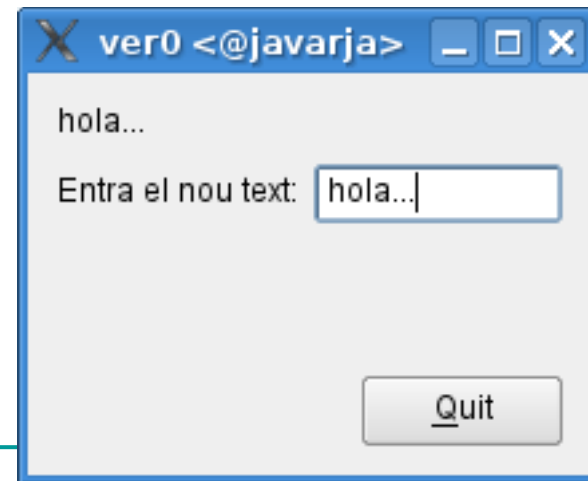

Laboratori:

Custom Widgets a Qt

Professors de IDI Q1 - 17/18

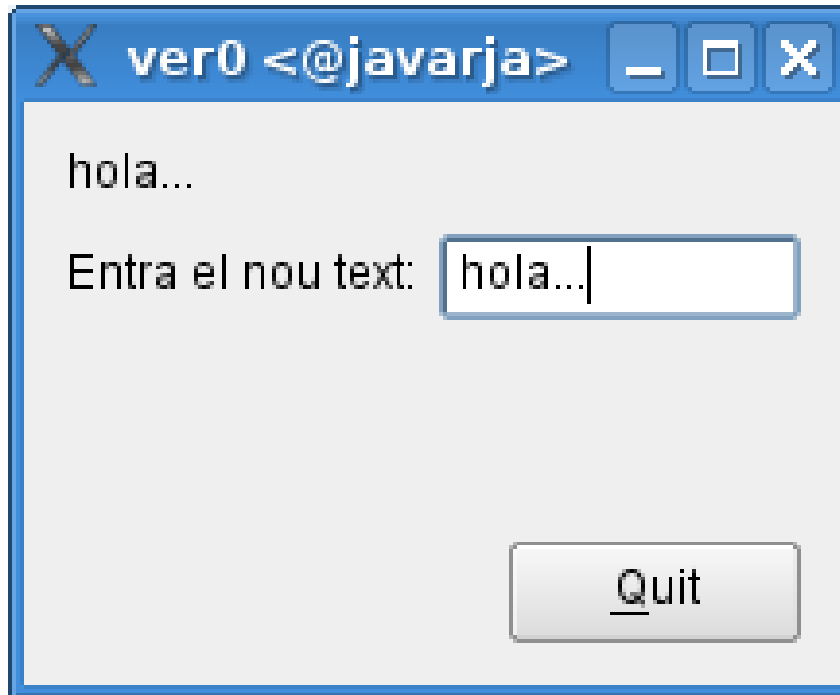
Llibreria Qt: Recordatori

- 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
`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: MyLineEdit.cpp

```
#include "MyLineEdit.h"
```

```
// constructor
```

```
MyLineEdit::MyLineEdit(QWidget *parent)  
    : QLineEdit(parent) {  
    // Inicialització d'atributs si cal  
}
```

El constructor ha de cridar al constructor de la classe base

```
// implementació slots
```

```
void MyLineEdit::tractaReturn() {  
    // Implementació de tractaReturn  
    emit enviaText (text());  
}
```

La implementació del slot només ha de 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:

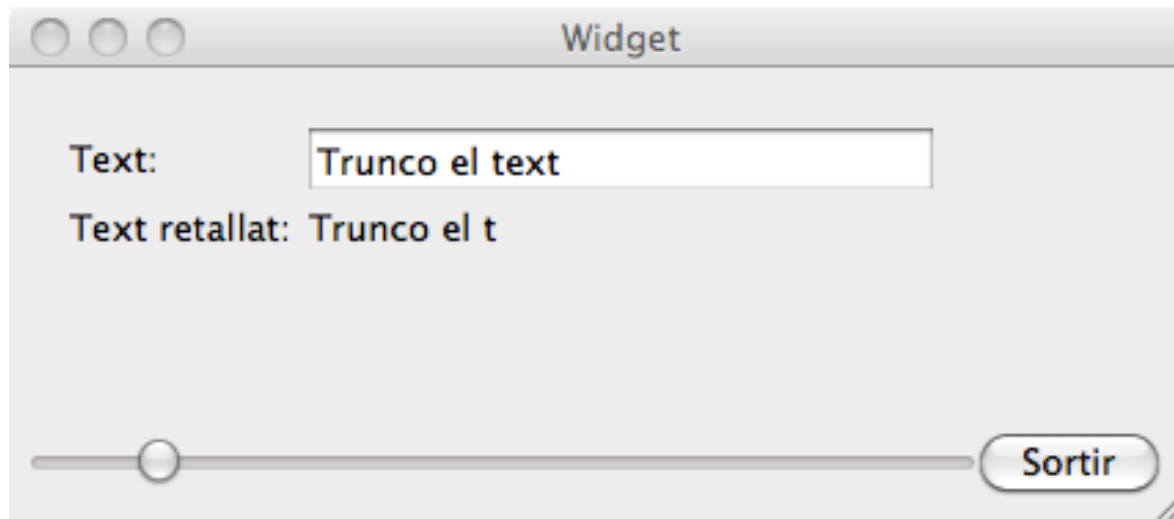
- promote...
-

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 afegir-li signals i slots)
- Recordeu afegir el “makeCurrent ()” al principi de qualsevol slot que hagi d'usar codi OpenGL.

Exercici 1:

Dissenya una aplicació amb una interfície com la que surt a la imatge. En aquesta aplicació tenim un element que ens permet entrar text (*QLineEdit*) i una etiqueta (*QLabel*) que ens el mostra. A més tenim un *QSlider* que cada vegada que es mou fa que la cadena de caràcters es trunqui amb el nombre de caràcters que indica l'*Slider*. Per a truncar una cadena de caràcters podeu usar el mètode *truncate* de *QString*. Per a solucionar-ho heu de derivar de la classe *QLabel*.

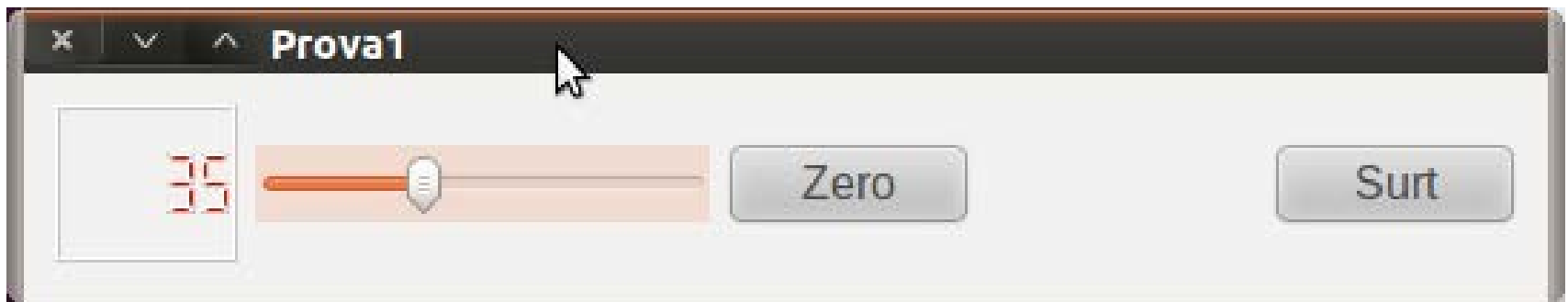


Exercici 2:

Dissenya una aplicació amb una interfície com la que surt a la imatge. En aquesta aplicació tenim un *QSlider*, un *QLCDNumber* i dos botons (veure la figura). El *QLCDNumber* ha de canviar el seu valor cada cop que es modifica el valor del *QSlider*, de manera que si està a zero, els dígitos han d'aparèixer de color verd, si està a un nombre senar els dígitos han de ser de color vermell i si el nombre és parell els dígitos han de ser de color blau.

Quan es premi “Zero”, cal que es posi el valor 0. No cal preocupar-se del valor de l'*Slider* en aquest cas.

Per a solucionar-ho heu de derivar de la classe *QLCDNumber*.



Mes exercicis:

- Comenceu els exercicis de la sessió 2.4
 - Feu-los en l'ordre en que estan, que es troben en ordre creixent de dificultat.