IDI

Connexió complexa de widgets

Prof Robert Joan Arinyo

---

1 Connexió de widgets amb parametres incompatibles

La connexió *signal-slot* entre dos widgets o entre un widget i ell mateix, només es pot establir de manera directa si el tipus del paràmetre generat pel *signal* coincideix amb el tipus del paràmetre que espera rebre l'*slot*. Cas de no ser així, cal construir una classe que faci de pont de connexió la qual calcularà el paràmetre que espera l'*slot* a partir del paràmetre generat pel *signal*. Resolguem el següent cas.

> *Dissenya una interfase que contingui un* `QLCDNumber`*, un slider horizontal, un botó, que anomenarem "Reset", i un altre botó per acabar l'execució. La Figura 1 mostra un exemple.*
>
> *La funcionalitat serà la següent. El* `QLCDNumber` *replicarà el valor de l'slider de manera que quan el valor sigui zero, el color dels digits sera verd. Quan el valor a mostrar sigui parell, el color serà blau i serà roig si el valor es senar. Cada cop que hom faci click sobre el botó de "Reset", l'slider se situarà en el valor zero, i el* `QLCDNumber` *prendrà valor zero amb color verd.*

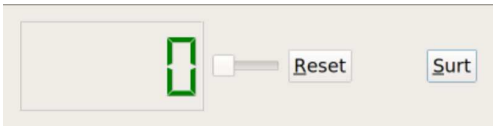


Figure 1: Exemple de disposicó en planta de la interfase.

Per a poser resoldre el problema cal analitzar els *signals* i els *slots* que cada widget ofereix. Com que els botons només generen *signals* del tipus sí/no, cal un connector que transformi aquest valor booleà en un valor per l'slider. Pel la seva banda, l'slider genera valors enters dins d'un rang. Aquesta informació permet fixar el valor que cal que el `QLCDNumber` mostri però no permet fixar el color dels dígits mostrat. Aleshores cal un connector que permeti definir el color a partir del valor. La Figura 2 mostra el conjunt de relacions *signal-slot* que calen.

Descarrega de la web del professor el codi de l'exemple d'un *slider* sense derivar classes. Estudia'l i tot seguit processa'l. Nota que les connexions entre widgets són objectes `QWidget` mentre que les connexions definides sobre connectors programats són objectes `QObject`.

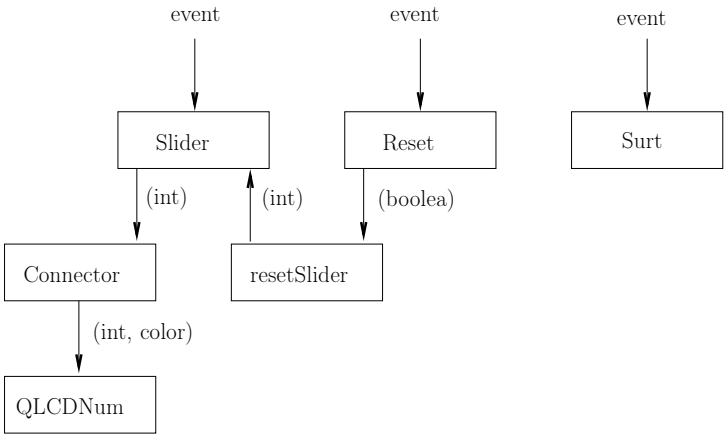


Figure 2: Connexions *signal-slot* que calen.

2 Connexió per derivació de classe

Una altra manera de resoldre la possible incompatibilitat *signal-slot* entre widgets és crear nous widgets per derivació dels existents. La idea central és afegir als widgets existent nous *signals* i *slots* que efectuin les noves funcions exigides per l'aplicació.

Un exemple de la classe `MyLCDNumber` derivada de `LCDNumber` tal que afegeixi funcions per a canviar el color dels digits i per a fer un reset gemeral del widget podria ser definida com

```
#include <QObject>
#include <QString>
#include <QLCDNumber>

class MyLCDNumber : public QLCDNumber
{
  Q_OBJECT

  public:
  MyLCDNumber(QWidget *parent, int amplada);
    //      MyLCDNumber();

  public slots:
    void setLCDColor(int valor);
    void resetLCD();

  private:

  public:
```

```
    QString color;
};
```

I la implementació sería

```
#include <QString>
#include "MyLCDNumber.h"

MyLCDNumber::MyLCDNumber(QWidget *parent, int amplada)
            :QLCDNumber(parent)
{
  setMinimumWidth(amplada);
  color = "color:green";
  setStyleSheet(color);
}

void MyLCDNumber::setLCDColor(int valor)
{
  if (valor == 0)
  {
    color = "color:green";
  }
  else if (valor % 2 != 0)
  {
    color = "color:red";
  }
  else
  {
    color = "color:blue";
  }
  setStyleSheet(color);
}

void MyLCDNumber::resetLCD()
{
  setStyleSheet("color:green");
  display(0);
}
```

La relació *signal-slot* ara és la donada a la Figura 3. La simplificació resultant és evident. Programa l'exemple de l'*slider* ara amb derivació de classes.

Per tal d'il.lustrar la inclusió de diverses operacions noves en la derivació, en l'exemple han estat definits dos mètodes: un per fixar el color i un altre per posar a zero el widget. Tenint en compte

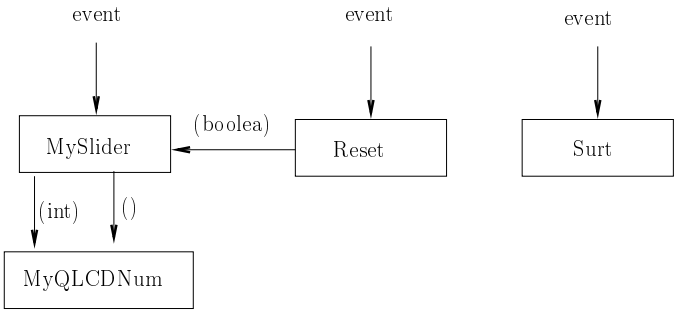


Figure 3: Connexions *signal-slot* per widgets derivats.

la relació entre el valor a mostrar i el color, modifica la classe derivada de manera que només calgui un mètode.

3 Widgets pròpis que emeten *signals*

De vegades, per a resoldre un problema, cal programar widgets tals que emeten *signals*. La tècnica implica dos conceptes que afecten al nou widget. El primer és que el nou widget es connectarà a la resta mitjançant *signals* i *slots* com ha estat descrit fins ara. El segon concepte és que el nou widget cal que generi *signals* mitjançant el mètode *emit*. La sintàxi del mètode és

```
emit nomSignal(tipus_parametre)
```

On `nomSignal` és el nom del mètode del widget que emet el *signal* i `tipus_parametre` defineix el tipus de la informació emessa. Un exemple de interfase a dissenyar és el següent.

> *Una interfase per controlar la variació d'una temperatura mostra el seu valor actual mitjançant un `QLCDNumber`. Cada cop que la temperatura depassa un valor màxim o mínim prefixats, cal mostrar un missatge de text que indiqui l'esdeveniment. Usa, per exemple un `QPlainTextEdit` que només permeti sortida. Per simular la variació de la temperatura al llarg del temps, s'utilitzarà un `QSlider`. La Figura 4 mostra un exemple.*

Descarrega de la web del professor el codi de l'exemple complet de widget que emet *signals* de la Sessió 2. Estudia'l i tot seguit processa'l.

4 Inclussió en el Designer de widgets externs

Els widgets que de vegades cal desenvolupar per a poder satisfer els requeriments que poden presentar les interfases podem agrupar-los segons dues categories: els que poden ser implemen-

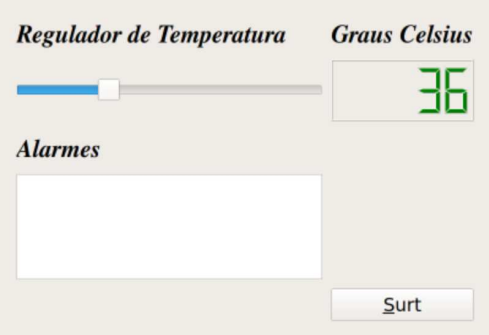


Figure 4: Exemple de disposició en planta de la interfase amb emissió de *signal*.

tats com una classe derivada d'algun widget bàsic i els que no. Els de la primera categoria són fàcilment integrables en una interfase dissenyada mitjançant el Designer. Els de la segona requereixen una mica més de feina. En aquest curs només considerarem els de la primera categoria. Aquells que estiguin interessats en els altres, llegiu les explicacions de les notes de curs *Laboratori IDI - Qt*, que podeu descarregar de la seu web del professor. Podeu trobar una extensa documentació i exemples a l'*assistant* de `Qt`.

La utilització d'un widget desenvolupat pel programador de la interfase té dues fases: 1) programació del widget i 2) inclusió en el disseny en curs del Designer.

4.1 Programació del widget

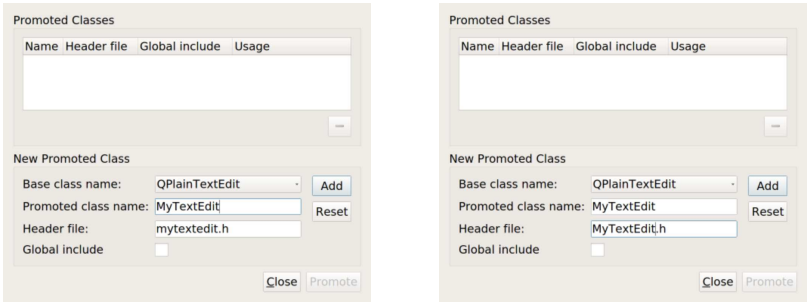
Per tal de fer l'explicació més concreta i entenedora, suposem que el widget que hom vol dissenyar s'usarà per mostrar un misstage de text i que serà derivat del `QPlainTextEdit`. El widget derivat s'identificarà com `MyTextEdit`. Com que cal programar-lo com una classe de **C++**, caldrà programar el fitxer de definició de la classe, `MyTextEdit.h`, i el fitxer de la implementació, `MyTextEdit.cpp`. Naturalment, caldrà afegir tots els mètodes que calgui corresponents a les noves funcions del widget derivat.

4.2 Inclusió del widget

La inclusió del nou widget en el disseny en curs del Designer contempla dues fases: la inclusió pròpiament dita i les definicions *signal-slot* corresponents a les funcions afegides. La inclusió s'efectua mitjançant l'operació `Promote` segons el següent procediment:

1. Entre els widget que ofereix el Designer, selecciona el que serà el widget base i inclou-lo en el canvàs.[1] En el cas que ens ocupa triem `QPlainTextEdit`.

[1] La paraula *canvàs* designa l'àrea el la qual l'usuari d'una aplicació gràfica defineix els grafismes.



(a) (b)

Figure 5: Identificació de la classe del nou widget.

2. Selecciona el widget en el canvàs i fes clic a sobre d'ell tot prement el botó dret del ratolí. Del menú desplegable selecciona l'opció `Promote to ...`.

3. Apareix una finestra com la que mostra la Figura 5, on es defineix el nom de la classe del nou widget. Si cal, modifica el nom assignat al fitxer de capçalera de la classe.

4. Ara cal fer clic al botó `Add` i, si tot ha anat bé, un clic sobre el botó `Promote` acaba el procès. A la finestra `Object Inspector`, el nom de la classe base `MyPlainTextEditor` haurà canviat al nom de la classe del nou widget. En el cas que seguim serà `MyTextEditor`.

La funcionalitat del nou widget es definirà mitjançant un conjunt d'*slots* i *signals* nous que no tenia el widget base. La definició es fa en el mateix moment d'establir les connexions. Suposem que la interfase inclou un `PushButton` i el widget nou `MyTextEdit` de manera que en fer clic sobre el botó, el nou widget mostri un text.

Comença seleccionant en el Designer l'operació de definició de connexions. En definir-ne una tal que el *signal* l'emeti el botó i el nou widget el rebi, s'obrirà una finestra d'edició de connexions com la mostrada a la Figura 6. Ara si, per exemple, hom selecciona en el botó l'acció `clicked()`, cal prèmer el botó `Edit` del nou widget i apareixerà la finestra de la Figura 7a. En aquesta finestra hom selecciona si cal definir un *slot* o un *signal* clicant sobre el símbol + corresponent.

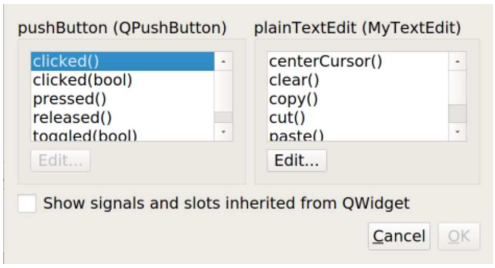
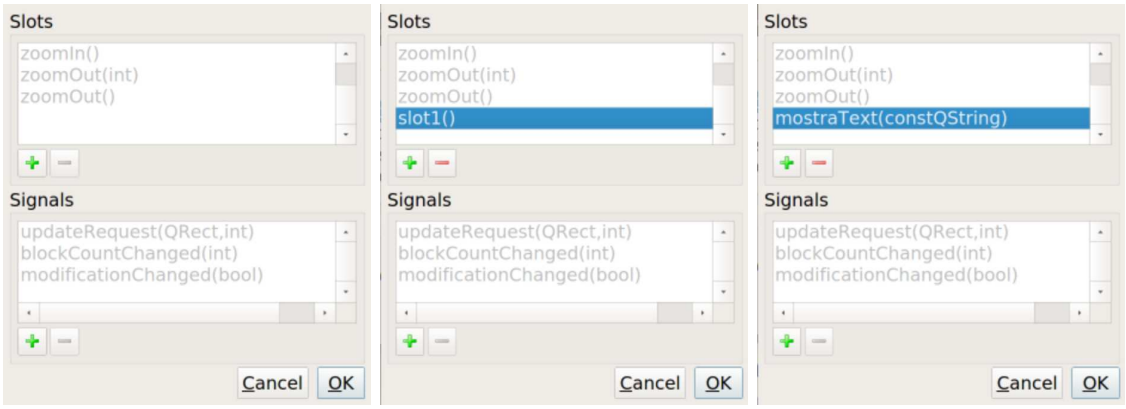


Figure 6: Edició de connexions noves.



(a) (b) (c)

Figure 7: Seqüència de definició d'un *slot* en un widget nou.

Per tal que el widget nou mostri el text, hom definirà un *slot* que tindrà per nom exactament el nom del mètode del widget programat per a que el rebi i el despatxi. Un click en el botó `OK` conclou la definició de la connexió funcional. La Figura 7 il.lustra la seqüència que caldrà repetir per a cada connexió *signal-slot* que involucri funcionalitats que no tenia el widget base.

L'operació caldrà repetir-la per a cada *slot* i *signal* nous que hom vulgui definir.

Malauradament, la inclusió de widgets nous programats per l'usuari fa que l'opció oferta pel Designer de prova de funcionament del disseny mitjançant les tecles `CTRL R` resta desactivada. Per tant, cal salvar el disseny en un fitxer `xxxx.ui` amb el nom adient i, tot seguit, processar amb `qmake` i `make`.