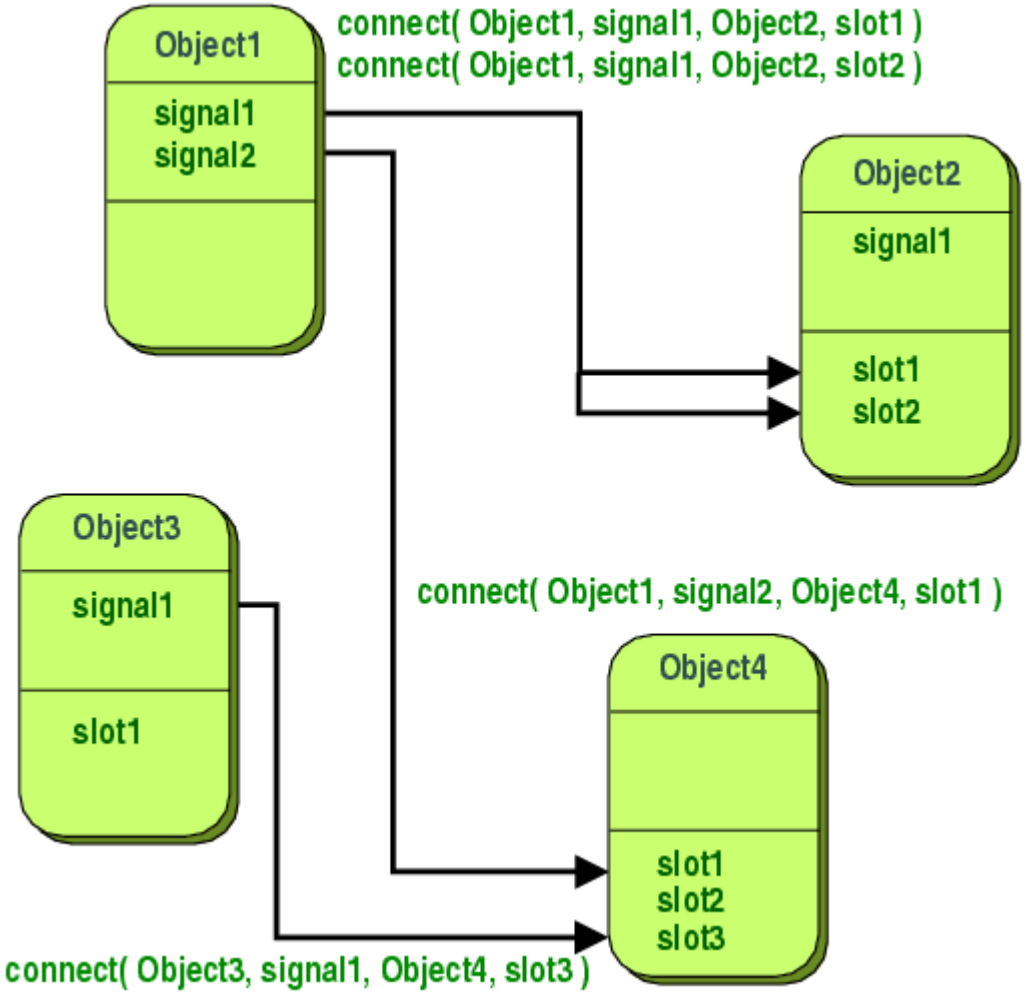
The Qt Object Model

- a very powerful mechanism for seamless object communication called signals and slots
- queryable and designable object properties
- powerful events and event filters
- contextual string translation for internationalization
- sophisticated interval driven timers that make it possible to elegantly integrate many tasks in an event-driven GUI
- hierarchical and queryable object trees that organize object ownership in a natural way
- guarded pointers (QPointer) that are automatically set to 0 when the referenced object is destroyed, unlike normal C++ pointers which become dangling pointers when their objects are destroyed
- a dynamic cast that works across library boundaries.

Signal and Slots

http://qt-project.org/doc/qt-5.1/qtcore/signalsandslots.html

- Event-driven control
- Signals and slots are functions with (matching) parameters
- An object which emits a signal neither knows nor cares which slots (possibly many) receive the signal
- a slot does not know if it has any signals (possibly many) connected to it



Simple example

- C++ counter class

```
class Counter
{
public:
    Counter() { m_value = 0; }

    int value() const { return m_value; }
    void setValue(int value);

private:
    int m_value;
};
```

- Qt counter
  - Derives from QObject
  - Code will be pre-processed by Meta-Object Compiler (MOC)

```
#include <QObject>

class Counter : public QObject
{
    Q_OBJECT

public:
    Counter() { m_value = 0; }

    int value() const { return m_value; }

public slots:
    void setValue(int value);

signals:
    void valueChanged(int newValue);

private:
    int m_value;
};
```

Simple example (continued)

- Slot implementation

```
void Counter::setValue(int value)
{
    if (value != m_value) {
        m_value = value;
        emit valueChanged(value);
    }
}
```

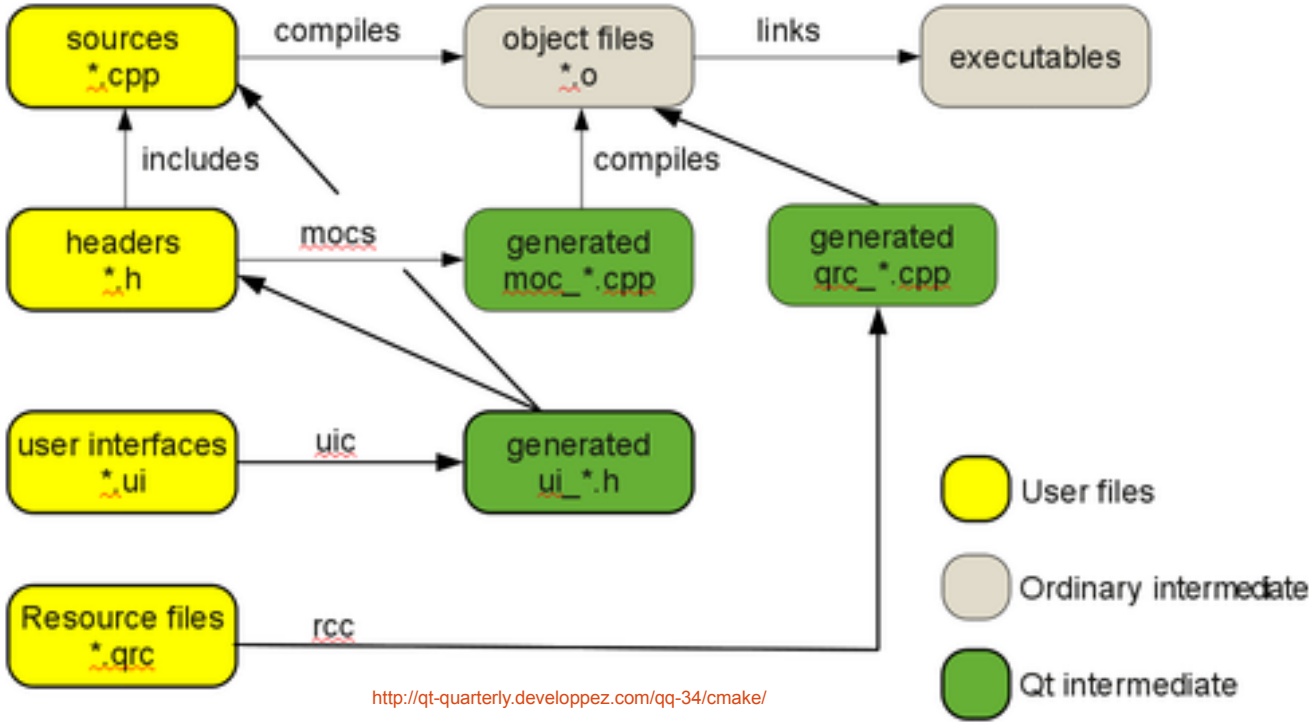
- Example of use

```
Counter a, b;
QObject::connect(&a, SIGNAL(valueChanged(int)),
                 &b, SLOT(setValue(int)));

a.setValue(12);     // a.value() == 12, b.value() == 12
b.setValue(48);     // a.value() == 12, b.value() == 48
```

Building a Qt project

- Q_OBJECTs require a MOC pass
- Widget descriptions (.ui) require the generation of .h and .cpp files
- Resources (.qrc) are translated to .cpp



http://qt-quarterly.developpez.com/qq-34/cmake/

- QMake makes this automatically
- Possible with CMake too

References

- Qt Documentation
  - http://qt-project.org/doc/
  - http://qt-project.org/doc/qt-4.8/how-to-learn-qt.html

- Other sources
  - Nice tutorial : http://fr.openclassrooms.com/informatique/cours/progra