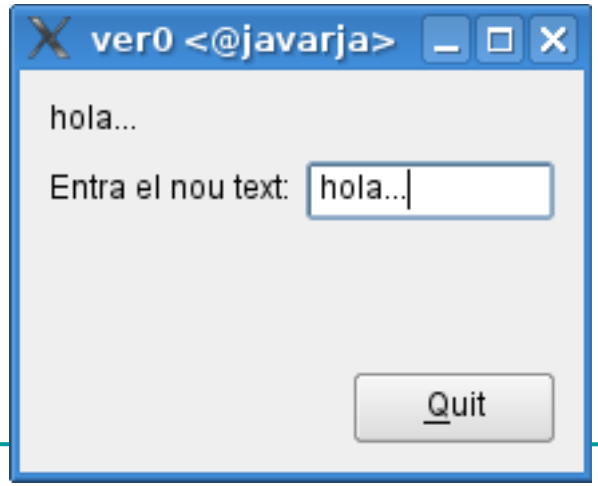
Laboratori:
Custom Widgets a Qt

Professors de IDI Q1 - 16/17

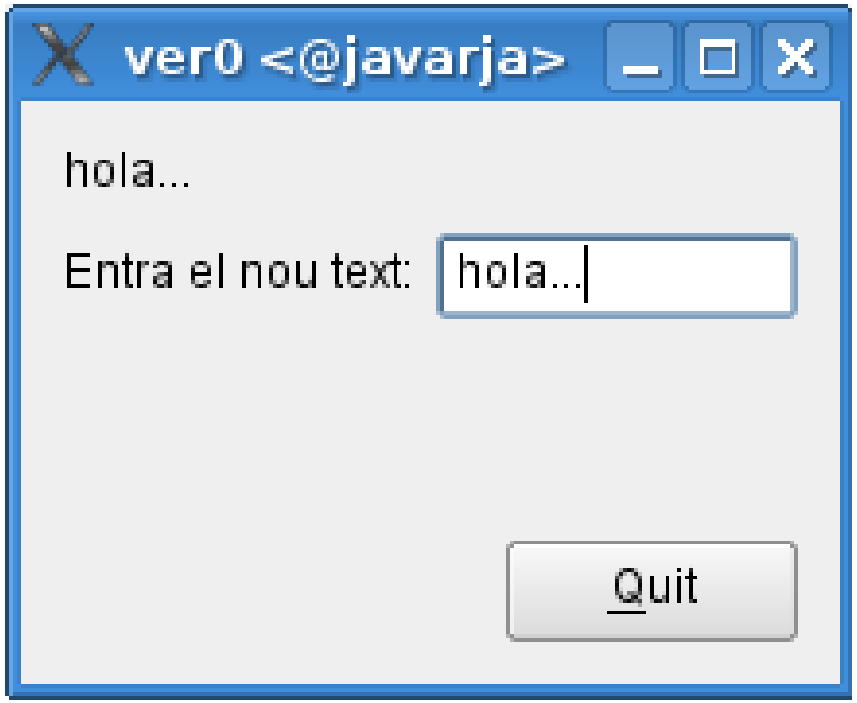
Llibreria Qt: Recordatori

- Widgets existents i configurables
- Connexions entre components mitjançant *signals* i *slots*
  - **Signal:** Esdeveniment que succeeix durant l'execució.
    - Ex: Clic sobre un widget...
  - **Slot:** mètodes especials que es poden connectar amb signals.



Llibreria Qt: No tot es pot fer directe

- Si volem que només copiï el text a l'etiqueta quan es fa <*return*>…



Signals QLineEdit:

- returnPressed ()
- textChanged (Qstring)

Slots QLabel:

- setText (Qstring)

NO ES POT FER!

Llibreria Qt: Classes pròpies

- En algunes ocasions ens caldrà crear les nostres pròpies classes derivades de les de Qt per a programar els slots i afegir els signals que calguin. Podem derivar de:
- QObject (per a objectes no gràfics)
- QWidget o qualsevol de les seves derivades (per a dissenyar nous components gràfics amb noves funcionalitats)

Exemple: MyLineEdit.h

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

Els slots els implementarem a MyLabel.cpp

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

Exemple: MyLineEdit.cpp

```
#include "MyLinEdit.h"
// constructor
MyLineEdit::MyLineEdit(QWidget *parent)
    : QLineEdit(parent) {
    // Inicialització d'atributs si cal
}
// implementació slots
void MyLineEdit::tractaReturn() {
    // Implementació de tractaReturn
    emit enviaText (text());
}
```

El constructor ha de cridar al constructor de la classe base

La implementació del slot és únicament produir el nou signal enviant el text.

Llibreria Qt: Classes pròpies

Per a compilar la classe MyLineEdit

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

Ho fa automàticament el Makefile si ho afegim al .pro

- Afegir MyLineEdit.h al HEADERS del .pro
- Afegir MyLineEdit.cpp al SOURCES del .pro

Per a usar un objecte d'aquesta nova classe al designer:

- promote...

Llibreria Qt: La classe MyGLWidget

- Com podeu veure, la nostra classe d'OpenGL MyGLWidget, en realitat és una classe pròpia derivada de QOpenGLWidget de Qt…
  - Podeu veure que el .h inclou la macro Q_OBJECT
  - I que tenim el fitxer .h en el tag HEADERS del .pro
- Per tant podem usar-la per a afegir comportament si volem que es pugui lligar amb altres components de Qt (és a dir, podem afegi-li signals i slots)

Exercicis 1 (fàcils):

1. Afegir un Radio Button a la nostra interfície que permeti decidir entre les dues òptiques: perspectiva / axonomètrica
2. Afegir un Radio Button a la nostra interfície que permeti canviar el model entre el Patricio I el legoman.
3. Afegir a la nostra interfície un Slider i un Spinbox sincronitzats entre ells que permetin controlar l'angle d'obertura de la càmera (FOV) per a fer el Zoom
4. Afegir a la nostra interficie un Spinbox que permeti modificar el factor d'escala del model

Exercicis 2:

5. Afegir a la nostra interfície dos Dials per a controlar i modificar els angles d'Euler Ψ i θ

   → Afegiu també la possibilitat que si els angles es modifiquen de manera directa amb el ratolí es vegin afectats també els valors dels Dials. Com ho podeu fer?

6. Implementeu una classe derivada de QLabel (MyLabel) com a classe pròpia que permeti mostrar mitjançant el color del seu *background* el color representat per 3 Spinbox que defineixen els valors de R, G i B del color mostrat.

   → Afegir aquesta MyLabel i els 3 Spinbox a la nostra interfície per a permetre decidir amb ells el color del terra