|  |  |  |
| --- | --- | --- |
| **BTS2 TP #3** | **Initiation à Qt** | **2020/21** |

# Objectifs

Connaître le squelette minimal d’un programme avec QT.  
Comprendre le mécanisme des signaux et slots.

# Configuration

Dans un premier temps Qt creator n'est pas correctement configuré. Il faut associer à Qt creator une version de Qt. Pour se faire il faut suivre les étapes suivantes :

|  |  |
| --- | --- |
| Aller dans « Outils > Options… » |  |
| Dans la partie « Compiler et Exécuter » |  |
| Sélectionner l'onglet « Version de Qt » |  |
| Appuyer sur le bouton Ajouter en haut à droite pour aller chercher le bon exécutable. On cherche l'executable qmake qui correspond à la version de Qt que l'on souhaite utiliser.  Chemin vers l'executable : « C:\Qt\4.8.4\bin »  Une fois le chemin sélectionné il ne reste plus qu'à double-cliquer sur le qmake et une version de Qt est rajoutée. |  |
| Appuyer sur le bouton Appliquer en bas à droite puis sur OK pour prendre en compte les modifications. |  |
| Sélectionner l'onglet « Projet non configuré » |  |
| Sélectionner pour les 2 champs la bonne version de Qt ainsi que le bon compilateur C++, comme sur l'image ci-contre. |  |

# Premier projet

## Création du projet

A présent nous pouvons créer notre premier programme Qt.

|  |  |
| --- | --- |
| Cliquer sur : |  |
| Sélectionner Application graphique Qt et cliquer sur Choisir… |  |
| Sur la page suivante donner un nom au projet et sélectionner le chemin du projet (au choix) les répertoires sont à créer s’ils n'existent pas. |  |
| Valider les étapes suivantes. Le projet est créé. |  |
| Appuyer sur la flèche verte en bas à gauche. |  |
| le programme se lance, la fenêtre est vide. |  |
| Dans la partie projet double cliquer sur le fichier mainwindow.ui. |  |

|  |  |
| --- | --- |
| La fenêtre suivante apparaît :    Détail du widget (méthodes et attributs) sélectionné  Contenu de la fenêtre  Aperçu de la fenêtre  Liste des widgets  Nous sommes à présent dans la partie Design c'est l'endroit où nous allons modifier notre fenêtre : | |
| Utiliser la touche « F1 » pour avoir de l’aide sur les classes des widgets et des classes Qt.  La documentation est disponible sur <https://doc.qt.io/> |  |

## Fichier mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~*MainWindow*()

{

delete ui;

}

this est un pointeur sur l'objet MainWindow

ui est un pointeur sur les widgets de la fenêtre.

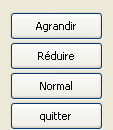
## Signaux et slots

|  |  |
| --- | --- |
| Pour ajouter un bouton sur la fenêtre faire un copier-coller du widget vers la fenêtre | Sans titre.bmp |
| Pour associer un slot sur le click du bouton, passer en mode slots (F4), cliquer sur le bouton et déplacer le curseur dans une zone de la fenêtre.  La liste des signaux et slots s'affiche. | Sans titre.bmp |
| Choisir le signal et le slot associé, ici, on choisit pushButton, clicked() -> MainWindow, close(). | Sans titre.bmp |
| Valider et visualiser le résultat graphique. | Sans titre.bmp |
| Résultat : un click sur le bouton ferme la fenêtre.  Lancer et tester l’application en appuyant sur la flèche verte en bas à gauche. |  |
| On peut aussi associer un signal à un slot en faisant un clic droit sur le widget et choisir "Aller aux slots". | Sans titre.bmp |
| Cela va générer un squelette de fonction pour le slot dans la classe MainWindow. | void MainWindow::on\_pushButton\_2\_clicked()  {    } |
| Pour mettre des traces dans votre programme vous pouvez utiliser qDebug().  qDebug s’utilise comme cout.  Les messages soumis à qDebug() s'affichent sur l'onglet "Sortie de l'application". | #include <QDebug>  ...  qDebug() << " je suis passé par ici"; |

# Exercices

## Exercice 1

1. Créer une application graphique QT Windows (fenêtre) dont le titre est " TP QT BTS2 ".
2. Ajouter 4 boutons



Utiliser l'association des signaux et slots (F4) pour réaliser ces actions (sauf bouton Agrandir ):

* Bouton Agrandir -> affiche la fenêtre en taille maximale, pour cela utiliser l'option "**Aller aux slots"**
* Bouton Réduire -> réduit la fenêtre dans la barre des tâches
* Bouton Normal -> affiche la fenêtre dans sa taille d'origine
* Bouton Quitter -> ferme l'application

## Exercice 2

On va maintenant ajouter un widget ComboBox et un widget LineEdit.

Le but est de renseigner le LineEdit par le texte sélectionné dans le ComboBox.

Utiliser l'association des signaux et slots (F4) pour réaliser cette action :

Sans titre.bmp

On utilisera la méthode addItem pour remplir la ComboBox dans la classe MainWindow.

## Exercice 3

Dans un GroupBox on va ajouter deux RadioButton. Suivant la sélection on va rajouter une image de fond à la fenêtre.



Pour cela on va rajouter un slot dans la classe MainWindow pour traiter le traitement des RadioButton.

Par programmation, connecter (avec la primitive connect) le signal clicked des radioButton sur notre slot.

On utilisera les méthodes suivantes :

setStyleSheet("background-image: url(D:/collines.jpg)");

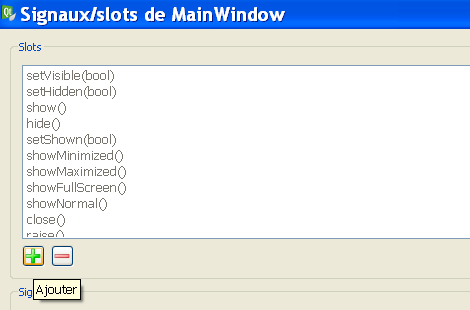
setStyleSheet("background-image: none");

## Exercice 4

On va rajouter un widget LineEdit qui va nous permettre de changer le titre de la fenêtre.

Le titre sera modifié sur l'action du la touche « Entrée ». Pour cela on va créer un slot dans la classe MainWindow qui changera le titre de la fenêtre en récupérant le texte du LineEdit saisi.

On va rajouter ce slot dans la liste des slots de MainWindow.



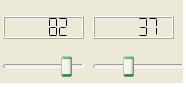
On va ensuite faire la connexion entre le bon signal et ce slot.

Sans titre.bmp

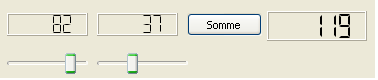
Réaliser cette fonctionnalité.

## Exercice 5

On va utiliser ici le passage de paramètres entre signal et slot. On va rajouter dans notre fenêtre deux Slider et LCD Number, chaque afficheur LCD étant relié à un des sliders et affiche la valeur de son slider.



On veut maintenant afficher la somme des valeurs des deux LCD Number dans un troisième LCD Number sur l'action d'un click sur un bouton. Le click sur le bouton doit envoyer un signal à un slot du LCD Number qui affiche la somme.



On va donc créer une classe myLCD qui hérite de QLCDNumber. Dans cette classe on va définir notre slot qui reçoit deux paramètres et qui affiche la somme des paramètres.

Le troisième LCD sera donc créé dynamiquement.

Il va falloir créer un signal dans notre classe MainWindow avec deux paramètres (valeurs des LCD )

On a donc la séquence suivante : un click sur bouton Somme déclenche l'émission d'un signal qui est relié au slot du troisième LCD qui affiche la somme des paramètres qu'il a reçus.

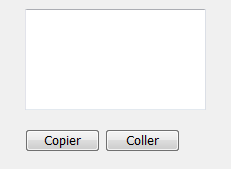
A l'aide des fonctions connect et emit, réaliser par programmation cette fonctionnalité. Il faudra créer la classe myLCD à l’aide de la fonctionnalité « Add New… ».

**On rappelle qu'un signal et un slot sont des fonctions membres d'un widget**. Un signal contrairement au slot ne contient pas de définition.

## Exercice 6

On va rajouter sur notre fenêtre un widget Textedit et deux boutons Copier et Coller pour copier du texte et le coller dans notre TextEdit.

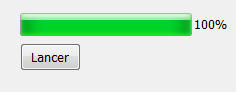
Utiliser l'association des signaux et slots (F4) pour réaliser cette action.



## Exercice 7

Ajoutons un widget Progress bar. Sur un bouton Lancer, lançons la progression de la barre en prenant comme unité le centième de seconde. Il va falloir dans le slot du bouton faire une boucle allant de 0 à 100 et qui sera temporisée pendant 100ms. Pour temporiser notre programme on utilisera la méthode msleep(int) de la classe QThread qui suspend le thread.

Réaliser cette fonctionnalité.



## Exercice 8

Ajoutons un menu sur notre fenêtre. Ce menu pour l'instant n'a que l'option Quitter qui fermera notre application. On utilisera le signal triggered() sur l'action du menu associé au slot close() de l'application.

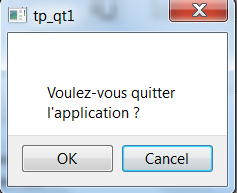
Réaliser cette fonctionnalité.

## Exercice 9

Maintenant on peut fermer notre application soit par le menu, soit par la croix, soit par le bouton Quitter. Pour centraliser la fermeture de l'application on va surdéfinir la méthode [QWidget::closeEvent(QCloseEvent \* event)](http://qt.developpez.com/doc/4.7/qwidget/#closeevent) pour notre classe. Cette méthode reçoit les événements de fermeture. Son implémentation par défaut accepte l'événement de fermeture. Si on veut ignorer l'évènement de fermeture il faudra utiliser la méthode event->ignore() et event->accept() pour accepter l'évènement.

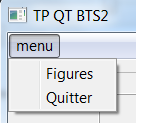
Dans notre méthode closeEvent on va demander une confirmation pour quitter l'application grâce à une QMessageBox msgBox.

Réaliser cette fonctionnalité

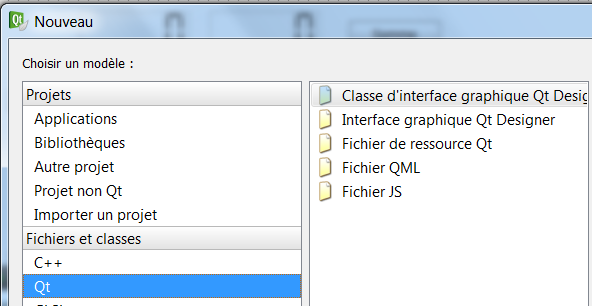


## Exercice 10

Dans le menu, insérer une option "Figures" qui va ouvrir une nouvelle fenêtre du type boite de dialogue.

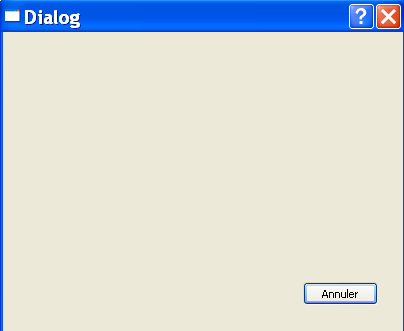
.

Pour créer la fenêtre on va rajouter au projet une nouvelle classe d'interface graphique.



Pour ouvrir cette fenêtre à partir du menu on va utiliser le signal SIGNAL(triggered()) et le slot showDialogModal() (à créer) dans lequel on lancera la fenêtre par la méthode exec();

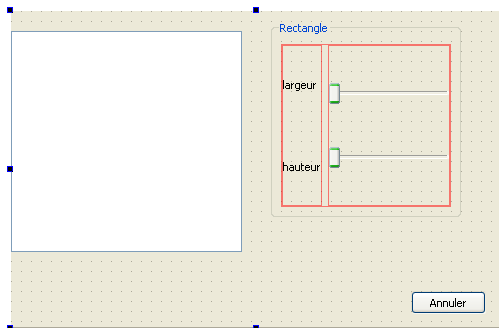
Dans cette fenêtre, placer un bouton pour quitter cette fenêtre.

.

Le bouton "Annuler" devra fermer la fenêtre.

Pour créer la fenêtre il va falloir la générer graphiquement et créer un objet de cette fenêtre dynamiquement dans le constructeur de la fenêtre principale dlg = new Dialog(this);

Dans cette fenêtre, on va l’écran suivant :



La cadre de gauche est un objet de type QGraphicsView. On ajoute deux sliders.

On va maintenant créer un rectangle dans notre zone graphique.

m\_Scene = new QGraphicsScene(this);

ui->graphicsView->setScene(m\_Scene);

// permet de fixer la partie de la scene que l'on souhaite observer

// cette partie est un rectangle d'origine 0,0 et de largeur=hauteur=200

m\_Scene->setSceneRect(0, 0, 200, 200);

// création du rectangle

m\_Rectangle = m\_Scene->addRect(10, 10, 80, 50);

Pour chaque slider, créer un slot pour modifier la longueur ou la largeur du rectangle.

Exemple pour la largeur :

//slot de gestion de la largeur du rectangle

void Dialog::LargeurChange(int largeur) {

QRectF rect;

rect = m\_Rectangle->rect();

rect.setWidth(largeur);

m\_Rectangle->setRect(rect);

}

Il faut faudra donc connecter le changement des sliders avec les dimensions du rectangle (signal -> slot)

1. Réaliser cette fonctionnalité
2. Réaliser cette fonctionnalité avec un cercle