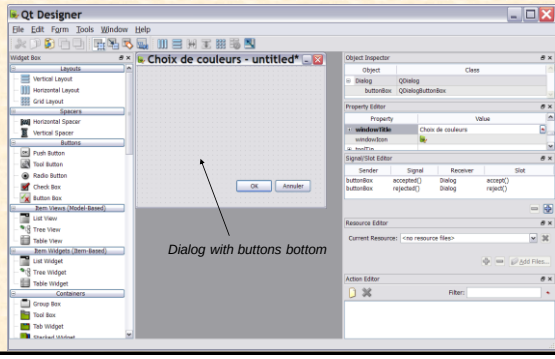
QT : "Code less. Create more."

- QT édité par trolltech, puis nokia et enfin http://qt.digia.com et http://qt-project.org :
  - Ensemble d'outils et de bibliothèques
    - *Cross-platform*
    - *C++ (java QT Jambi)*
    - *Existe en version OpenSource y compris pour Windows (QT4)*
  - QT3 et QT4 (version actuelle 4.8.3, QT5 beta)
  - Qtopia (Greenphone)…
  - Outils
    - *QT Assistant : aide !*
    - *QT Linguist : traducteur*
    - *QT Designer : conception d'IHM et générateur de code*
    - *QT Demo : Démonstration des possibilités de QT*
  - Implémentation totale du MVC (depuis QT4)
  - Intégrable à Eclipse, Kdevelop, Visual Studio (Express & PSDK) et qtcreator

QT : "Code less. Create more."

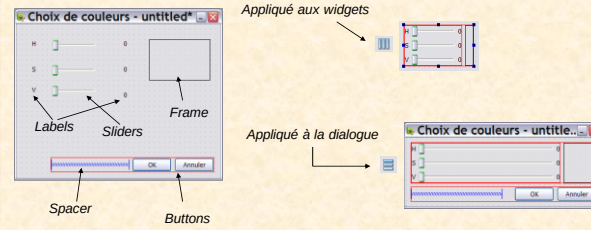
- QT édité par trolltech, puis nokia et enfin http://qt.digia.com et http://qt-project.org :
  - Documentation en ligne http://qt-project.org/doc
  - Documentation locale QT Assistant (**assistant**)
  - Documentation en livres !
    - *BU : QT3 et QT4*
- QT Designer (**designer**)
  - QT3 Designer : presque IDE
  - QT4 Designer : uniquement l'IHM et génération de code
  - Basé sur la notion de widgets, objets…
  - Envoi de messages d'objet à objet : slots et connectors

QT4 Designer



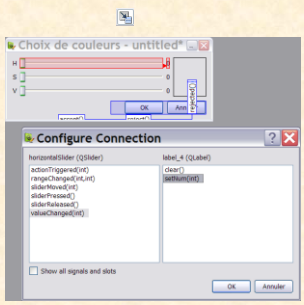
QT4 Designer

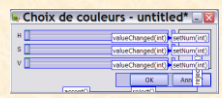
Agencement des contrôles (layouts)



QT4 Designer

Slots et connectors





Ctrl+R et hop ça marche !

QT4 Designer

- Il faut penser à la génération de code
  - Nommer correctement les objets utilisés de la *QDialog* y compris la boîte de dialogue elle-même (propriété *objectName*. Par exemple, *dlgChooseColor*, *_slHue*, *_slSaturation*…)
  - Ne pas oublier l'ordre des focus
  - Sauvegarder : un fichier *.ui* est créé
- Créer le programme principal

QT4

```
    // include QT4 : définition d'une application et utilisation de QDialog
#include <QApplication>
#include <QDialog>

    // Utilisation d'un fichier généré par l'outil uic
    //  à partir du fichier dlgChooseColor.ui
#include "ui_dlgChooseColor.h"

    //
int main( int argc, char *argv[] )
{
        // Déclaration de variables
    QApplication app( argc, argv );
    QDialog dlg;
    Ui::dlgChooseColor ui;  // User Interface de la boîte de dialogue

    ui.setupUi( &dlg );     // Assiciation de QDialog avec l'interface utilisateur
    dlg.show( );            // Affichage de la boîte de dialogue
    return app.exec( );     // Boucle d'exécution de l'application
}
```

Programme principal

QT4 *(Visual Studio)*

- qmake -project *-tp vc* -o <*projectName.pro*>
  - *Le fichier .pro est créé*

```
TEMPLATE = vcapp
TARGET =
DEPENDPATH += .
INCLUDEPATH += .

# Input
FORMS += dlgChooseColor.ui
SOURCES += main.cpp
```

- qmake
  - Création des fichiers pour la compilation (Makefile, .vcproj…)
- make, nmake, F7…
- MVC, les sliders et les labels fonctionnent correctement mais la couleur n'est pas la bonne… Pas mieux que Ctrl+R
- Ajout de la gestion spécifique de la boîte de dialogue

QT4

```
#ifndef __CHOOSE_COLOR__
#define __CHOOSE_COLOR__ 1

#include <QDialog>
#include "ui_dlgChooseColor.h"

    // Définition de la classe dlgChooseColor
    // Cette classe hérite des classes QDialog et Ui::dlgChooseColor
    //  Le namespace Ui contient la définition de la classe Ui::dlgChooseColor
    //  Ui::dlgChooseColor vient de la génération de code à partir du fichier .ui
class dlgChooseColor : public QDialog, protected Ui::dlgChooseColor
{
        // Déclaration obligatoire pour déclarer la classe comme étant QT
    Q_OBJECT

public:
        // Constructeur
    dlgChooseColor( QWidget * parent = 0 );

protected slots:
        // slot de connexion pour changer la couleur du frame de la boîte de dialogue
    void changeColor( );

};

#endif // __CHOOSE_COLOR__
```

Code spécifique pour la couleur dans le cadre

QT4

```
#include "dlgChooseColor.h"

dlgChooseColor::dlgChooseColor( QWidget * parent )
    : QDialog( parent )
{
        // Initialisation de l'Interface Utilisateur
    setupUi( this );

        // Gestion des messages : les sliders envoient l'info à la dialogue et appellent changeColor()
    connect( _slHue,        SIGNAL( valueChanged(int)), this, SLOT( changeColor() ) );
    connect( _slSaturation, SIGNAL( valueChanged(int)), this, SLOT( changeColor() ) );
    connect( _slValue,      SIGNAL( valueChanged(int)), this, SLOT( changeColor() ) );
}

void dlgChooseColor::changeColor( )
{
    QColor color;   // On travaille en couleur codée en HSV
    color.setHsv( _slHue->value(), _slSaturation->value(), _slValue->value() );

    QPalette palette;   // Création de la palette représentant la couleur pour une fenêtre
    palette.setColor( QPalette::Window, color );

        // La couleur est affectée
        // La propriété autoFillBackground est à true (QTDesigner) donc la frame se redessinera seule
    _frameColor->setPalette( palette );
}
```

Code spécifique pour la couleur dans le cadre

QT4

```
    // include QT4 : définition d'une application et utilisation de QDialog
#include <QApplication>
#include <QDialog>

    // include de la définition de notre boîte de dialogue
#include "dlgChooseColor.h"

    //
int main( int argc, char *argv[] )
{
        // Déclaration de variables
    QApplication app( argc, argv );
        // Déclaration de notre boîte de dialogue
    dlgChooseColor dlg;
    dlg.show( );            // Affichage de notre boîte de dialogue
    return app.exec( );     // Boucle d'exécution de l'application
}
```

Nouvelle version du Programme principal

QT4 *(Visual Studio)*

- qmake -project *-tp vc* -o <*projectName.pro*>
  - *Le fichier .pro est créé/modifié*

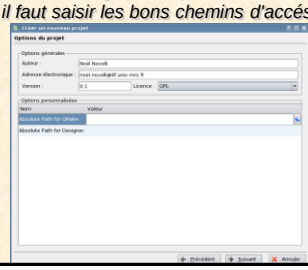
```
TEMPLATE = vcapp
TARGET =
DEPENDPATH += .
INCLUDEPATH += .

# Input
HEADERS += dlgChooseColor.h
FORMS += dlgChooseColor.ui
SOURCES += dlgChooseColor.cpp main.cpp
```

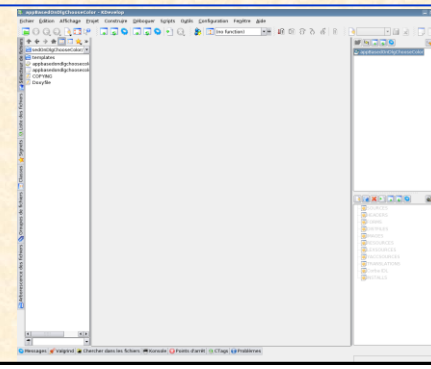
- qmake
  - Création des fichiers pour la compilation (Makefile, .vcproj…)
- make, nmake, F7…
- La *Frame* change de couleur !!!!!

QT4 *(KDevelop)*

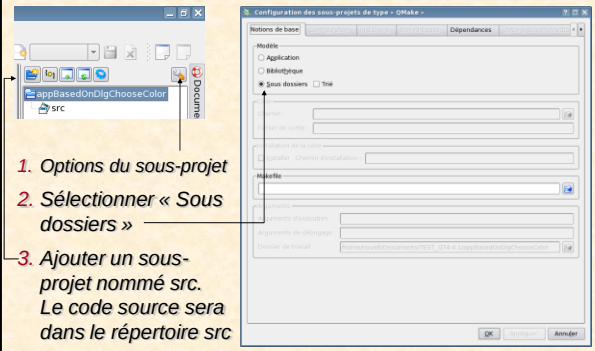
- kdevelop en mode C/C++
  - *Créer un nouveau projet vide C++, QMake project.*
  - *Si vous avez configuré Qt4 par défaut, rien d'autre à faire sinon il faut saisir les bons chemins d'accés.*



QT4 *(KDevelop)*



QT4 *(KDevelop)*



1. *Options du sous-projet*
2. *Sélectionner « Sous dossiers »*
3. *Ajouter un sous-projet nommé src. Le code source sera dans le répertoire src*

QT4 *(KDevelop)*

- Choisir « Ajouter les fichiers existants »
- Ajouter tous vos codes sources : .cpp, .h, .ui...
  - Cette action effectue une recopie des fichiers dans votre projet.
- L'arborescence du projet est complète.
- Choisir « Option du sous-projet » pour src.
- Dans l'onglet « Notions de base », saisir ../bin dans Cible/Chemin et nommé votre exécutable.
- Compiler « Construire le projet » et exécuter Qmake
- Exécuter.

QT4 *(qtcreator)*

QT4 *(qtcreator)*

- Cf TP2 ;)