Les GUI avec Qt

Emmanuel Guiraud - Quentin Philippot

16 Octobre 2014

Sommaire

Sommaire

Sommaire

CLI : Console, très chère console ...

Command Line Interface

La plus ancienne des interfaces conversationnelles développées sur des ordinateurs. Avant cela, c'était le règne des cartes perforées.

Aujourd'hui : utilisation du terminal.

Command Line Interface

```
quentin@RED:~$ ls -al
total 292
drwxr-xr-x 46 quentin quentin  4096 oct.  12 08:57 .
drwxr-xr-x  3 root    root     4096 févr.  9  2014 ..
-rw-r--r--  1 quentin quentin 12288 sept. 18 16:36 .aa.aux.swp
drwx------  3 quentin quentin  4096 févr. 14  2014 .adobe
-rw-------  1 quentin quentin 33826 oct.  12 00:14 .bash_history
-rw-r--r--  1 quentin quentin   220 févr.  9  2014 .bash_logout
-rwxrwxrwx  1 quentin quentin  3840 oct.   5 16:23 .bashrc
-rwxrwxrwx  1 quentin quentin  3599 sept. 25 18:23 .bashrc~
drwxr-xr-x  5 quentin quentin  4096 sept. 27 14:01 Bureau
drwxrwxr-x  5 quentin quentin  4096 sept. 25 16:41 C
drwxrwxr-x  8 quentin quentin  4096 sept. 27 17:57 C++
drwx------ 26 quentin quentin  4096 oct.  12 08:50 .cache
drwx------ 31 quentin quentin  4096 oct.  11 22:11 .config
drwxrwxr-x  4 quentin quentin  4096 oct.  11 19:05 .core
drwx------  3 quentin quentin  4096 févr.  9  2014 .dbus
-rw-r--r--  1 quentin quentin    41 sept. 17 19:29 .dmrc
drwxr-xr-x 12 quentin quentin  4096 oct.  12 08:57 Documents
drwxrwxr-x  2 quentin quentin  4096 oct.  12 08:57 Downloads
drwx------  3 quentin quentin  4096 févr. 14  2014 .emacs.d
drwxr-xr-x  2 quentin quentin  4096 sept. 17 15:04 .fontconfig
drwx------  4 quentin quentin  4096 oct.  12 00:16 .gconf
drwx------  4 quentin quentin  4096 mars  22  2014 .gegl-0.0
drwxr-xr-x 22 quentin quentin  4096 août  26 21:20 .gimp-2.6
drwxr-xr-x 24 quentin quentin  4096 oct.  12 08:53 .gimp-2.8
-rw-r-----  1 quentin quentin     0 sept. 17 16:03 .gksu.lock
drwx------  4 quentin quentin  4096 oct.  11 18:08 .gnome2
-rw-------  1 quentin quentin     0 févr. 28  2014 .goutputstream-38Y6BX
-rw-------  1 quentin quentin     0 févr. 21  2014 .goutputstream-3TLVBX
-rw-------  1 quentin quentin     0 avril 15 00:27 .goutputstream-5BY2DX
-rw-------  1 quentin quentin     0 juin  21 10:22 .goutputstream-6JPXHX
-rw-------  1 quentin quentin     0 avril 16 17:48 .goutputstream-7QJCEX
-rw-------  1 quentin quentin     0 févr. 28  2014 .goutputstream-BNSDBX
-rw-------  1 quentin quentin     0 mars   2  2014 .goutputstream-EXXZBX
```

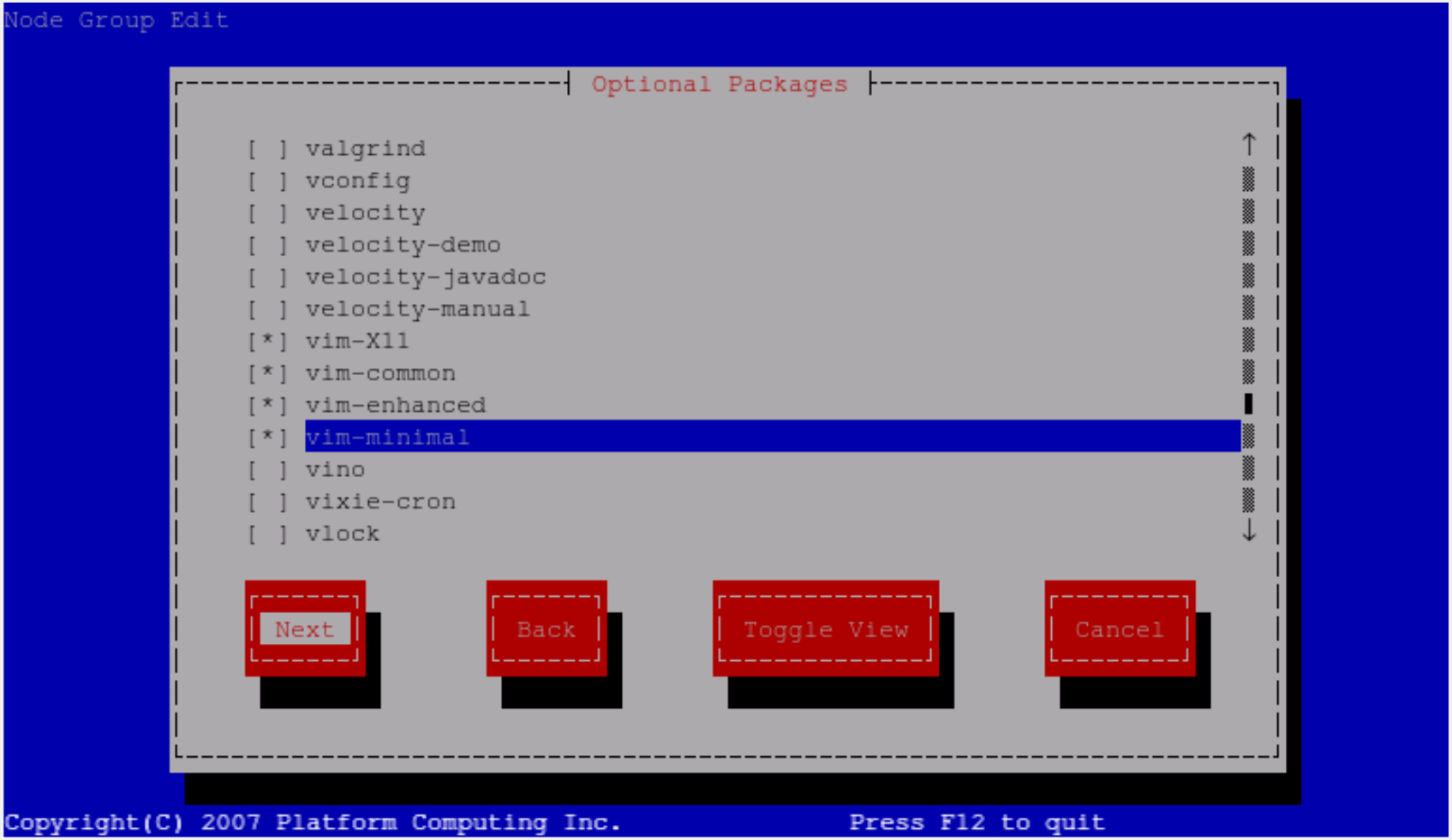
TUI : Une avancée visuelle

Text User Interface

Ce n'est pas encore une GUI, mais elle possède néanmoins des fenêtres, est capable de générer des pop-up et possède un système rudimentaire de menu déroulant. Les interactions se font avec le clavier et la souris.

Aujourd'hui : Menu BIOS, par exemple.

Text User Interface



Sommaire

Mais qu'est-ce qu'une GUI ?

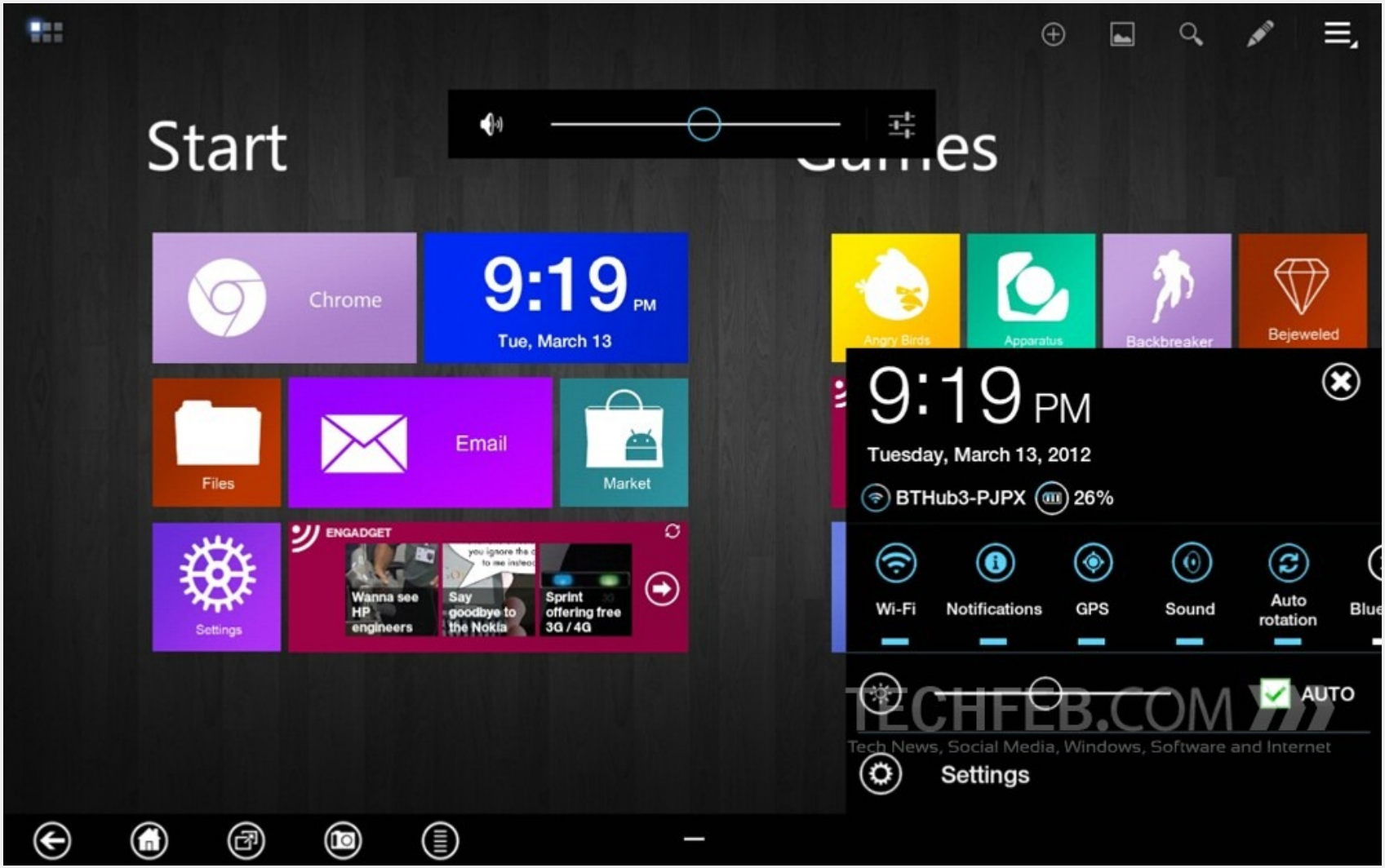
Graphical User Interface

Les fenêtres telles que nous les connaissons. Les objets sont affichés sous forme de pictogramme.

- Interface la plus **user-friendly**: objectif rendre l'interaction avec le système transparente.
- L'écriture d'une commande complexe est remplacée par un simple clic ou la pression d'une touche.

Aujourd'hui : La majorité des interfaces sont des GUI !

Graphical User Interface



Sommaire

Sommaire

Qu'est-ce que Qt ?

- C'est un **Framework** : un ensemble de bibliothèques (plus de 13Go !)
- Projet débuté en 1991 par Trolltech, repris aujourd'hui par Qt-Project.
- Permet de créer des programmes sous licence LGPL ou propriétaire.
- Surcouche du C++ (livré avec un IDE et un gestionnaire de surcouche).
- Multi-plateformes.

Sommaire

Les Widgets

Qu'est-ce qu'un Widget? (Window Gadget)

- C'est la brique de nos fenêtres Qt.
- En pratique, tout type de widget est une spécialisation de *QWidget* :
  - Les layouts (*QLayout*)
  - Les champs d'édition de texte (*QLineEdit*)
  - Les boutons (*QButton*)
  - Et bien d'autres encore...

Les Layouts

L'agencement des widgets

Comment organiser sa fenêtre ?

Les Layouts

Il s'agit d'un widget dédié à l'organisation spatiale : c'est un **conteneur** qui ordonne son contenu suivant ses spécificités, et de manière transparente pour l'utilisateur.

On retrouve notamment:

- les layouts horizontaux
- les layouts verticaux
- les layouts de grille
- les layouts de formulaire
- les layouts piles

Les Layouts

L'emboîtement

Comment utiliser les layouts ?

Propriété des Layouts

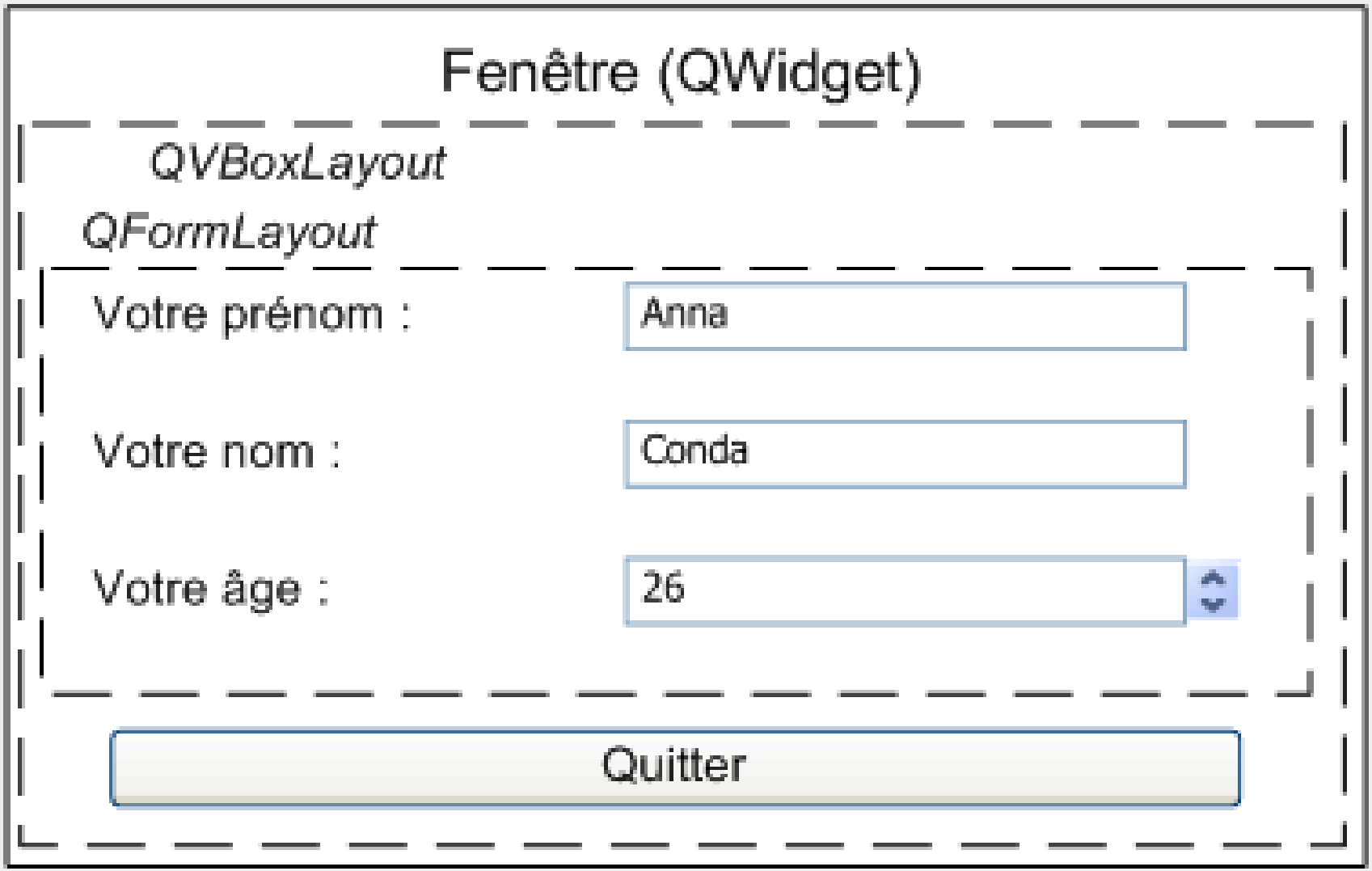
Un layout peut contenir n'importe quel type de widget.
Il peut donc contenir d'autres layouts.

Principe de l'emboîtement

Un layout ne peut gérer qu'une mise en forme à la fois, il suffit alors d'imbriquer plusieurs layouts pour parvenir à une mise en forme plus complexe.

Les Layouts

Illustration



Interagir avec sa fenêtre

Les boutons

Comment créer le lien entre l'interface et notre programme ?

Les types de boutons

Il existe plusieurs types de boutons fourni par Qt, chacun répondant a un besoin spécifique :

- Les boutons cliquables
- Les cases à cocher
- Les boutons de barre d'outils
- Et d'autres encore...

Interagir avec sa fenêtre

Les champs de texte

Comment afficher/recevoir des informations ?

Les champs d'affichage

Il existe plusieurs champs d'affichages :

- Les champs de texte
- Les champs de lecture
- Les champs d'images

Les champs d'écriture

On n'utilise qu'un seul widget en général : *QLineEdit*.
Ses propriétés le rendent très souple : on peut ainsi masquer les caractères entrés, modifier la taille du champ, limiter la longueur du texte à écrire...

Interagir avec sa fenêtre

Pour aller plus loin

Lorsque que notre GUI commence à se complexifier, on peut avoir besoin d'avoir besoin de plus d'options/fonctionnalités.

Les menus

Contrairement aux widgets vus jusqu'à présent, les menus ne sont pas pré-implémentés : c'est à l'utilisateur de spécifier les noms de menus, les options proposées, leurs actions, et autres propriétés (raccourcis clavier, par exemple.)

La personnalisation

Qt reposant sur la POO, il est ainsi possible de créer ses propres widgets héritant de widgets préexistants.

Sommaire

Interagir avec la GUI

Quel moyen ?

- Comment communiquer avec notre interface ?

On utilise des événements

Ce sont des "messages" envoyés par un périphérique (ou le système d'exploitation) à l'application.
Quelques exemples d'événements :

- L'utilisateur presse une touche du clavier.
- Il bouge la souris.
- Il réduit la fenêtre.

Remarque : Qt nomme ses événements *QEvent*.

Interagir avec la GUI

Des événements, et ensuite ?

- Comment intercepter/traiter les événements reçus ?

On utilise une boucle principale (ou mainloop)

- Tout d'abord, le système place les événements dans une file (transparent pour le programmeur).
- Puis on récupère l'événement en tête de file, le traite, et réitère jusqu'à ce qu'un événement de fin mette un terme à la boucle.

Exemple (1)

mainLoop

mainLoop() :

```
Faire evenementCourant := recupererEvenement()
|   si evenementCourant == evenement1
|   |    alors traitement prévu pour evenement1
|   si evenementCourant == evenement2
|   |   alors traitement prévu pour evenement2
|
|   ... On étudie chaque cas de figure ...
|
tant que evenementCourant != evenementFin
```

Exemple (2)

code programme

Le code de notre programme devient :

Programme :

```
definirGUI() //Structuration avec widgets
mainLoop()
```

Constat

La programmation événementielle dans la boucle principale est très lourde (il faut prévoir chaque cas de manière exhaustive, et déterminer un traitement spécifique.)

Programmation Événementielle

Les deux mécanismes

Qt propose deux mécanismes parallèles pour traiter les événements :

- le traitement des *QEvent*.
- Un mécanisme de signaux et de slots.

Les signaux et les slots

Les signaux

Les signaux sont simplement des messages qui peuvent être connectés à des actions que nous voulons exécuter.

les slots

Les slots sont des fonctions (ou méthodes) qui exécutent un traitement.
Les slots sont connectés à des signaux qui les appellent.

Quand la POO s'en mêle

Comportement des objets

Quel intérêt ?

Plusieurs objets peuvent être soumis à des événements similaires, par exemple :

- Les objets de types bouton (*QPushButton*) peuvent être "cliqués".
- Les fenêtres (*QWindow*) redimensionnées, leur visibilité changée, etc.

On peut alors généraliser en ajoutant des signaux (événements) et des slots (traitements) à nos déclarations de classes.

Schéma I : graphiquement

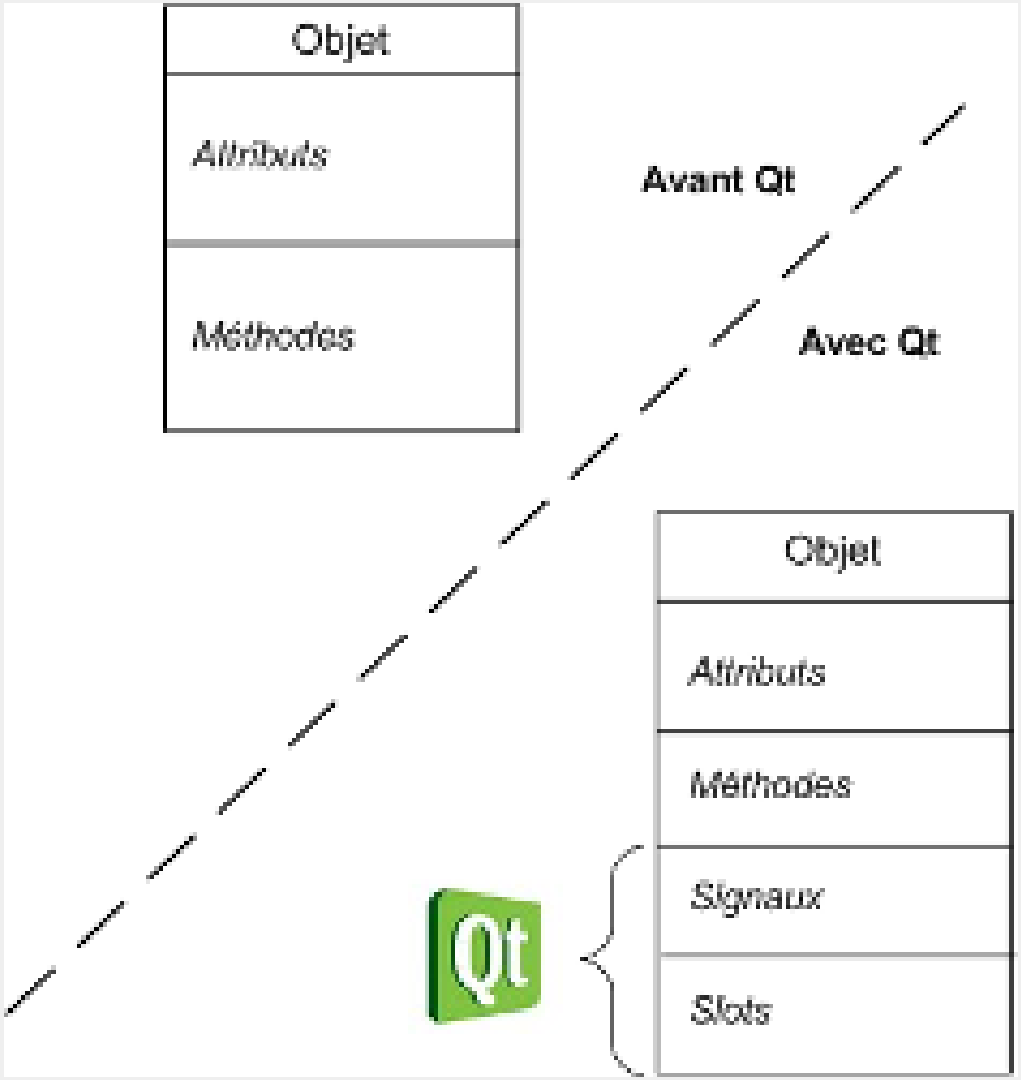


Schéma II : pseudo-code

```
public class MaClasse{

  /* Déclaration attributs */
  /* Déclaration méthodes */

  public signals:
      void monSignal();
      void monSignal(Type monParam);

  public slots:
     void monTraitement();
     void monTraitement(Type monParam);
};
```

Schéma II : pseudo-code (suite)

Puis on connecte les signaux aux slots (généralement dans le constructeur) :

```
MaClasse(){
    ...
    QObject::connect(emeteur, SIGNAL(monSignal()),
receveur /* ici this */, SLOT(monTraitement()));

    QObject::connect(emeteur, SIGNAL(monSignal(Type)),
receveur /* ici this */, SLOT(monTraitement(Type)));
}
```

Principe général vs système signaux-slots

- La boucle événementielle est appelée par *QApplication::exec()*
- Les Signaux et les Slots masquent l'existence des *QEvent* (c'est un mécanisme de plus haut niveau).
- La récupération des messages, leur transformation en signal ou en *QEvent* est transparent pour l'utilisateur.
- Les Signaux et les Slots permettent de définir facilement le comportement de nos Widgets en fonction des actions de l'utilisateur.

Démonstration

Démonstration !

C'est à vous !

Des questions ?
Lancez-vous !