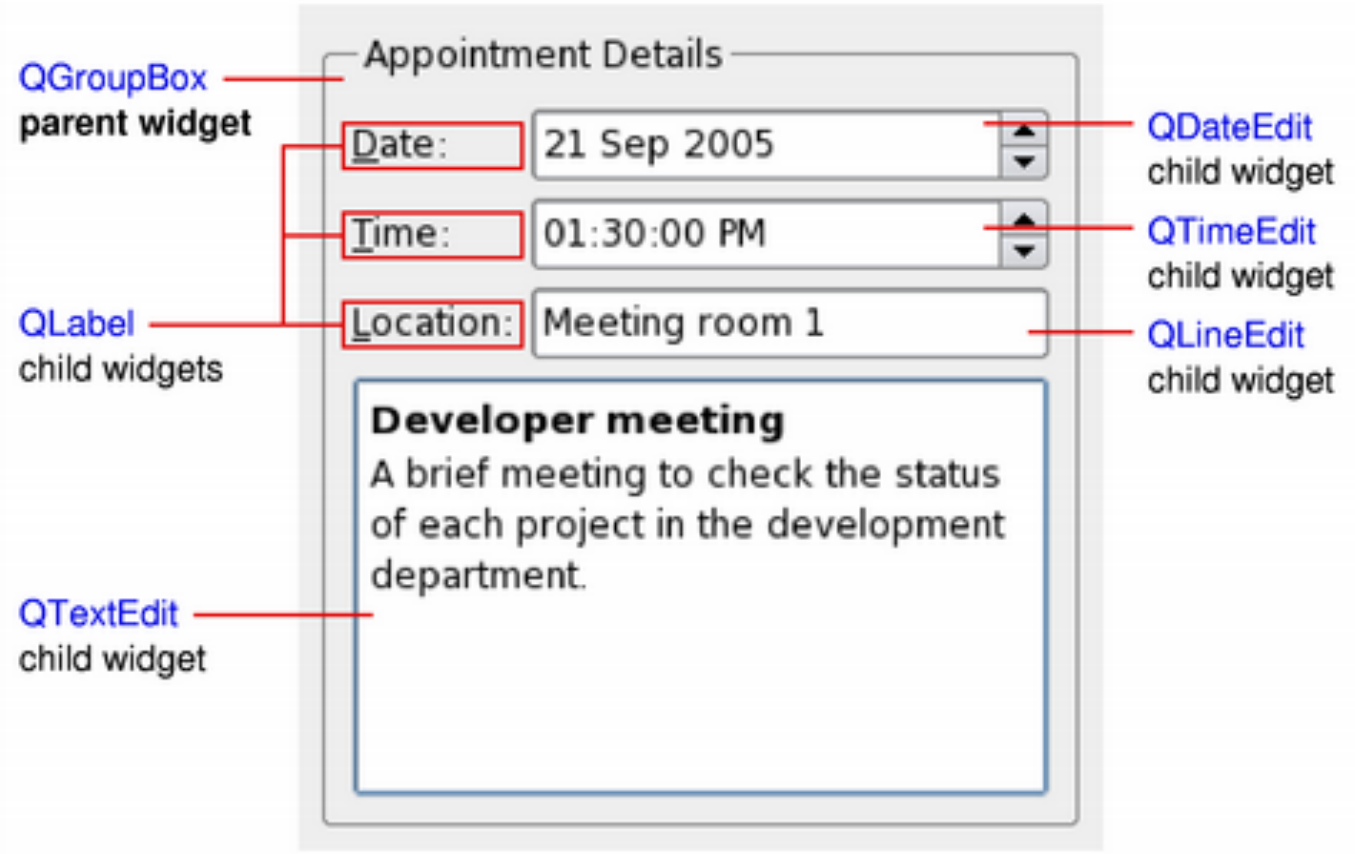
Tampereen yliopisto
Tampere University

Käyttöliittymä

16.09.2019

Käyttöliittymä ja komponenttien asemointi



Komponenttien asemointi

- Kolme vaihtoehtoa:
  - Absoluuttiset koordinaatit
  - Manuaalinen layout
  - Layout-managerit
- Qt:ssa myös QML

Absoluuttiset koordinaatit

- Ohjelmoija määrittää jokaisen komponentin koordinaatit ja koon
- Käyttäjä ei voi muuttaa ikkunan kokoa
- Fonttien muutos voi katkaista tekstin
- Kaikki tyylit eivät välttämättä toimi (komponenttien kokojen takia)
- Ohjelman ylläpitäminen työlästä

Manuaalinen layout

- Ohjelmoija määrittää komponenttien paikat, mutta niiden koot lasketaan ikkunan koon mukaan.
- Tehdään ylikirjoittamalla dialogin resizeEvent() –funktio
- Lopputulos usein parempi kuin absoluuttisilla koordinaateilla, mutta toteuttaminen työlästä

Qt:n layout-managerit

Yksinkertainen tapa käyttöliittymäkomponenttien automaattiseen järjestämiseen

- kuvaavat, miten komponentit asettuvat käyttöliittymään
- automaattisesti asemoivat komponentteja ja säätävät niiden kokoa
- takaavat, että komponenteilla on säännönmukainen järjestys ja että käyttöliittymä pysyy käytettävänä

Qt:n layout-managerit

- Kokomääritykset summittaisia
  - Minimum size
  - Maximum size
  - Preferred size
- Widgetti voi olla skaalautuva tai säilyttää oman oletuskoon (preferred size)
- Layout-manageri laskee widgetin koon ja sijainnin uudelleen aina, kun layout muuttuu tai ikkunan kokoa muutetaan – Resize event

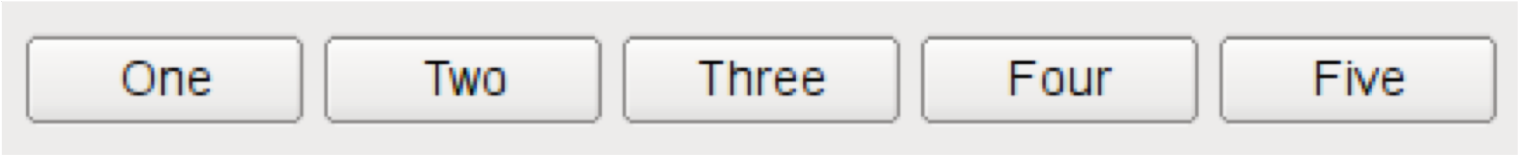
Qt:n layout-managerit

- Suoraviivaisin tie nättiin asemointiin on Qt:n valmiit layout-managerit:
  - `QHBoxLayout`
  - `QVBoxLayout`
  - `QGridLayout`, ja
  - `QFormLayout`

    joita on mahdollista latoa sisäkkäin

QHBoxLayout

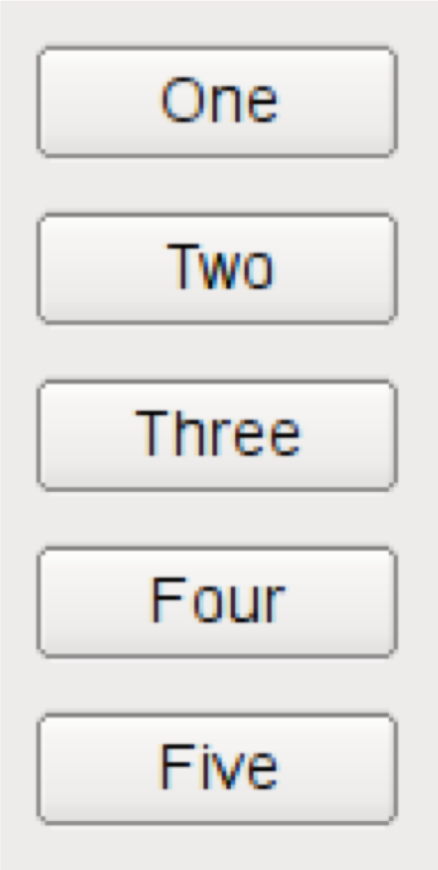
Latoo komponentit vaakasuoraan



Kuva: Qt:n dokumentaatio

QVBoxLayout

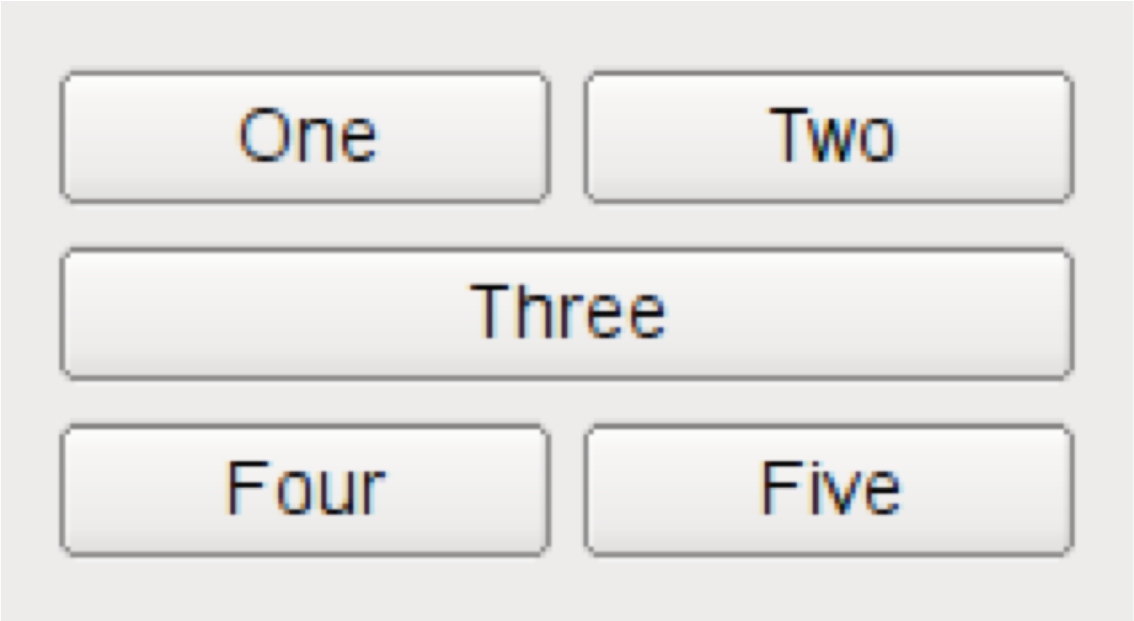
Latoo komponentit pystysuoraan



Kuva: Qt:n dokumentaatio

QGridLayout

Latoo komponentit ruudukkoon



Kuva: Qt:n dokumentaatio

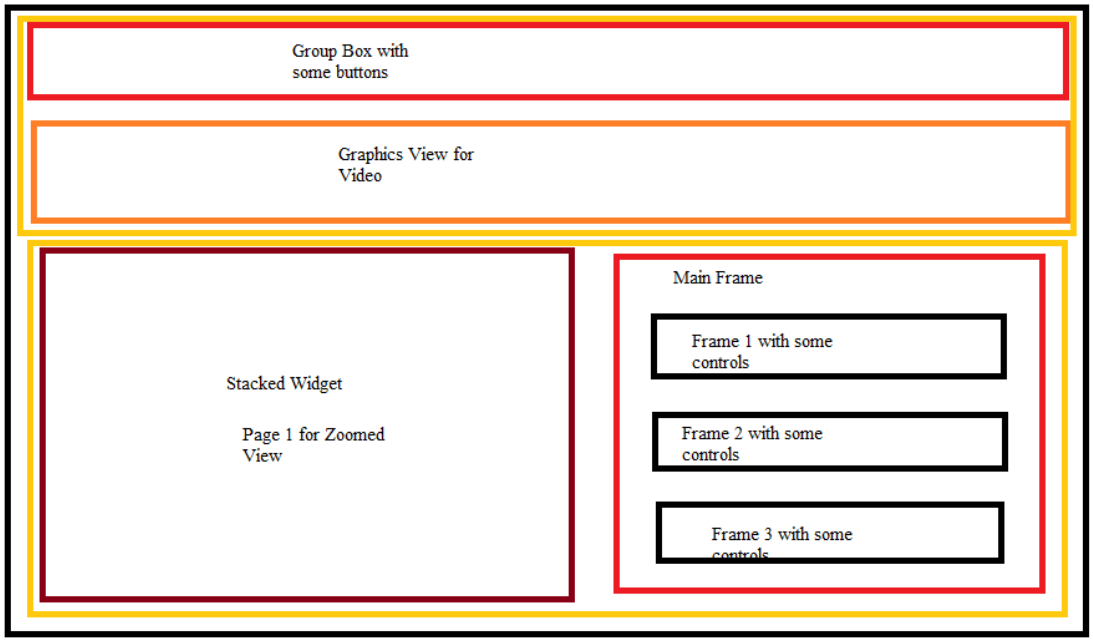
QFormLayout

Latoo komponentit kahteen sarakkeeseen

Kuva: Qt:n dokumentaatio

Sisäkkäiset layoutit

Dialogi-ikkuna koostuu tyypillisesti useista sisäkkäisistä layouteista



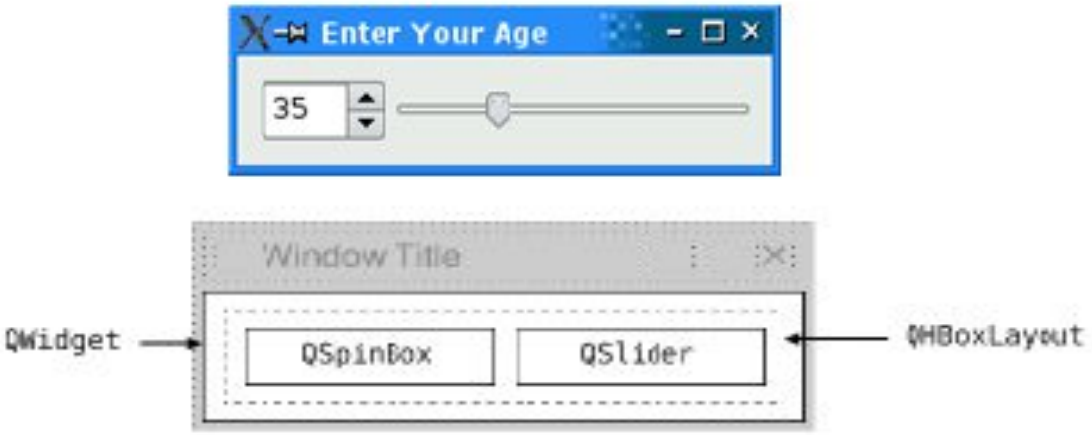
Kuva: Stackoverflow

Parent

- Layout asettaa widgeteille automaattisesti parent-widgetin (reparent)
- Layoutiin asetetun widgetin parent-widget ei ole layout itse vaan widgetti, johon layout on asetettu (setLayout)

Esimerkki

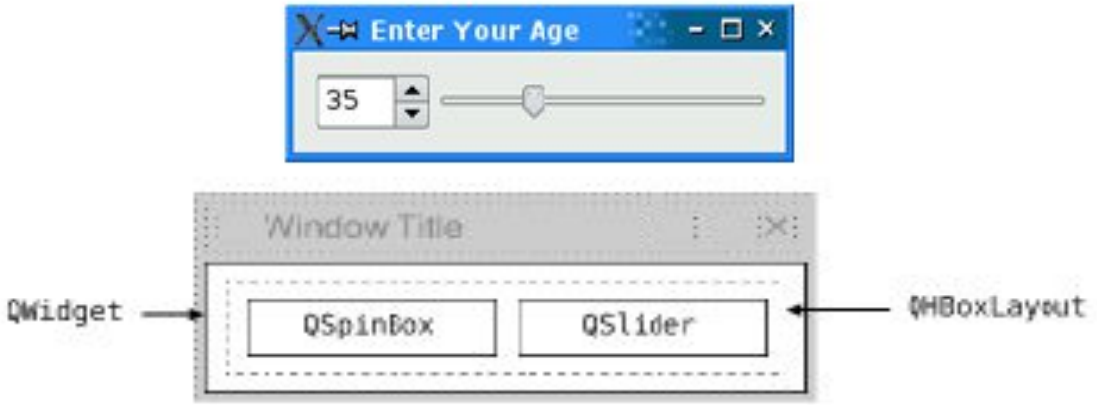
- Kolme widgettia:
  - parent: QWidget (pääikkuna)
  - children: QSpinBox ja QSlider
- Signaalit ja slotit synkronoivat widgetteja



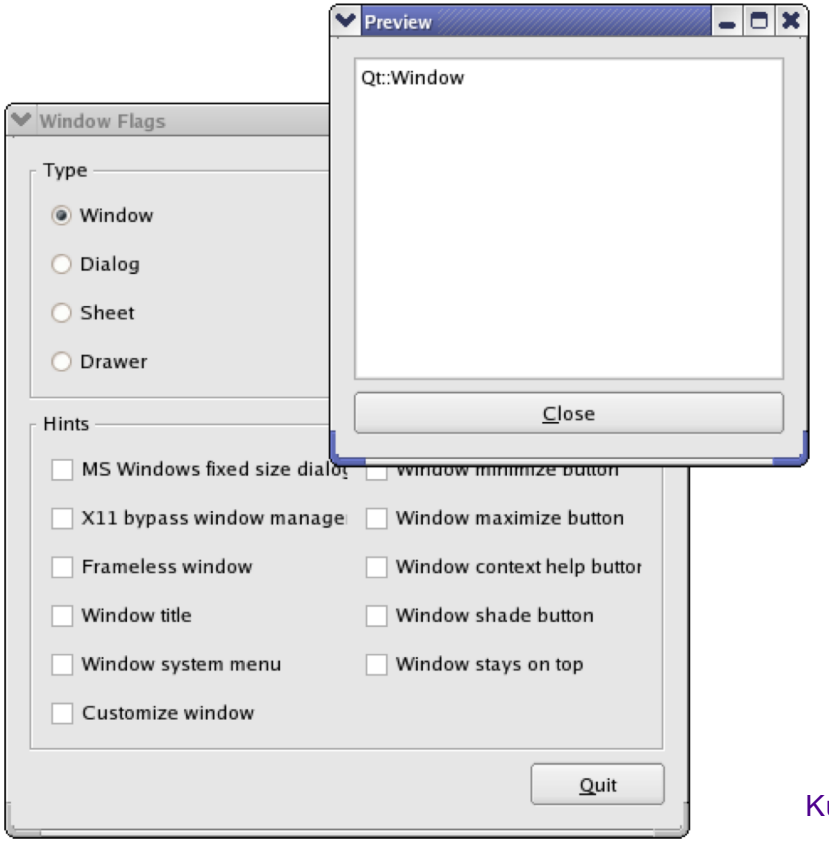
```
#include <QApplication>
#include <QHBoxLayout>
#include <QSlider>
#include <QSpinBox>

int main(int argc, char *arch[]) {
    QApplication app(archc, argv);
    QWidget *window = new QWidget;
    window->setWindowTitle("Enter your age");
    QSpinBox *spinBox = new QSpinBox;
    QSlider *slider = new QSlider(QT::Horizontal);
    spinBox->setRange(0, 130);
    slider->setRange(0, 130);          // jatkuu
```

```
    QObject::connect(spinBox, SIGNAL(valueChanged(int)),
                     slider, SLOT(setValue(int)));
    QObject::connect(slider, SIGNAL(valueChanged(int)),
                     spinBox, SLOT(setValue(int)));
    spinBox->setValue(35);
    QHBoxLayout *layout = new QHBoxLayout;
    layout->addWidget(spinBox);
    layout->addWidget(slider);
    window->setLayout(layout);
    window->show();
    return app.exec();
}
```



Usean ikkunan käytöstä



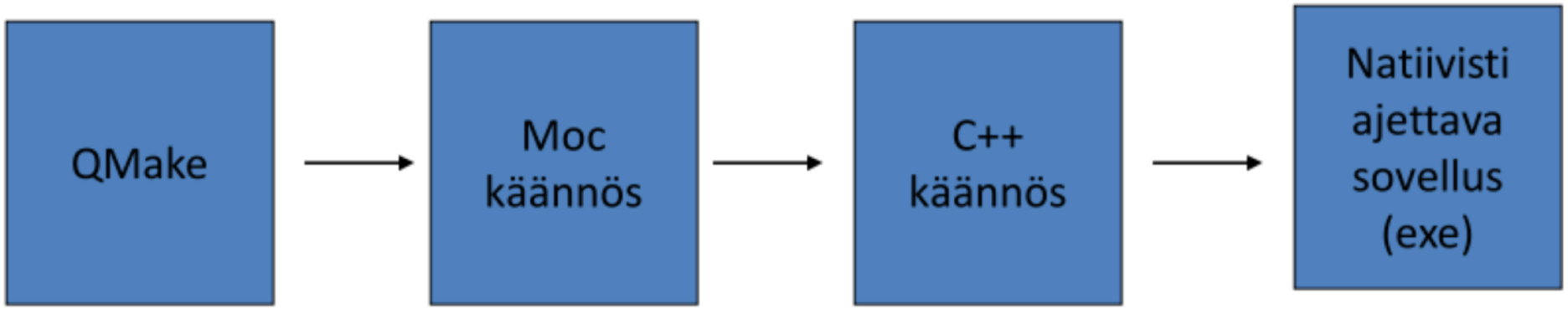
Kuva: Qt:n dokumentaatio

Tampereen yliopisto
Tampere University

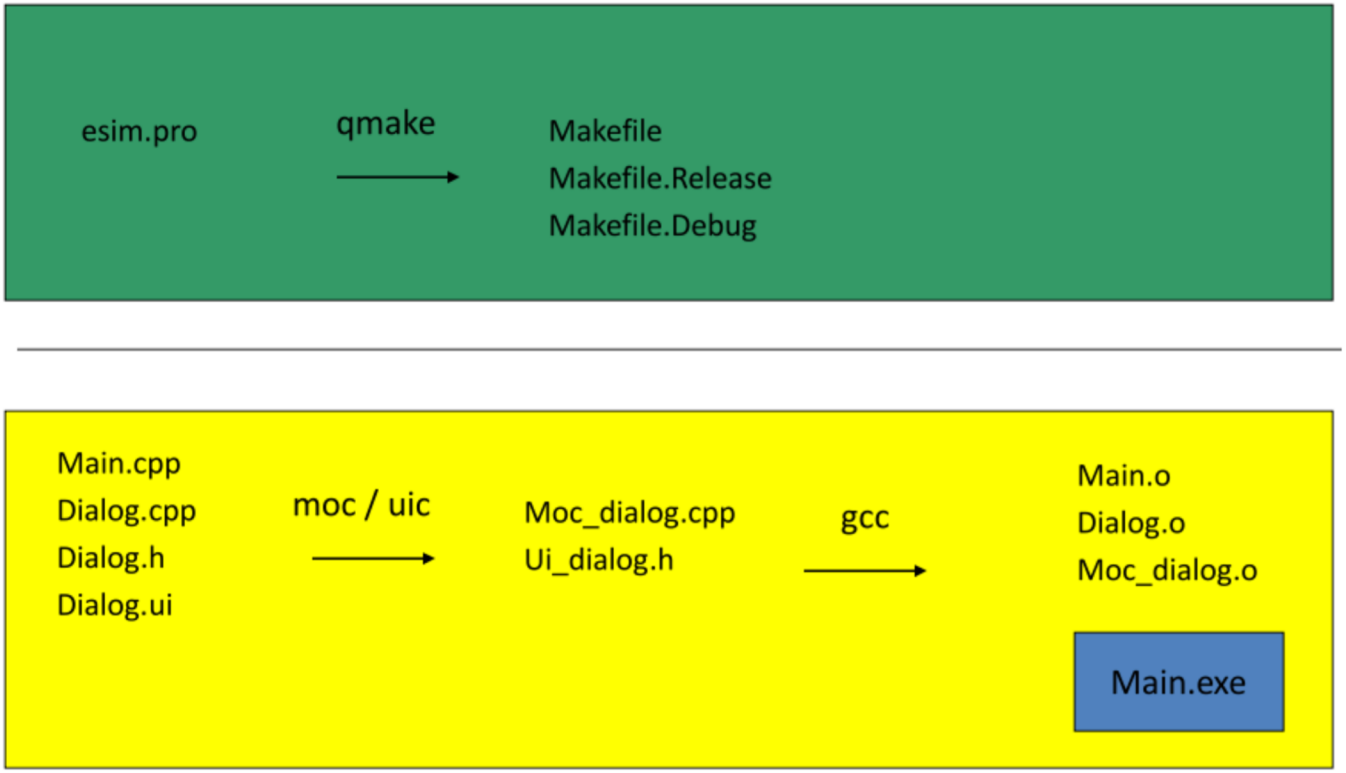
Qt konepellin alla

Code less. Create more.
Deploy everywhere.

Qt: käännösprosessi



Qt: käännösprosessi



Qt: käännösprosesi

- Käännökseen tarvittavat tiedostot luodaan qmake-ohjelmalla projektitiedoston (.pro) tietojen perusteella
- Käyttöliittymätiedostot (.ui) käännetään uic-kääntäjällä (User Interface Compiler) ja metatiedot moc-kääntäjällä (Meta Object Compiler)
- Metakääntäjä generoi väliaikaista C++-kääntäjälle kelpaavaa koodia

UI-kääntäjä (uic)

User Interface (käyttöliittymä)

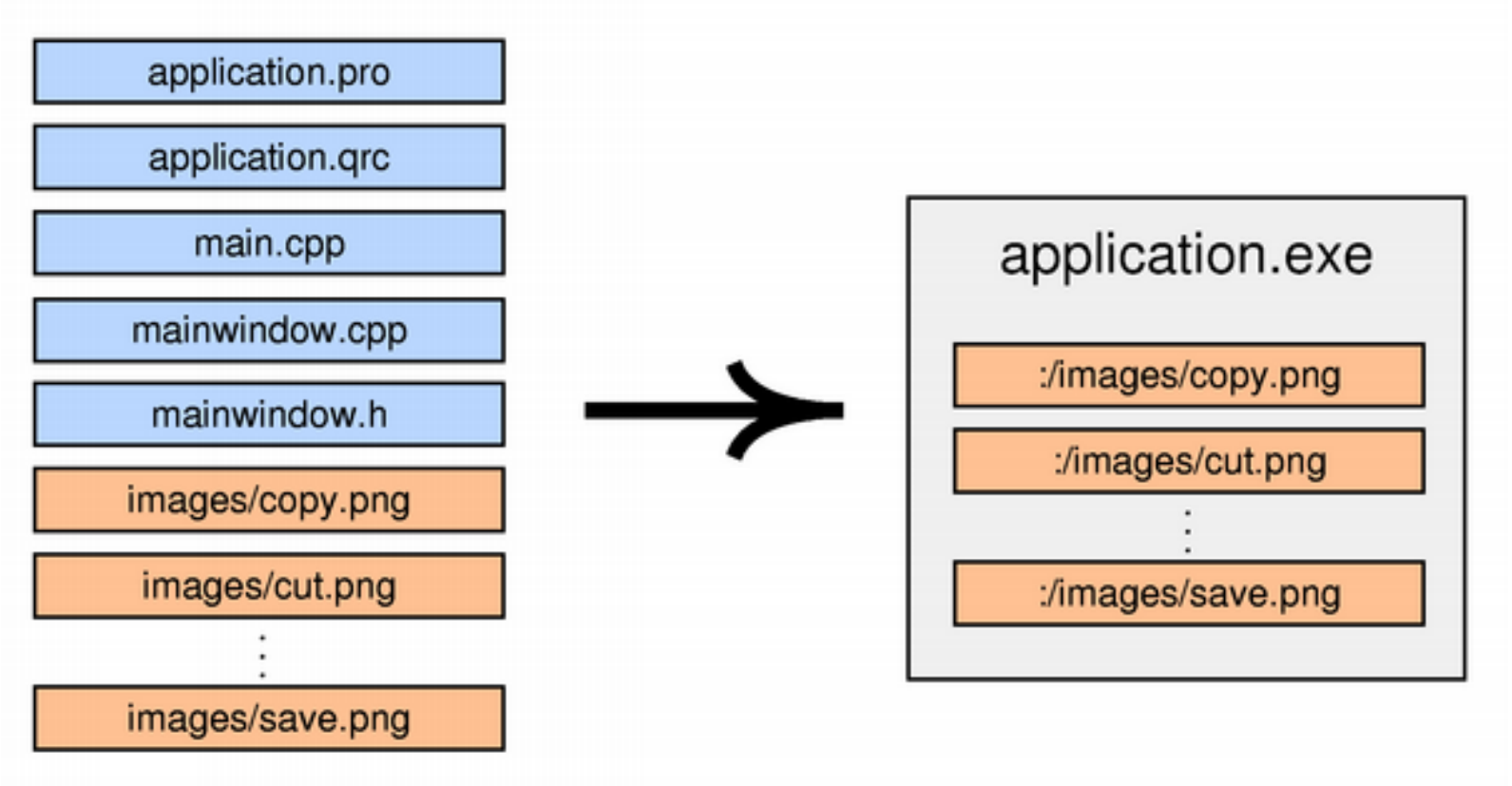
- Qt Designer generoi XML-tiedoston
- uic lukee ja kääntää tiedoston vastaavaksi C++-otsikkotiedostoksi

Metaobjektikääntäjä (moc)

Meta Object System

- `QObject`-luokka kertoo, että olio hyödyntää meta-objektisysteemiä
- `Q_OBJECT`-makro kääntää metaobjekti-ominaisuudet päälle (esim. signaalit ja slotit)
- Meta-Object-kääntäjä (moc) lisää `QObject`-aliluokkaan ominaisuuksien toteuttamiseen vaadittavan koodin

Resources


application.pro
application.qrc
main.cpp
mainwindow.cpp
mainwindow.h
images/copy.png
images/cut.png
images/save.png
application.exe
:/images/copy.png
:/images/cut.png
:/images/save.png