Laboratori:
Custom Widgets

Professors de INDI 2020/21 - Q2

Taula de continguts

Widgets propis

- Per crear la interfície d'una aplicació complexa sovint els widgets dels que disposa Qt no són suficients (no tenen les interaccions que volem).
- Caldrà crear les nostres pròpies classes derivades de les de Qt per a programar els slots que calguin. Podem derivar de:
  - QtObject (per a objectes no gràfics)
  - QtWidget o les seves derivades (per a dissenyar nous components gràfics amb noves funcionalitats)

Herència

Crear una classe derivada d'una classe base implica que la classe derivada hereta tots els **mètodes i atributs públics i protegits** de la classe base.

```
class MyLabel: public QLabel
{
        ...
}
```

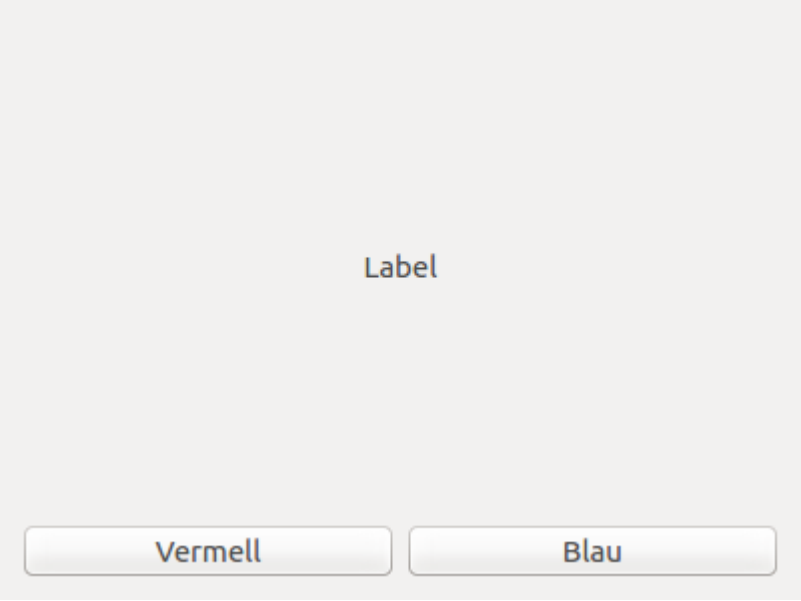
Taula de continguts

Creació d'una nova interfície

- Passos a seguir per crear una interfície en Qt:
  1. Dissenyar la interfície amb el ***designer***.
  2. Dissenyar les interaccions (connexions dels signals cap els slots).
  3. En cas que algun widget necessiti slots/signals no disponibles crear una nova classe que hereti d'aquest widget amb els nous slots/signals.
  4. En el ***designer*** Promoure el widget cap a la nova classe que has creat.

Exemple 1: Widget propi

- Feu una aplicació que tingui dos botons que permetin canviar el color de fons (a vermell o blau) d'una etiqueta.



Exemple 1: Nova classe QLabel

- Una QLabel no té cap slot que permeti canviar el color de fons.
- Per aquest motiu, cal fer una classe que enriqueixi QLabel amb nous slots (i en cas necessari signals).
- **Alerta!!** Podem definir nous slots i signals en una nova classe però només s'implementen els slots.
- Sempre farem una nova classe heretant del widget que volem enriquir.

Exemple 1: MyLabel.h

```
#include <QLabel>
class MyLabel: public QLabel
{
    Q_OBJECT    <----- IMPORTANT
public:
    MyLabel(QWidget *parent);
    ~MyLabel();
public slots:    <------- IMPORTANT
    void changeToRed();
    void changeToBlue();
};
```

Els slots els implementarem a MyLabel.cpp

Exemple 1: MyLabel.cpp

```
#include "MyLabel.h"


MyLabel::MyLabel(QWidget *parent) : QLabel(parent) { }


MyLabel::~MyLabel() { }


void MyLabel::changeToRed() {
  // el mètode setStyleSheet l'heretem de QWidget i
  // permet canviar l'estil del widget
  setStyleSheet("background-color: red;");
}


void MyLabel::changeToBlue() {
  setStyleSheet("background-color: blue;");
}
```

Exemple 1: Compilar MyLabel

- No és codi C++.
- Necessita ésser preprocessat amb el meta-object compiler (MOC).

  Però es fa automàticament el Makefile si treballeu amb .PRO
- Cal afegir al .PRO la nova classe:

```
HEADERS += MyLabel.h

SOURCES += MyLabel.cpp
```

Promoure un widget

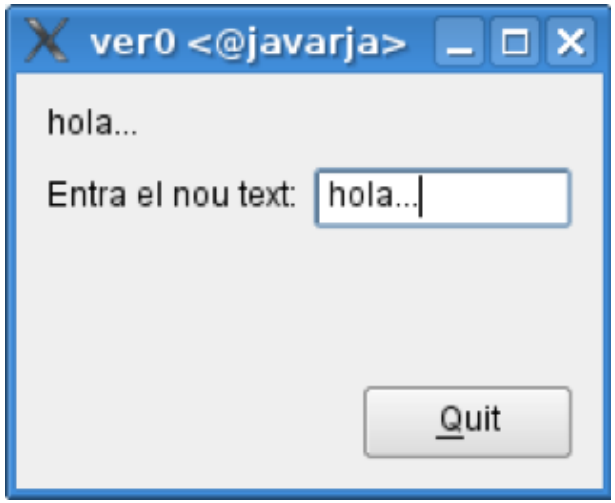
- En el ***designer*** cal dir que el widget indicat és de la classe nova que hem implementat.
- Cal cliclar el widget que volem canviar i fer `Promote to...` des del menú contextual.
- El fet de promoure un widget no afegeix automàticament els signals/slots de la nova classe. Cal fer-ho manualment.
- **IMPORTANT!!** Els signals/slots que afegim en el designer han de tenir el mateix nom que hem definit a la classe.

Exemple 1: Connexions signals/slots

Objecte origen	Signal	Objecte destí	Slot
QPushButton 1	clicked()	QLabel	changetoRed()
QPushButton 2	clicked()	QLabel	changeToBlue()

Exemple 2: Widget propi amb signal

- Feu una aplicació que tingui dues etiquetes i un LineEdit. Aquesta aplicació ha de copiar el text que hi ha en el LineEdit cada cop que es prem <return>.



Exemple 2: Signals i slots disponibles

Signals QLineEdit:

- returnPressed ()
- textChanged (QString)

Slots QLabel:

- setText (QString)

NO ES POT FER!

returnPressed() no passa el QString

Exemple 2: MyLineEdit.h

```
#include <QLineEdit>
class MyLineEdit: public QLineEdit
{
    Q_OBJECT     <-----  IMPORTANT
public:
    MyLineEdit(QWidget *parent);
    ~MyLineEdit();
public slots:  <------  IMPORTANT
    void tractaReturn();
signals:       <------  IMPORTANT
    void enviaText(const QString &);
};
```

Els slots els implementarem a MyLineEdit.cpp

Els signals no els implementem però es poden llençar en qualsevol punt del codi cridant a la funció:
emit nom_signal(paràmetres)

Exemple 2: MyLineEdit.cpp

```
#include "MyLineEdit.h"


MyLineEdit::MyLineEdit(QWidget *parent) : QLineEdit(parent) {
  // El constructor sempre ha de cridar el constructor de la
  // classe base.
}


MyLineEdit::~MyLineEdit() { }


void MyLineEdit::tractaReturn() {
  // Aquest slot només produeix un nou signal que és qui
  // envia el text.
  emit enviaText(text());
}
```

Exemple 2: Connexions signals/slots

Objecte origen	Signal	Objecte destí	Slot
QLineEdit	returnPressed()	QLineEdit	tractaReturn()
QLineEdit	enviaText(QString)	QLabel	setText(QString)

Taula de continguts

Exercicis

- **Consell**: Feu un directori per cada exercici.

1) Genereu l'executable amb la interfície del problema plantejat a l'exemple 1.

2) Genereu l'executable amb la interfície del problema plantejat a l'exemple 2.

3) Feu els exercicis de la col·lecció que està penjada en el campus digital.