***Objectifs pédagogiques:***

- Créer des interfaces graphiques GUI (**graphical user interface)** à l’aide d’un RAD (*Rapid Application Development ou développement rapide d'applications)* (référentiel BTS SN savoir : S4.8 )

- Gérer la disposition des objets dans une interface graphiques à l’aide des layouts

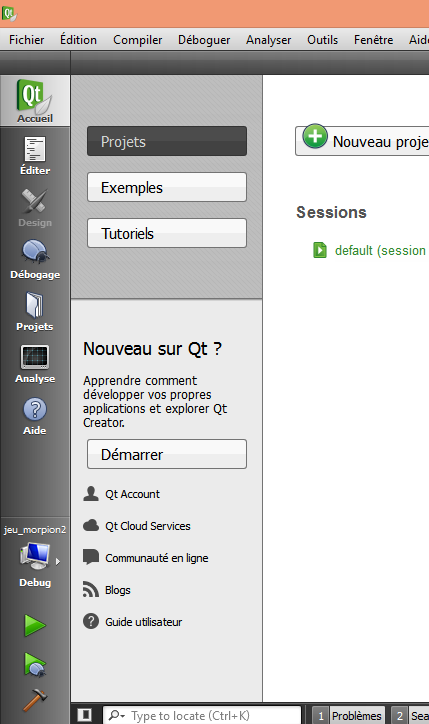
- Gestion des événements de l’interface graphique (Signal / Slot )

[I. Création d’un projet Qt et d’une première fenêtre (Graphical User Interface : GUI) 1](#_Toc439012510)

[II. Création d’une fenêtre avec gestion d’un layout 3](#_Toc439012511)

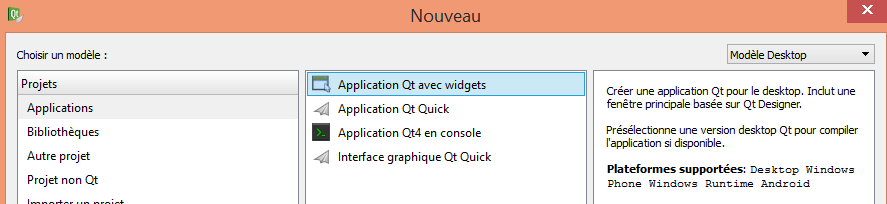
[III. Mise en œuvre d’interaction dans la fenêtre : 5](#_Toc439012512)

1. Création d’un projet Qt et d’une première fenêtre (Graphical User Interface : GUI)
2. Lancer l’application Qt Creator, vous avez la possibilité de consulter des exemples de programmes qui fonctionnent ou des tutoriels afin de comprendre comment fonctionne la programmation sur Qt.

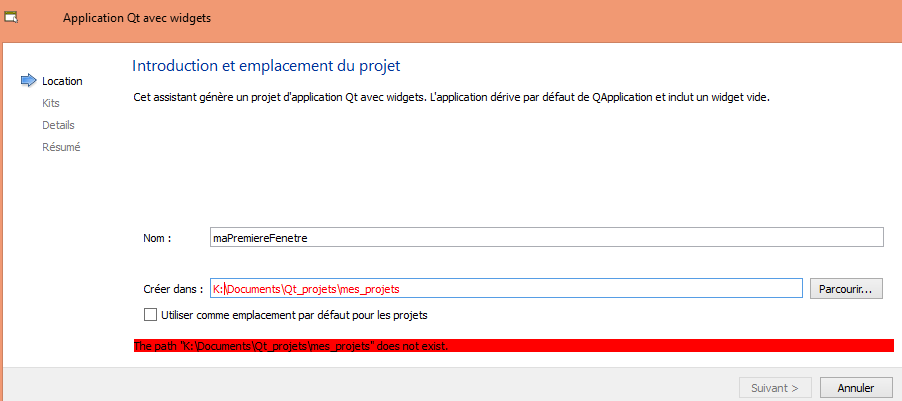


La barre de navigation sur la gauche permet de revenir à la page d’accueil, d’éditer notre programme, de consulter l’aide, de compiler (le marteau) et d’exécuter avec le bouton Play.

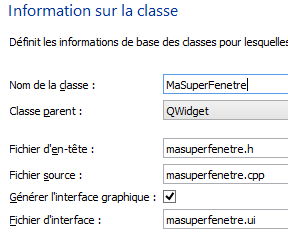
1. Créer votre premier projet, pour cela, sélectionner l’onglet Fichier -> Nouveau fichier ou projet



1. Selectionner le nom et l’emplacement de votre dossier projet.



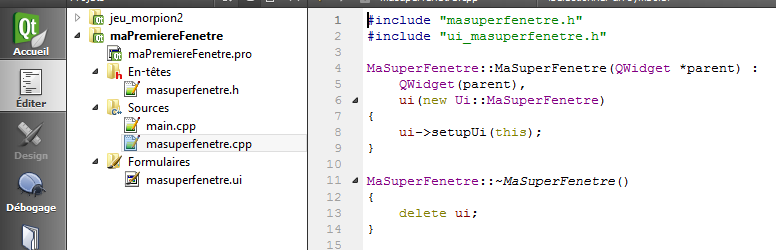
1. Sélectionnez le kit par défaut.
2. Selectionnez la classe parente QWidget et non QMainWindow que l’on verra plus tard. Nommez votre classe MaSuperFenetre.



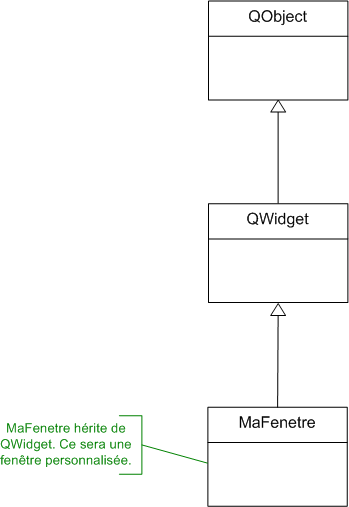
1. Cliquer sur terminez.

*Qt a généré pour vous une application de type fenêtre simple, en créant tous les fichiers nécessaires :*

* *La classe « MaSuperFenetre » avec son .h et .cpp*
* *Le main.cpp pour créer une instance de MaSuperFenetre*
* *Un fichier masuperfenetre.pro qui est spécifique à Qt qui permet d’avoir des modules offrant des possibilités à notre application*
* *Le formulaire permet de dessiner l’interface graphique sans programmer d’extension.ui*



On voit que l’objet w de la classe MaSuperFenetre est créé dans le main.cpp.



Et dans MaSuperFenetre.h on voit que notre classe hérite de QWidget qui elle-même hérite de QObject (nous y reviendrons pour la méthode connect pour rendre notre fenêtre intéractive).

La méthode w.show() dans le main.cpp permet d’afficher l’objet w qui est votre super fenêtre !!!

(CF annexe 1 pour voir tous les QWidgets utilisables)

1. Lancer la compilation et l’exécution avec le bouton Play.
2. Création d’une fenêtre avec gestion d’un layout

Dans Qt, un QWidget peut contenir d’autres QWidget qui sont des objets de type graphiques. Nous allons donc créer des QPushButton (bouton à presser), QLabel (permettant d’écrire du texte ou d’afficher des images) et des lignes de textes à saisir QLineEdit.

1. Dans le constructeur de votre fenetre, ajouter la création dynamique d’une instance de la classe QPushButton :

*QPushButton \* monBouton = new QPushButton(« Pressez ici ! »,this) ;*

« This » indique que l’objet w est le parent de l’objet monBouton.

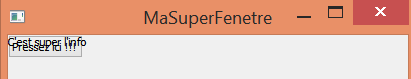
Ajouter l’include pour que Qt reconnaisse cette classe : #include <QPushButton>

*Très utile :*

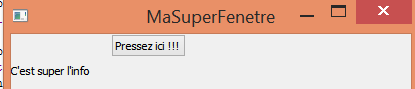
*Si vous voulez des détails sur une classe ou une méthode, cliquez dessus et pressez F1, qui lance l’aide sur la méthode.*

1. Lancer l’éxécution et vérifiez que votre bouton apparait dans la fenêtre.
2. Créer à présent d’après ce qui précède et de la documentation un objet QLabel monTexte permettant d’écrire le texte « C’est super l’info » et tester votre programme.

On constate que les objets sont positionner en haut à gauche par défaut dans la fenêtre.



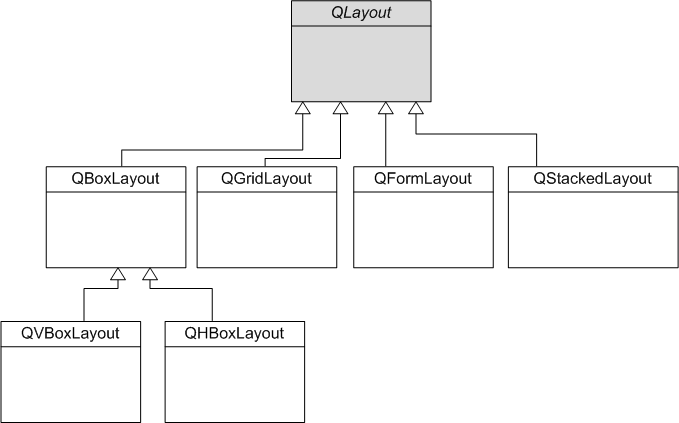
1. Pour améliorer cela, modifier leur position grâce à la méthode : move(int x, int y) ;



La méthode consistant à précisez de manière chiffrée est une mauvaise méthode car si vous modifiez la taille de la fenêtre, votre affichage ne s’adaptera pas.

*A retenir :*

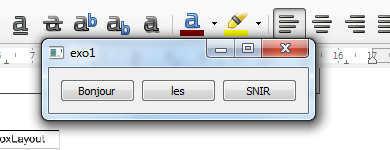
C’est la raison pour laquelle nous allons utiliser les layouts (gestionnaire de positions des widgets afin qu’il gère les positions des objets pour nous).



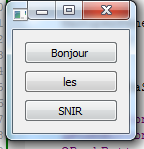
Voir aussi :

<https://openclassrooms.com/courses/programmez-avec-le-langage-c/positionner-ses-widgets-avec-les-layouts>

1. Créer toujours dans le constructeur de la fenêtre, un objet dynamique de la classe QHBoxLayout. (layout horizontale).

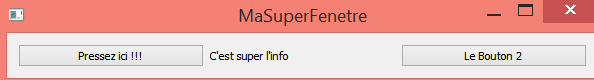


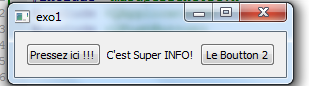
1. Ajouter les objets monbouton et monTexte à l’objet QGridLayout que vous avez créés (trouver la méthode en regardant l’aide de QHBoxLayout ou Qlayout).



On s’aperçoit que les widgets sont alignés Horizontalement !!!

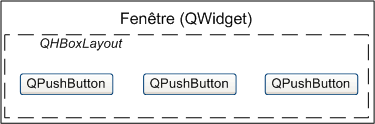
1. Ajouter un nouveau bouton monBouton2. Et ajoutez-le au layout de votre fenêtre. Modifiez la taille de la fenêtre vous remarquez que tout reste correctement aligné, et la taille des objets varient.





**Schéma des conteneurs**

En résumé, la fenêtre contient le layout qui contient les widgets. Le layout se charge d'organiser les widgets.  
Schématiquement, cela se passe donc comme à la figure suivante.



On vient de voir le layout QHBoxLayout qui organise les widgets horizontalement.

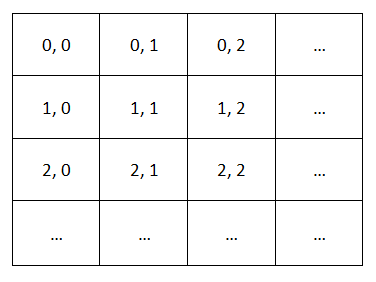
**Le layout de grille**

Les layouts horizontaux et verticaux sont gentils mais il ne permettent pas de créer des dispositions très complexes sur votre fenêtre.

C'est là qu'entre en jeu QGridLayout, qui est en fait un peu un assemblage de QHBoxLayout et QVBoxLayout. Il s'agit d'une disposition en grille, comme un tableau avec des lignes et des colonnes.

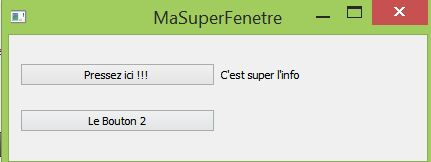
**Schéma de la grille**

Il faut imaginer que votre fenêtre peut être découpée sous la forme d'une grille avec une infinité de cases, comme à la figure suivante.

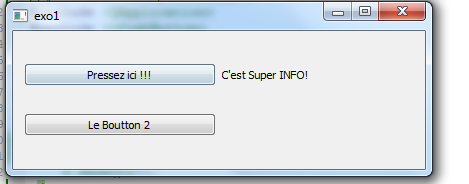


Si on veut placer un widget en haut à gauche, il faudra le placer à la case de coordonnées (0, 0).  
Si on veut en placer un autre en-dessous, il faudra utiliser les coordonnées (1, 0) et ainsi de suite.

1. Modifier votre layout horizontal en un QGridLayout (layout en forme de grille) et placer les widgets en (0,0), (0,1) et (1,1).



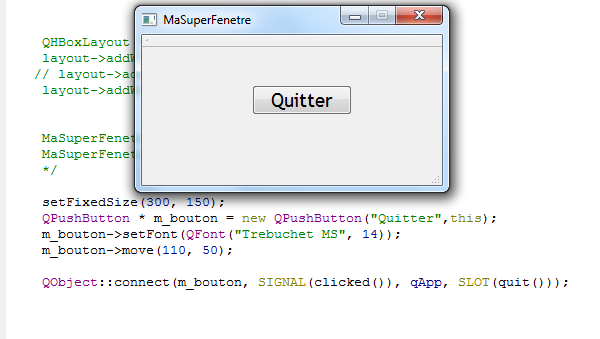
Il est également possible d’imbriquer les layouts afin d’organiser la fenêtre en sous partie.



1. Mise en œuvre d’interaction dans la fenêtre :

Dans Qt, il est possible de créer des messages interactifs entre objet à l’aide de la méthode QObject ::connect() avec les signaux et slots.

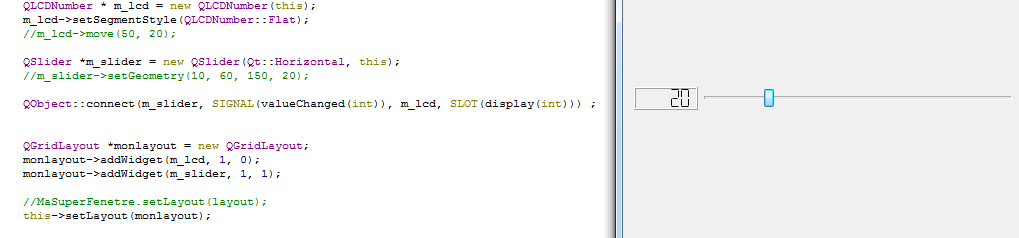
Cf : Aide sur Qt dans : Alire\Mechehar\Qt\Aide sur Qt

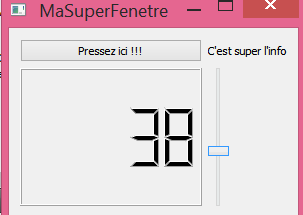


1. Signaux et Slots déjà présent avec les objets de base :
2. Créer 2 objets, un QLCDNumber et un QSlider puis ajouter-les à votre layout grid en (1,0) et en (1,1).

A présent on veut que lorsque l’on clique sur la glissière QSlider monSlider, on veut que le SIGNAL « valueChanged(int) » déclenche un changement du SLOT display(int) de l’objet QLCDNumber monLCD.

1. Pour cela, consulter l’aide Qt et utiliser la méthode QObject ::connect( ) entre ce SIGNAL et ce SLOT. Tester votre connexion.





1. Créer une connexion qui permet lors d’un appui sur le bouton « Pressez ici » de faire disparaitre le texte « C’est super l’info »
2. Créer ses propres signaux et slots

Voici maintenant une partie très intéressante, bien que plus délicate. Nous allons créer nos propres signaux et slots.

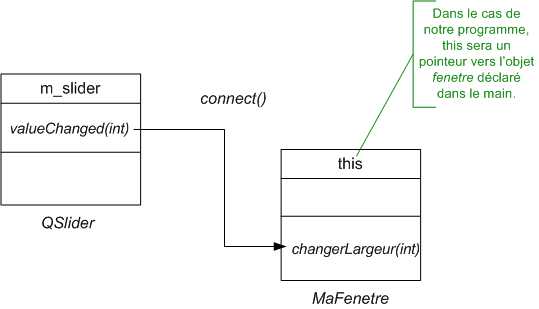
En effet, si en général les signaux et slots par défaut suffisent, il n'est pas rare que l'on se dise « Zut, le signal (ou le slot) dont j'ai besoin n'existe pas ». C'est dans un cas comme celui-là qu'il devient indispensable de créer son widget personnalisé.

Pour pouvoir créer son propre signal ou slot dans une classe, il faut que celle-ci dérive directement ou indirectement de QObject. C'est le cas de notre classe MaFenetre : elle hérite de QWidget, qui hérite de QObject. On a donc le droit de créer des signaux et des slots dans MaFenetre.

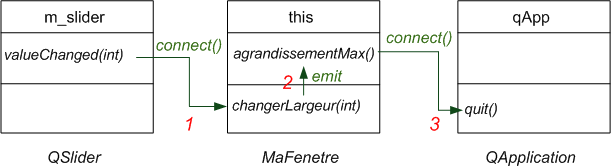
Nous allons commencer par créer notre propre slot, puis nous verrons comment créer notre propre signal.

1. Regarder la page web suivante et créer le slot changerLargeur(int)du schéma suivant :

<https://openclassrooms.com/courses/programmez-avec-le-langage-c/les-signaux-et-les-slots-2>



1. A présent créer le SIGNAL void agrandissementMax( ) et déclencher la fin de l’application pour une valeurLargeur de 60.
2. Et enfin déclencher la fin de l’application à l’aide du schéma suivant.



Vous avez réalisez une fenêtre avec des gestions d’événements.

