Cours QT

Pierre POMPIDOR

15 mars 2002

Création d'une application

```
#include <qapplication.h>
#include <qpushbutton.h>

int main (int argc, char **argv)
{
    QApplication a (argc, argv);
    QPushButton hello ("Hello world!");
    hello.resize (100, 30);

    a.setMainWidget (&hello);
    hello.show ();
    return a.exec ();
}
```

Un fichier d'entête par classe d'objets utilisée :
qpushbutton.h

La classe **QApplication** gère la file d'événements de l'application.
L'objet QApplication doit être créé avant tout autre objet Qt.

Pour dimensionner un PushButton :
– QPushButton : :**resize** (*largeur*, *hauteur*)

Si la QApplication : :**setMainWidget** de l'application est fermée, l'application est quittée.

Une widget n'est jamais visible quand vous la créez.
Vous devez appeler **show()** de la classe QWidget pour la rendre visible.

QApplication : :**exec** () met en place la boucle d'événements.

La gestion des événements :

```
#include <qapplication.h>
#include <qpushbutton.h>
#include <qfont.h>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    QPushButton quit( "Quit" );
    quit.resize( 75, 30 );
    quit.setFont( QFont( "Times", 18, QFont::Bold ) );

    QObject::connect( &quit, SIGNAL(clicked()),
                      &a,    SLOT(quit()) );

    a.setMainWidget( &quit );
    quit.show();
    return a.exec();
}
```

setFont de la classe QWidget fixe la fonte de la widget.
QFont (*famille*, *taille : 12 par défaut*, *poids : normal par défaut*, ...)

QObject : :**connect** : liaison entre deux objets Qt
– SIGNAL : pour envoyer des messages
– SLOT : pour en recevoir

SIGNAUX d'un PushButton :
– clicked ()
– pressed ()
– released ()

Affiliation des widgets :

```
#include <qapplication.h>
#include <qpushbutton.h>
#include <qfont.h>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    QWidget w;
    w.resize( 200, 120 );

    QPushButton quit( "Quit", &w );
    quit.move( 62, 40 );
    quit.resize( 75, 30 );
    quit.setFont( QFont( "Times", 18, QFont::Bold ) );

    QObject::connect( &quit, SIGNAL(clicked()),
                      &a,    SLOT(quit()) );

    a.setMainWidget( &w );
    w.show();
    return a.exec();
}
```

La classe **QWidget** est la classe de base de tous les objets d'interface.
Par exemple QPushButton hérite de QButton qui hérite de QWidget.

Ici, la widget ne "connaît" pas son fils.

Pour créer un objet Qt (exemples sur un PushButton) :
- QPushButton quit("Quit") ;
- QPushButton quit("Quit", **&w**) ;

Pour positionner un objet Qt :
QWidget : :**move** (*abscisse*, *ordonnée*)
par rapport au coin supérieur gauche de la widget mère

Création d'une nouvelle widget :

```
#include <qapplication.h>
#include <qpushbutton.h>
#include <qfont.h>

class MyWidget : public QWidget
{
public:
    MyWidget( QWidget *parent=0, const char *name=0 );
};


MyWidget::MyWidget( QWidget *parent, const char *name )
        : QWidget( parent, name )
{
    setMinimumSize( 200, 120 );
    setMaximumSize( 200, 120 );

    QPushButton *quit = new QPushButton( "Quit", this, "quit" );
    quit->setGeometry( 62, 40, 75, 30 );
    quit->setFont( QFont( "Times", 18, QFont::Bold ) );

    connect( quit, SIGNAL(clicked()), qApp, SLOT(quit()) );
}


int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    MyWidget w;
    w.setGeometry( 100, 100, 200, 120 );
    a.setMainWidget( &w );
    w.show();
    return a.exec();
}
```

`MyWidget( QWidget *parent=0, const char *name=0`
Le premier argument est la widget mère.
Le second argument est le nom de la widget,
(ce n'est pas le texte qui apparait dans la bannière).

Pour l'instant le retaillage n'est pas géré :
- setMinimumSize(200, 120) ;
- setMaximumSize(200, 120) ;

Pour créer un Qt objet (exemples sur PushButton) :
- QPushButton quit("Quit") ;
- QPushButton quit("Quit", &w) ;
- QPushButton *quit = new QPushButton ("Quit", this, **"quit"**) ;

Pour dimensionner un objet Qt :
- resize (*largeur*, *hauteur*)
- setGeometry (*abscisse*, *ordonnée*, *largeur*, *hauteur*) ;

Mise en place de plusieurs widgets filles :

```
#include <qapplication.h>
#include <qpushbutton.h>
#include <qscrollbar.h>
#include <qlcdnumber.h>
#include <qfont.h>

class MyWidget : public QWidget
{
public:
    MyWidget( QWidget *parent=0, const char *name=0 );
protected:
    void resizeEvent( QResizeEvent * );
private:
    QPushButton *quit;
    QScrollBar  *sBar;
    QLCDNumber  *lcd;
};


MyWidget::MyWidget( QWidget *parent, const char *name )
        : QWidget( parent, name )
{
    setMinimumSize( 200, 200 );

    quit = new QPushButton( "Quit", this, "quit" );
    quit->setGeometry( 10, 10, 75, 30 );
    quit->setFont( QFont( "Times", 18, QFont::Bold ) );

    connect( quit, SIGNAL(clicked()), qApp, SLOT(quit()) );

    lcd  = new QLCDNumber( 2, this, "lcd" );
    lcd->move( 10, quit->y() + quit->height() + 10 );

    sBar = new QScrollBar( 0, 99,
                           1, 10,
                           0,
                           QScrollBar::Horizontal,
                           this, "scrollbar" );

    connect( sBar, SIGNAL(valueChanged(int)),
             lcd,  SLOT(display(int)) );
}

void MyWidget::resizeEvent( QResizeEvent * )
{
    sBar->setGeometry( 10,           height() - 10 - 16,
                       width() - 20, 16 );
    lcd->resize( sBar->width(), sBar->y() - lcd->y() - 5 );
}

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    MyWidget w;
    w.setGeometry( 100, 100, 200, 200 );
    a.setMainWidget( &w );
    w.show();
    return a.exec();
}
```

La widget doit “connaître” ses filles pour les retailler : création de variables privées

- qpushbutton.h → QPushButton → quit
- qscrollbar.h → QScrollBar → sBar
- qlcdnumber.h → QLCDNumber → lcd

Pour connaître le positionnement d’un objet QT :
- objet->**x**()
- objet->**y**()
- objet->**height**()
- objet->**width**()

La classe **QScrollBar** :
Constructeur **QScrollBar** avec comme arguments :
- int minValue, int maxValue : rang des “valeurs”
- int LineStep : valeur de chaque “ligne” (ici 1)
- int PageStep : valeur de chaque “page”(ici 10)
- int value : valeur initiale
- Orientation
- QWidget * parent=0 : widget mère
- const char * name=0 : nom de la widget

Signaux de **QScrollBar** :
- valueChanged (int value)
- sliderPressed () → slider = curseur
- sliderMoved (int value)
- sliderReleased ()
- nextLine ()
- prevLine ()
- nextPage ()
- prevPage ()

La classe **QSlider** :
Constructeur **QSlider** avec comme arguments :
- int minValue, int maxValue : rang des “valeurs”
- int step : valeur de chaque pas (ici 1)
- int value : valeur initiale
- Orientation
- QWidget * parent=0 : widget mère
- const char * name=0 : nom de la widget

Signaux de **QSlider** :
- valueChanged (int value)
- sliderMoved (int value)
- sliderPressed ()
- sliderReleased ()

Gestion de la connexion signaux/slots :

Déclaration des signaux ou des slots :

Lors de la création d'une classe dont des méthodes vont émettre ou recevoir des signaux, vous devez intégrer du code spécifique (mots clefs **signals** et**slots**) :

```
class MyWidget : public QWidget
{
   Q_OBJECT

   signals:
      void monSignal1 ( ... );
      ...

   public slots:
      void monSlot1 ( ... );
      ...

   ...
};
```

Les signaux seront émis par **emit** :
Par exemple : `emit monSignal1 ( 10 );`

Au niveau de la compilation :

Les classes utilisant des signaux ou des slots doivent être exécutées par le moc ("Compilateur de méta-objets") fourni avec Qt. C'est lui qui génère la glu nécessaire au mécanisme signaux/slots.

```
point_d_entree: MyWidget

MOC=/usr/lib/qt-2.3.0/bin/moc
INC=-I/usr/lib/qt-2.3.0/include
LIB=-L/usr/lib/qt-2.3.0/lib

moc_MyWidget.cc: MyWidget.h
        $(MOC) $(INC) MyWidget.h -o moc_MyWidget.cc

moc_MyWidget.o: moc_MyWidget.cc
        g++ $(INC) -c moc_MyWidget.cc

MyWidget.o: MyWidget.cc MyWidget.h
        g++ $(INC) -c MyWidget.cc

main.o: main.cc
        g++ $(INC) -c main.cc

MyWidget:  MyWidget.o main.o moc_MyWidget.o
        g++ $(LIB)  MyWidget.o moc_MyWidget.o main.o -o MyWidget -lqt
```