Qt – a short intro

- Qt (/ˈkjuːt/ "cute")
- Cross platform
- C++ plus some sugar
- Several language bindings
- Sql, xml, json, thread, collections, network...
- GPL v3, LGPL v2 and commercial
- IOS, Android, BlackBerry, Linux, Windows, Mac

Qt – a short intro

- Qt (/ˈkjuːt/ "cute")
- Cross platform
- C++ plus some sugar
- Several language bindings
- Sql, xml, json, thread, collections, network...
- GPL v3, LGPL v2 and commercial
- IOS, Android, BlackBerry, Linux, Windows, Mac

History

- Started 1991 by Haavard Nord and Eirik Chambe-Eng
- 1995 1st release for Linux & Windows
  - 1998 KDE 1.0
  - 1998 Trolltech – KDE – Free Licensing Agreement
  - 2000 KDE 2.0 based on Qt2
- 2001 Qt3 added Mac OS X
  - 2002 KDE 3.0
- 2005 Qt4
  - 2008 KDE 4.0
- 2012 Qt5, current 5.1.0

Qt Applications

- See http://en.wikipedia.org/wiki/Category:Software_that_uses_Qt
  - Adobe Photoshop Album
  - Autodesk Maya
  - Doxygen
  - EAGLE
  - Google Earth
  - Guitar Pro
  - Qcad

Qt GUI

- QtQuick
  - CSS-like description
  - Javascript interaction
- Qwidget
  - Several widgets
  - Layout managers
  - GUI designer

Qt for vi – part 1: hello world

- SKD: http://qt-project.org/downloads -> Qt 5.1

```
#include <QLabel>
#include <QApplication>

int main(int argc, char* argv[])
{
  QApplication app(argc,argv);
  (new QLabel("Hello world!"))->show();
  return app.exec();
}
```

```
cd hello1
vi main.cpp
qmake -project
qmake
make
./hello1
```

Qt Widgets

- http://qt-project.org/doc/qt-5.1/qtwidgets/gallery.html
- Buttons: QPushButton, QCheckBox, QRadioButton
- Containers: QGroupBox, QTabWidget, QFrame, QToolBox
- Item Views: QListView, QTreeView, QTableView
- Input: QLineEdit, QDateEdit, QTimeEdit, QDateTimeEdit, QComboBox, QCalendarWidget

Qt for vi – part 2: create widgets

```
#include <QWidget>
class MyWidget : public QWidget
{
Q_OBJECT
public:
   MyWidget(QWidget* parent = 0);
};
```

header mywidget.h

```
#include "mywidget.h"

MyWidget::MyWidget(QWidget *parent)
  : QWidget(parent)
{
```

impl mywidget.cpp

```
#include <QApplication>
#include "mywidget.h"

int main(int argc, char* argv[])
{
  QApplication app(argc,argv);
  (new MyWidget)->show();
  return app.exec();
}
```

main.cpp

Qt for vi – part 3: add content

```
#include <QLabel>
#include <QLineEdit>

class MyWidget : public QWidget
{
  Q_OBJECT
public:
  MyWidget(QWidget* parent = 0);
private:
  QLabel *label;
  QLineEdit *edit;
};
```

header mywidget.h

```
#include "mywidget.h"

MyWidget::MyWidget(QWidget *parent)
  : QWidget(parent),
    label(new QLabel(this)), edit(new QLineEdit(this))
{
  resize(200,300);
  label->setText("Hello world!");
  label->show();
  edit->setPlaceholderText("type here");
  edit->move(0,100);
  edit->show();
}
```

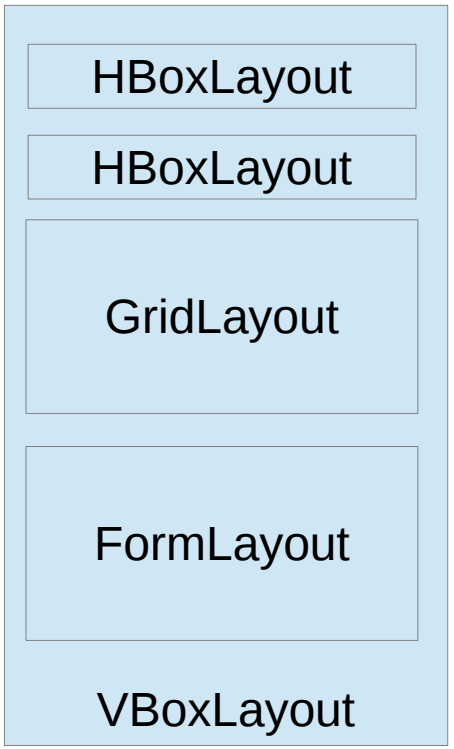
impl mywidget.cpp

- fixed positioning is bad, very bad!
- never use such code!
- for eduction only!

Layout Manager

- Positioning of child widgets
- Sensible default/minium sizes for widgets
- Resize handling
- Automatic updates when contents change:
  - Font size, text or other contents of child widgets
  - Hiding or showing a child widget
  - Add/Removal of child widgets
- Commonly used: QHBoxLayout, QVBoxLayout, QGridLayout, and QFormLayout.

Qt for vi – part 4: layouts




```
#include <QtWidgets>
#include "mywidget.h"


MyWidget::MyWidget(QWidget *parent)
  : QWidget(parent)
{
  QVBoxLayout *vbox = new QVBoxLayout;
  // first HBoxLayout
  QHBoxLayout *hbox = new QHBoxLayout;
  label = new QLabel("First:");
  hbox->addWidget(label);
  edit = new QLineEdit;
  hbox->addWidget(edit);
  vbox->addLayout(hbox);
  // second HBoxLayout
  hbox = new QHBoxLayout;
  label = new QLabel("Second:");
  hbox->addWidget(label);
  edit = new QLineEdit;
  hbox->addWidget(edit);
  vbox->addLayout(hbox);
  // now the grid
  QGridLayout *grid = new QGridLayout;
  for(int i=0;i<3;++i)
    grid->addWidget(new QLabel(QString("-- 0,%1 --").arg(i)),0,i);
  grid->addWidget(new QLabel("-- 1,0-1 --"),1,0,1,2,Qt::AlignHCenter);
  grid->addWidget(new QLabel("-- 1,2 --"),1,2);
  vbox->addLayout(grid);
  // and the form
  QFormLayout *form = new QFormLayout;
  form->addRow(new QLabel("Form first:"), new QLineEdit);
  form->addRow(new QLabel("Form second:"), new QLineEdit);
  vbox->addLayout(form);
  // finally set the layout for this widget
  setLayout(vbox);
}
```

Interaction – slots and signals

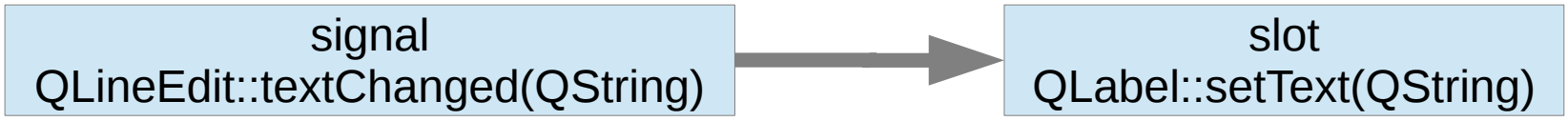
- qt specific extensions to c++
- objects may emit or receive signals

```
#include <QtWidgets>

class MyWidget : public QWidget
{
  Q_OBJECT
public:
  MyWidget(QWidget* parent = 0);
private:
  QLabel *label;
  QLineEdit *edit;
};
```

header mywidget.h

```
#include "mywidget.h"

MyWidget::MyWidget(QWidget *parent)
  : QWidget(parent),
    label(new QLabel(this)), edit(new QLineEdit(this))
{
  QVBoxLayout *l = new QVBoxLayout;
  l->addWidget(edit);
  l->addWidget(label);
  setLayout(l);
  connect(edit, SIGNAL(textChanged(QString)),
                label,SLOT(setText(QString)));
}
```

impl mywidget.cpp



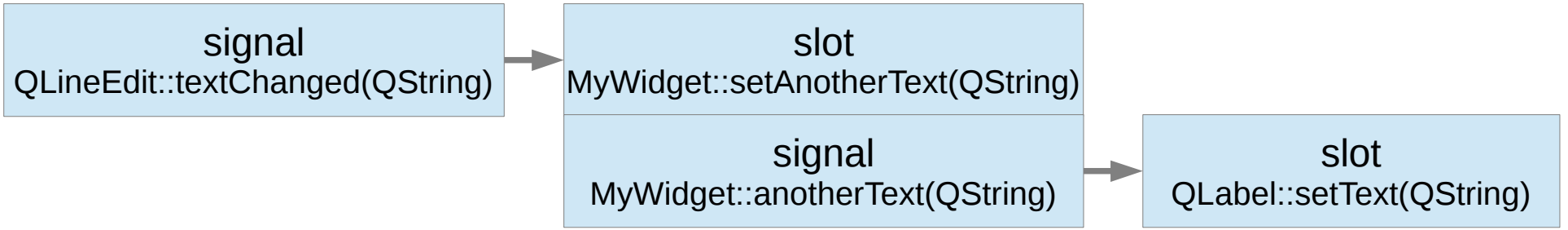
Self made slots and signals

```
#include <QtWidgets>

class MyWidget : public QWidget
{
  Q_OBJECT
public:
  MyWidget(QWidget* parent = 0);
signals:
  void anotherText(QString);
private slots:
  void setAnotherText(QString);
private:
  QLabel *label;
  QLineEdit *edit;
};
```

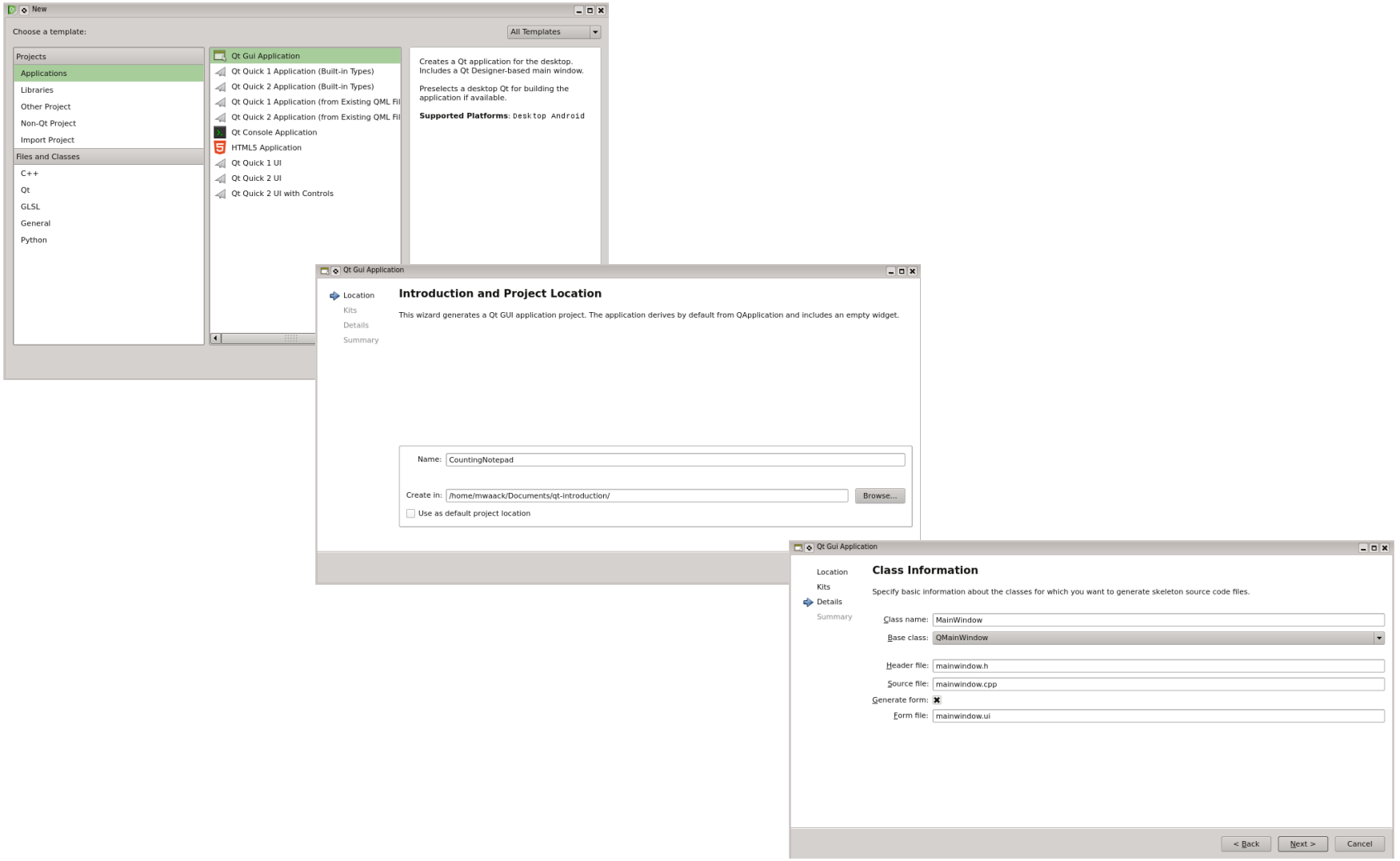
header mywidget.h

```
#include "mywidget.h"
#include <algorithm>

MyWidget::MyWidget(QWidget *parent)
  : QWidget(parent), label(...), edit(...)
{ // the layout stuff
  connect(edit, SIGNAL(textChanged(QString)),
                this,SLOT(setAnotherText(QString)));
  connect(this, SIGNAL(anotherText(QString)),
                label,SLOT(setText(QString)));
}

void MyWidget::setAnotherText(QString text)
{
  std::reverse(text.begin(),text.end());
  emit anotherText(text);
}
```

impl mywidget.cpp



QtCreator


New
Choose a template:
All Templates
Projects
Applications
Libraries
Other Project
Non-Qt Project
Import Project
Files and Classes
C++
Qt
GLSL
General
Python
Qt Gui Application
Qt Quick 1 Application (Built-in Types)
Qt Quick 2 Application (Built-in Types)
Qt Quick 1 Application (from Existing QML Fil
Qt Quick 2 Application (from Existing QML Fil
Qt Console Application
HTML5 Application
Qt Quick 1 UI
Qt Quick 2 UI
Qt Quick 2 UI with Controls
Creates a Qt application for the desktop.
Includes a Qt Designer-based main window.
Preselects a desktop Qt for building the application if available.
Supported Platforms: Desktop Android
Qt Gui Application
Location
Kits
Details
Summary
Introduction and Project Location
This wizard generates a Qt GUI application project. The application derives by default from QApplication and includes an empty widget.
Name: CountingNotepad
Create in: /home/mwaack/Documents/qt-introduction/
Browse...
Use as default project location
Qt Gui Application
Location
Kits
Details
Summary
Class Information
Specify basic information about the classes for which you want to generate skeleton source code files.
Class name: MainWindow
Base class: QMainWindow
Header file: mainwindow.h
Source file: mainwindow.cpp
Generate form:
Form file: mainwindow.ui
< Back
Next >
Cancel