LE FRAMEWORK QT

Présentation du framework

Exposé Ingénieurs2000 par Sébastien MOURET
Janvier 2009

TABLE DES MATIÈRES

INTRODUCTION

Dans le cadre de mes études au sein de l'école Ingénieurs2000 dont les cours sont dispensés à l'université Marne-La-Vallée, il nous est demandé de choisir un sujet en Informatique et de faire une présentation sur celui-ci. Cette présentation est composée d'un site web accompagnée d'une présentation orale devant l'ensemble de la promotion.

J'ai décidé dans le cadre de cette présentation de choisir de présenter le framework Qt qui pour moi va permettre de remplacer à terme le Java. Qt permet d'enrichir les fonctionnalités du langage de programmation C++ et permet de développer des application multi-formes sur les principaux systèmes d'exploitation. Ce framework a atteint un niveau de maturité depuis 1994 date où il a été créé afin de fournir un ensemble de fonctionnalités et des outils facilitant grandement le développement d'applications.

Je vais au cours de cette présentation vous présenter rapidement l'histoire du framework et de sa société de création. Puis dans une seconde partie, je vais vous présenter le framework et ses qualités par rapport au simple langage de programmation C++. Enfin je ferai un tour succinct sur les outils fournis dans le framework afin de faciliter la vie du développer. Dans la dernière partie je fournirai une vidéo de présentation d'un mini-projet Qt afin de montrer la puissance de l'IDE.

La denière page fournit l'ensemble des sources que j'ai utilisées pour réaliser cette présentation afin de vous permettre d'approfondir l'apprentissage du framework.

GÉNÉRALITÉS

Qt Software

Qt Software est une société Norvégienne anciennement connue sous les noms de Quasar Technologies et Trolltech. Elle fût créée par les Norvégien Eirik Eng et Haavard Nord en 1994.

Depuis le 28 Janvier 2008 elle appartient à la société de fabrication de téléphones mobiles Nokia qui a investi dans Trolltech dans le but de concurrencer Android et l'Iphone OS dans la course des OS embarqués.

Le nom du framework Qt provient de l'affichage de la lettre sous Emacs qui plaisait aux créateurs et de la lettre T pour X-Toolkit.

La principale activité de la société Trolltech est la vente et le support sur deux framework, Qt et Qt-Extended. Nous allons voir au cours de cette présentation la différence entre ces deux framework.

Historique

Le framework Qt a vu le jour sur sa première version stable 0.90 en 1995. Cette dernière a permis la création des premières applications graphiques sous Windows et Linux.

Sortie ensuite la version 1.2 qui sera à la base du projet KDE dont le leader était Matthias Ettrich. Ce dernier choisi ce framework pour ces qualités de développement d'application graphiques riches et sa capacité à être adapté sur les environnement Windows et Linux.

Tout d'abord, le framework Qt était vendu sous deux types de licences distinctes, une première licence commerciale et une deuxième Open Source. La deuxième implique que les applications développées ne soient distribuées que sous la même licence. C'est d'ailleurs à cause de cette petite subtilités que naquit le projet Gnome. La licence commercial a disparu depuis la version 4.6 de ce même framework.

En l'an 2000 est sorti le framework Qt-Extended qui permet de développer des application dans les terminaux embarqués. A l'époque le nom utilisé était Qtopia, c'est en quelques sortes un système d'exploitation pour les appareils mobiles.

En 2001 est sortie une des versions un plus complète du framework Qt la version 3, cette dernière apporta le support de développement d'application sur le système d'exploitation à la mode Mac Os. C'est à ce moment que nous pouvions parler de framework graphique multiplateforme. En effet, Qt permet le développement d'applications graphiques dans un premier

temps. Ce n'est qu'à partir de la version 4 qui sortie en 2005 que nous pouvons parler de framework complet permettant de développer des applications et des jeux sous les trois plus grands systèmes d'exploitation.

La version 4 a apporté un ensemble de bibliothèques supportant le développement d'applications réseau, OpenGL, permettant ainsi le développement d'applications complètes.

La dernière version à ce jour (Janvier 2009) est la 4.6, version rachetée par Nokia.

LE FRAMEWORK QT

En quelques mots

Dans cette partie nous allons explicités les différents avantages et inconvénients du framework Qt, ces éléments seront expliqués dans la suite de ce même document.

Tout d'abord le framework Qt possède un ensemble de bibliothèques permettant de développer des application qui requièrent le support XML, réseau, manipulation de bases de données ou encore il offre la possibilités de développer des jeux grâce au support de l'OpenGL.

L'équipe Trolltech a mis à la disposition du développer un ensemble d'outils facilitant le développement d'applications. Il faut savoir que comme le framework fonctionne sur les trois systèmes d'exploitation, ces outils sont eux aussi compatibles avec l'ensemble des systèmes. Bien entendu leur version ne sont pas aussi stables sur l'une ou l'autre des plate-forme.

Le builder d'interface Qt Designer offre la possibilité de dessiner ses interfaces graphiques à l'aide de simples glisser déplacer.

Il possède de plus un IDE à l'image de ce qu'est Eclipse pour le langage de programmation Java, le framework Qt possède Qt Creator dont le fonctionnement sera présenté dans la suite du document.

Le framework Qt facilité l'internationalisation des applications développées grâce à l'outil Qt Linguist.

Tout comme Java et sa JavaDoc, Trolltech met à la disposition des développer une documentation riche et complète qui peut être visualiser sur leur site ou bien par l'intermédiaire de l'outil Qt Assistant qui permet d'avoir l'intégralité de la documentation en local.

De plus, Trolltech tout comme Sun fournit le code source des ses bibliothèques à l'exception de la bibliothèque XML.

Il faut savoir que le framework Qt permet de développer des applications multi-plates-formes en C++ (framework Qt) mais il existe aussi des binding permettant de développer dans des langes tels que Java (QtJambi), Python (PyQt) ou encore en Perl (PerlQt). Les applications développées dans ces langages pourront être exécutées sous Windows, Linux et Mac OS.

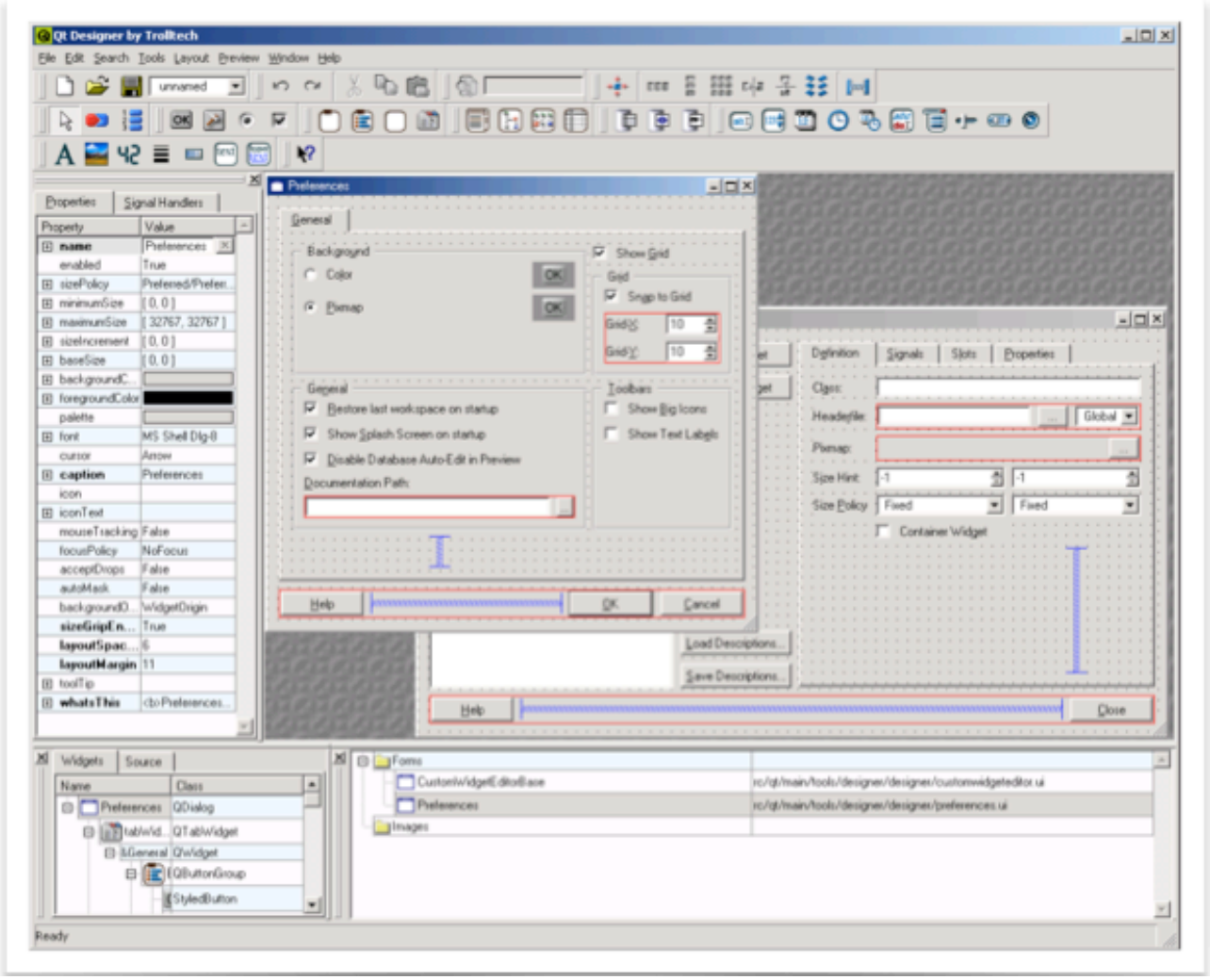
Toutes les applications développées par l'intermédiaire de Qt possèdent le look and feel de chaque système d'exploitation.

Qt est utilisé par les plus grandes entreprise du monde de l'informatique tels que Adobe avec Adobe Photoshop Ambum ou Google avec Google Earth mais encore KDE ou Skype ou encore l'excellentissime VLC.

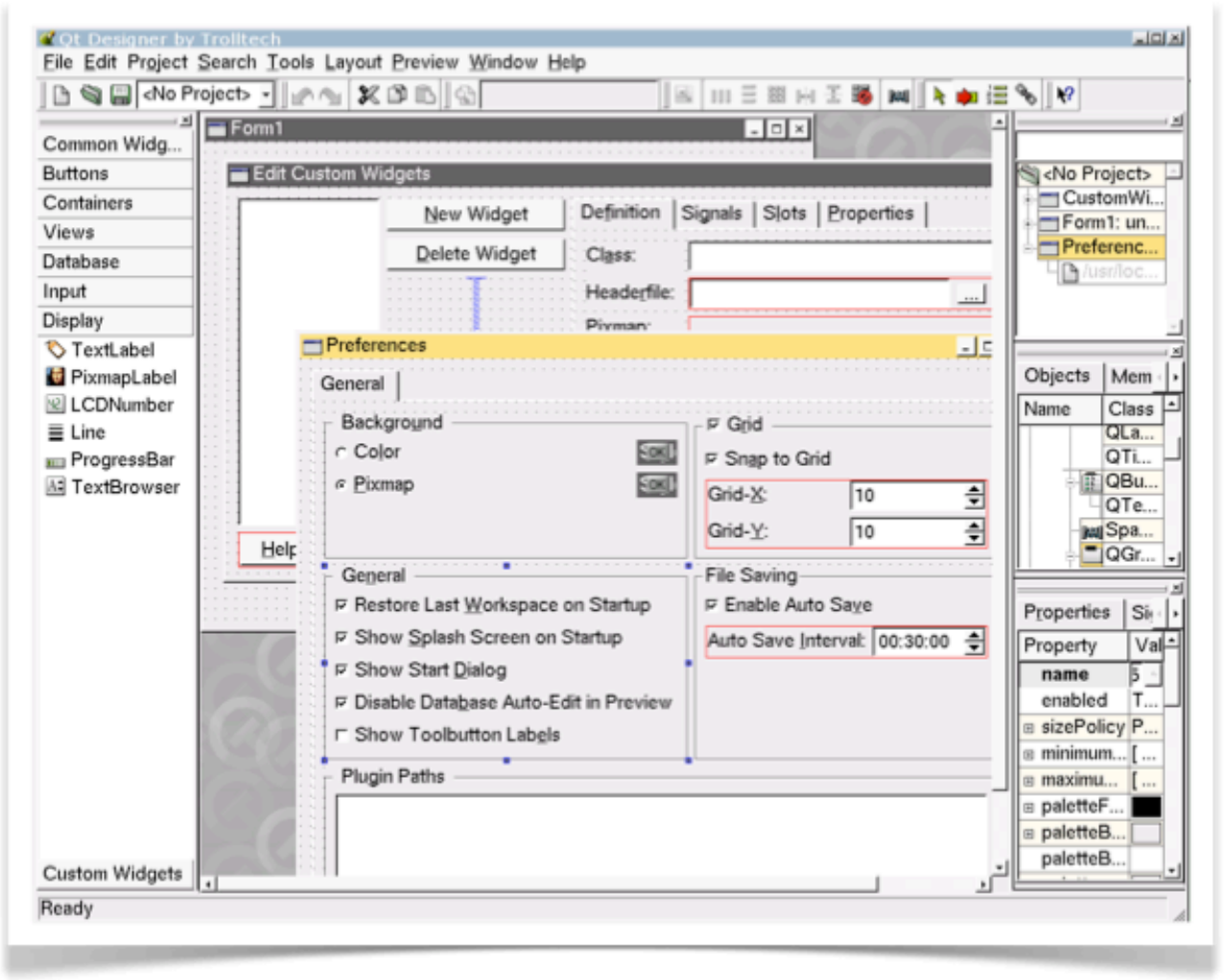
Look&Feel

Par l'intermédiaire des captures suivantes nous pouvons nous rendre compte de l'intégration des applications développées par l'intermédiaire du framework Qt.

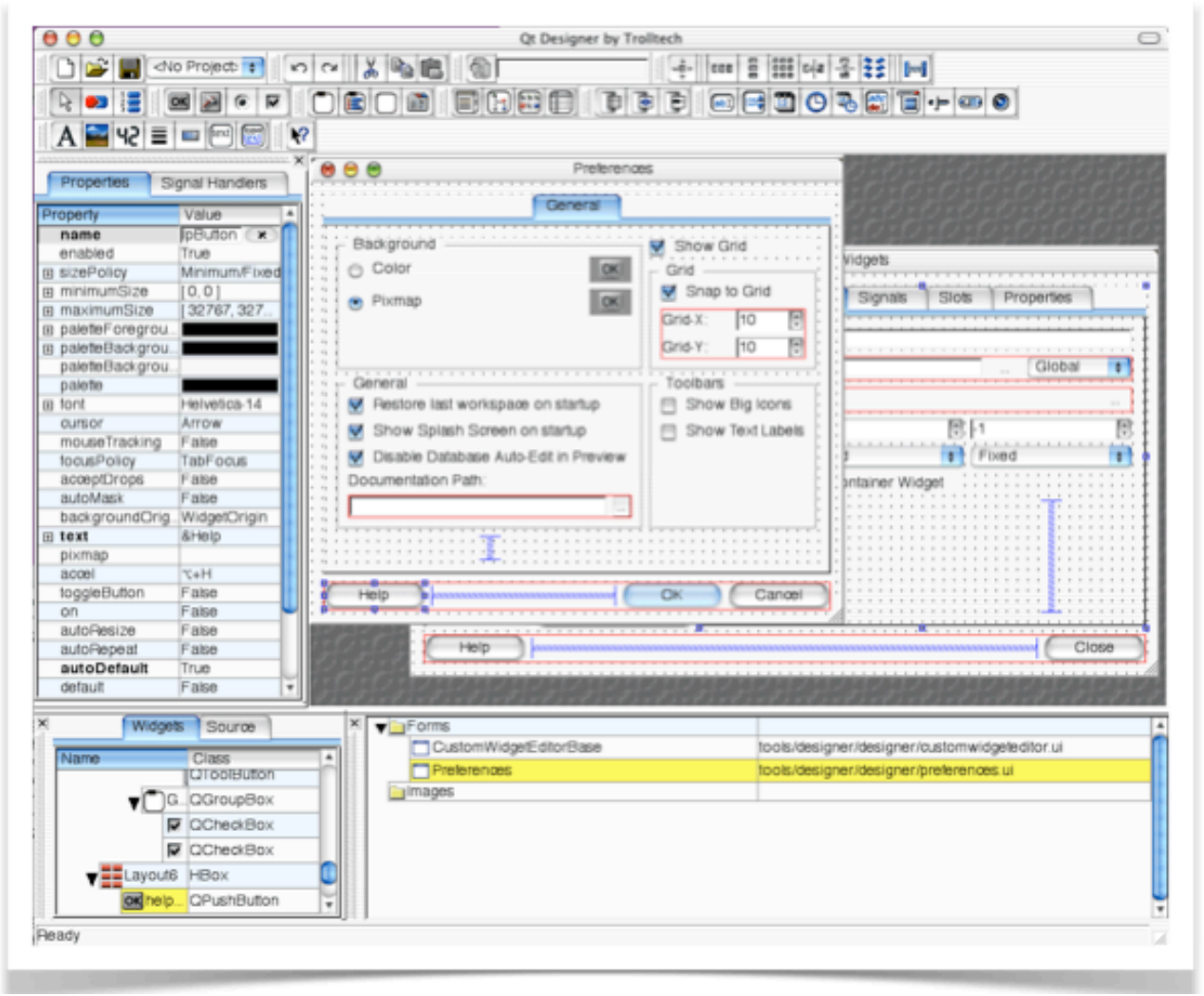
Application sous Windows :



Application sous Mac OS :

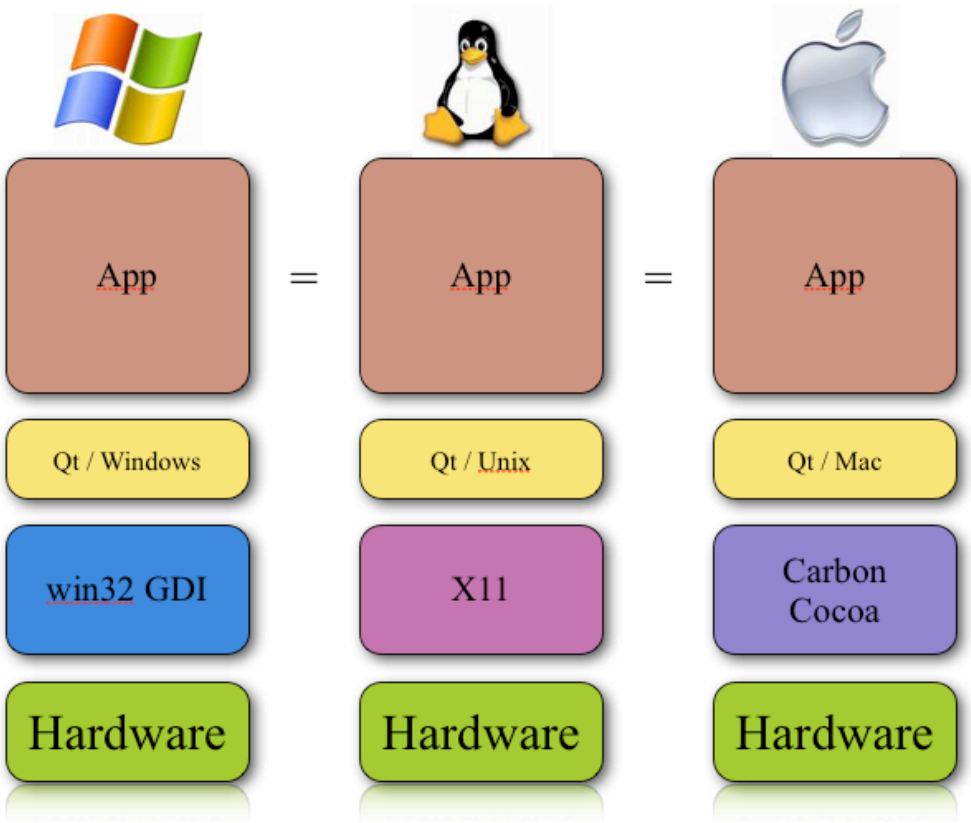


Application sous Linux :



Architecture

Le framework Qt est conçu de telle sorte que les applications développées soient compatibles avec les systèmes d'exploitations suivants Windows, Linux, Mac OS. Le Framework est basé sur la couche graphique des trois OS, win32 GDI pour Windows, X11 pour Linux ou encore Carbon / Cocoa pour Mac OS.



L'utilisation des outils de développement standards sont compatibles avec Qt (IDE, debugger...), le développement est transparent pour un développer C++. Qt est indépendant du système et ne demande qu'une simple recompilation pour pouvoir être adapté.

L'API Qt est la même sur tous les systèmes. Les applications sont compilées en exécutables natifs pour le système d'où le slogan «Write Once, Compile Everywhere».

Mieux que C++

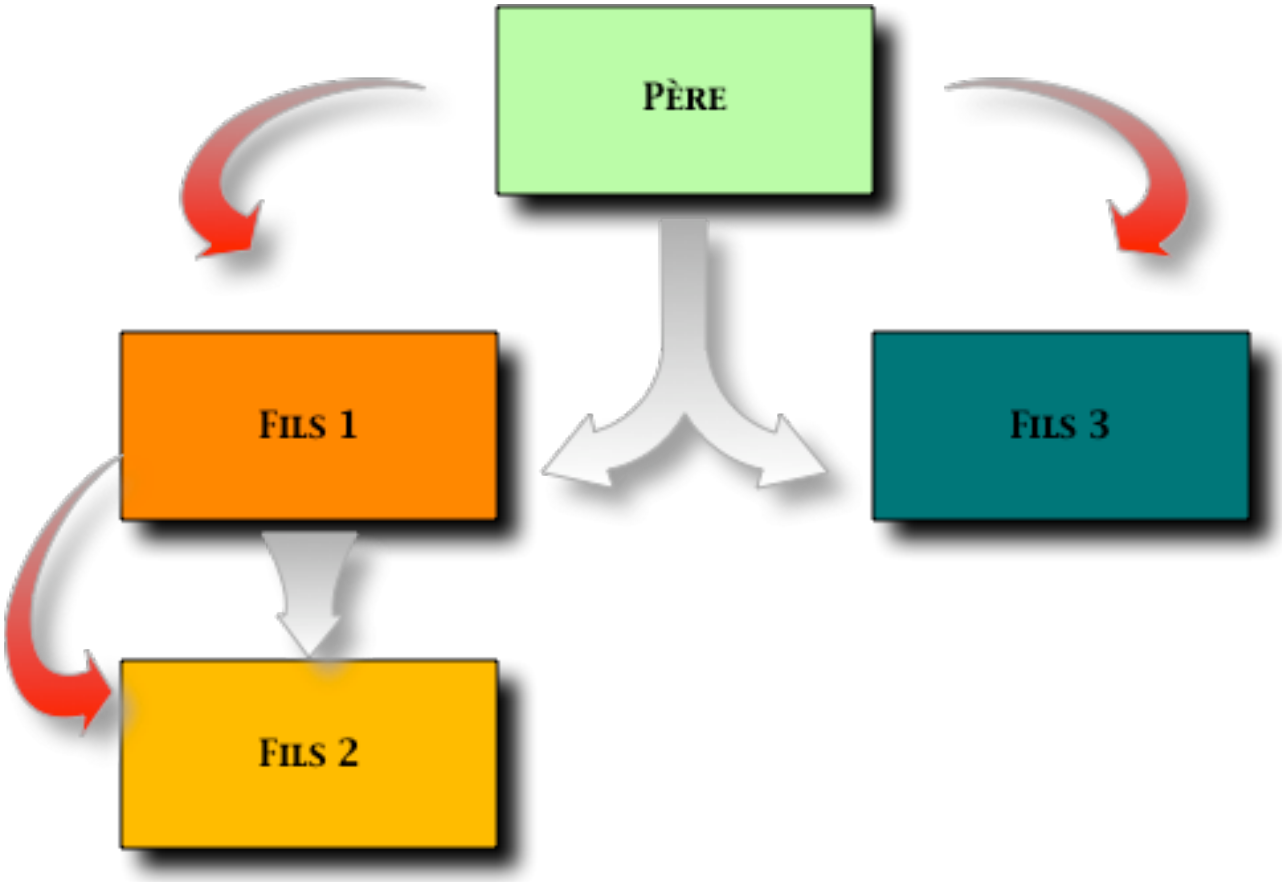
Qt étend les capacités importantes de C++, pour développer avec Qt il faut avoir les connaissances requises pour développer en C++. Cependant, Qt apporte des amélioration au C++, d'abord un système de signaux et de slots qui à l'image des Listeners en Java permettent de faire communiquer des objets entre eux sans les couplés.

La bibliothèque Qt est aussi important que celle proposée pour Java, par exemple il fournit le nécessaire pour manipuler de façon plus simple qu'en C++ les chaines de caractères avec les QString, ou encore la gestion des listes avec les QList.

Qt améliore la gestion de la mémoire contrairement à C++, il faut partir du principe que tous les objets héritent de QObject comme les objets Java héritent de Object. Nous allons voir dans la partie suivante ce qu'implique cet héritage.

La gestion de la mémoire

Il n'y a pas de garbage collector en Qt contrairement à Java. Il faut faire hériter tous les objets Qt de Qobject ainsi ils gardent une référence sur un objet père. Ce mécanisme permet de donstruire une linked liste puisque tous les objets sont chainees entre eux ainsi dès la destruction du père tous les objets fils sont libérés de la mémoire.



Dans la captures ci-dessus, lors de la construction du fils 1 et 3 nous leur donnons un pointeur sur leur père, alors que le fils 2 n'a qu'une référence sur le fils 1. Lors de la destruction du père à l'aide de la commande «delete», le fils 1 va être supprimé ce qui va supprimé le fils 2, enfin le fils 3 sera supprimer en dernier.

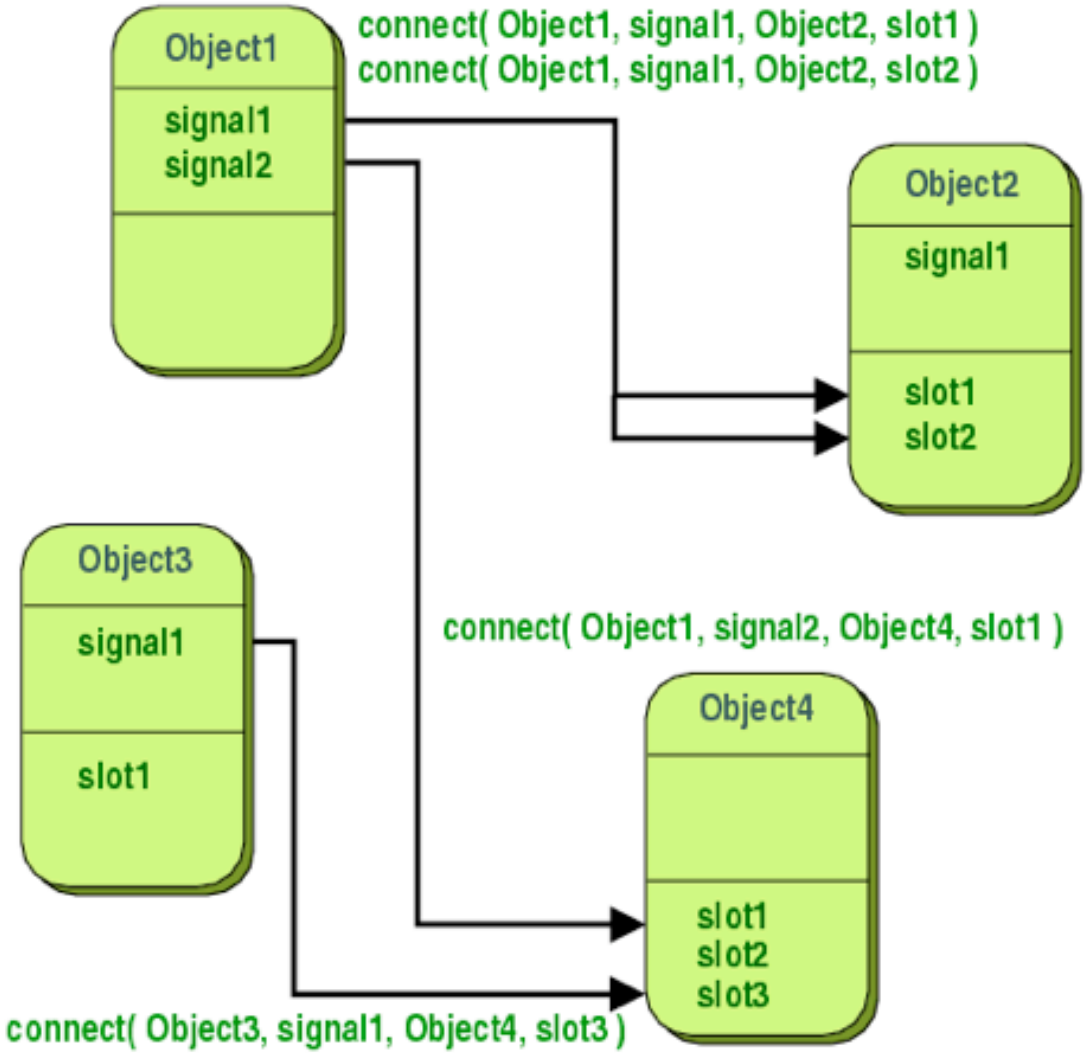
Les signaux et les slots

Les signaux et les slots sont aux C++ ce que les listeners sont au Java. Ils permettent de faire communiquer les objets entre eux sans les liés ce qui permet une réutilisation du code par la suite. En effet, un objet ne connaît pas quel objet l'appel mais il ne connait pas non plus l'objet qu'il appelle.

Un signal est une fonction émise par un objet (fonction «emit»), ce signal est associé à une fonction d'un autre objet (ou le même) ces fonctions sont appelées des slots. Nous pouvons

connecter un signal à un autre et créé des cascades d'appels, on peut supprimer un signale (fonction «disconnect»). La connexion d'un signal à un slot se fait à l'aide de la commande «connect».

Dans la capture suivante nous allons voir les différents scénarios possibles.



L'objet 1 a connecté ses signaux 1 et 2 aux slots des objets 2 et 4. Ainsi lorsque le signal 1 de l'objet 1 sera émis, le slot 1 et 2 de l'objet 2 sera appelé, ceci revient à exécuter les fonctions 1 et 2 de l'objet 2. Lorsque que le signal 2 de l'objet 1 sera émis le slot 1 de l'objet 4 sera appelée. Enfin lorsque le signal 1 de l'objet 3 sera sera émis le slot 3 sera appelé.

Remarque : nous pouvons constaté qu'un slot n'a pas besoin d'être connecté à un signal tout comme un signal n'a pas besoin d'être connecté à un slot pour exister.

Dans la capture ci-dessous nous allons voir comment connecter les slots et les signaux entre eux. Nous faisons le lien entre le listener en Java et les signaux en C++.

```
//JAVA
button.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
           exit();
        }
});

//QT Jambi
bouton.clicked.connect(QApplication.instance(), "quit");

//QT
QObject::connect(bouton, SIGNAL(clicked()), &app,
SLOT(quit()));
```

En Java nous ajoutons un listener au bouton pour que dès qu'il émet le signal «cliqué» la fonction exit soit appelée. Avec QtJambi cela revient à ajouter un objet connect à la liste clicked afin d'appliquer la fonction quit.

Avec Qt nous connectons le le signal «clicked» de l'objet bouton au slot «quit» de l'objet «app».

Ce mécanisme de signal comme nous pouvons le constater est très simple et très efficace cependant, nous n'avons pas vu comment passer des paramètres à la fonction appelée puisqu'elle n'en prenait pas. Naturellement nous serons tentés d'écrire ceci :

```
connect( &a, SIGNAL(ageChange(40)),
&b, SLOT(majAgePersonne(int)) );
```

Qt n'accepte pas que dans un connect nous passions les arguments donc cette façon n'est pas la bonne. Voici la bonne manièere de procéder :

```
connect( &a, SIGNAL(ageChange(int)),
&b, SLOT(majAgePersonne(int)) );


emit ageChange(40);
```

Il faut dans la fonction «connect» déclarer les prototypes des fonctions. Ici nous spécifions que la fonction «agechange» prend un entier comme argument. Lors de l'émission de la fonction «agechange» nous spécifions le nombre à envoyer au slot «majAgePersonne».

De plus, pour permettre l'utilisation des slots et des signaux, il faut dans les fichiers d'en-têtes ajouter l'instruction «Q_OBJECT» et les instructions «slots» et «signals». Voici un exemple de déclaration :

```
class Personne : public QObject
{
Q_OBJECT
public:
   Personne(QObject *parent=0, char *name=0 );
   int agePersonne() const { return age; }
public slots:
   void majAgePersonne( int );
signals:
   void ageChange( int );
private:
   int age;
};
```

La compilation

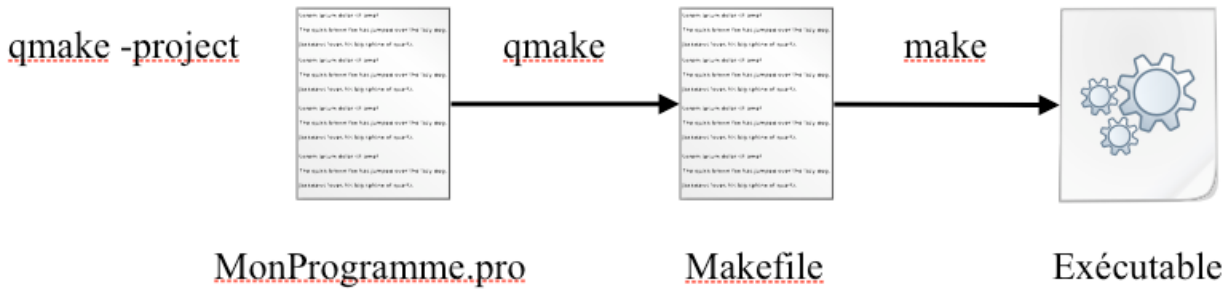
La compilation est la phase la plus importante dans un projet Qt, c'est à ce moment précis que l'application est compilée pour le système. La compilation d'un projet Qt diffère d'un projet C++ dans le sens où elle se fait en trois étape.

La première étape consiste à créer un fichier de projet Qt à l'aide de la commande «qmake -project». Lors de l'exécution de cette commande Qt balaye de façon récursive le dossier du projet afin de créer un fichier «.pro» qui sera utile pour l'étape de compilation suivante.

La deuxième étape de la compilation consiste à compiler le fichier «.pro» créé à l'étape précédente à l'aide de la commande «qmake», cette étape va créer un Makefile traditionnel qui pourra être lancé de la façon habituelle.

La dernière étape consiste à compiler le projet de façon classique (C++) à l'aide du Makefile obtenu à l'étape précédente. L'exécutable obtenu à cette étape est exécutable sur le type de système d'exploitation sur lequel le projet a été compilé.

Si toute fois, l'utilisateur veut utiliser le projet sur un autre système d'exploitation, il lui faudra le compiler sur le système à l'aide des fichiers sources du projet.



Il est dans certains cas préférable de renseigner le fichier «.pro» afin de spécifier soi-même les fichiers à inclure dans le projet. Ce fichier a la forme suivante :

```
INCLUDEPATH += .
TARGET = HelloWorld
QT = core gui
TEMPLATE = app
SOURCES += main.cpp
SOURCES += myhelloworld.cpp
HEADERS += myhelloworld.h
FORMS += myhelloworld.ui
RESOURCES += RandHelloWorld.qrc
TRANSLATIONS += traduction_fr.ts
```

TEMPLATE = spécifie si on souhaite compiler une appliccation (app) ou une librairie (lib)

TRANSLATIONS: fichiers de langues

TARGET: nom de l'exécutable

INCLUDEPATH: répertoire des fichiers sources et des en-têtes

SOURCES: répertoire des fichiers sources

HEADERS: répertoire des en-têtes

FORMS ou INTERFACES: fichier de l'interface graphique obtenus à l'aide du builder graphique Qt Designer

Qt-Extended

Qt Extended (anciennement Qtopia) est une plate-forme libre développée par Qt Software, basée sur la bibliothèque Qt pour les systèmes embarqués équipés du noyau Linux.

LES OUTILS

Qt Script

Qt Scrippt est un outils fournit par le framework Qt qui donne la possibilité au développeur de fournir un moteur de script dans leur application. Cette capacité permet aux administrateur système utilisant des applications Qt d'étendre les possibilités de celles-ci sans avoir à recompiler les sources ni à toucher au code source. Le langage de script est basé sur le standard EXMAScript connu notamment pour sa syntaxe proche du Javascript.

Dans notre application Qt on va tout d'abord créer un QScriptEngine qui est le moteur permettant d'interpréter le script. Le script entré par l'utilisateur sous forme de QString va être évalué par ce moteur. Voici un exemple d'utilisation :

```
QScriptEngine engine;
QScriptValue result = engine.evaluate(userCode);
```

Qt Designer

Qt Designer est un builder d'interfaces graphiques qui fournit au développeur la capacité de créer ses propres interfaces à l'aide de glisser-déplacer des composants graphiques tels que des bouton des textfiled etc.

Ce même logiciel va ensuite générer un fichier «.ui» basé sur la norme XML. Ce fichier est utilisé par le compilateur Qt lors de la compilation du fichier «.pro». De plus, la génération des fichiers sources et des fichiers d'en-têtes est automatisée par Qt Designer à l'aide du compilateur intégré User Interface Compiler.

Contrairement à Netbeans et sa génération de code lors de la création d'interfaces graphiques à l'aide de ce dernier, la génération du code source à l'aide de Qt Designer est propre. Les fichiers sources sont réutilisables très facilement.

Qt Designer simplifie grandement la construction d'interfaces graphiques même des plus complexes.

Qt Linguist

Qt Linguist est un outils fourni avec le framework Qt qui permet de simplifier la vie des développer afin de traduire l'interface de leur application. Ce logiciel va scanner les fichiers sources

à la recherche des QString dans lequel le développer à mis une chaîne de caractères précédée de «tr». Ainsi toutes les chaînes de caractères trouvées seront mises dans un fichiers «.ts».

Qt Linguist permet d'ouvrir les fichiers «.ts» et donne la possibilité de traduire ses fichiers dans la langue de son choix. Par exemple si le développer à généré un fichier mon_app_francais.ts qui contient les mots tels que bonjour, il pourra traduire ce mot en hello et générer le fichier mon_app_anglais.ts pour sa traduction en anglais.

Une fois les fichier «.ts» créés il faut générer les fichiers «.qm» qui sont utilisable par l'application Qt afin de traduire l'interface. Ces fichier s'obtiennent en convertissant les fichiers «.ts» à l'aide de Qt Linguist.

Voici un exemple d'utilisation de la fonction tr :

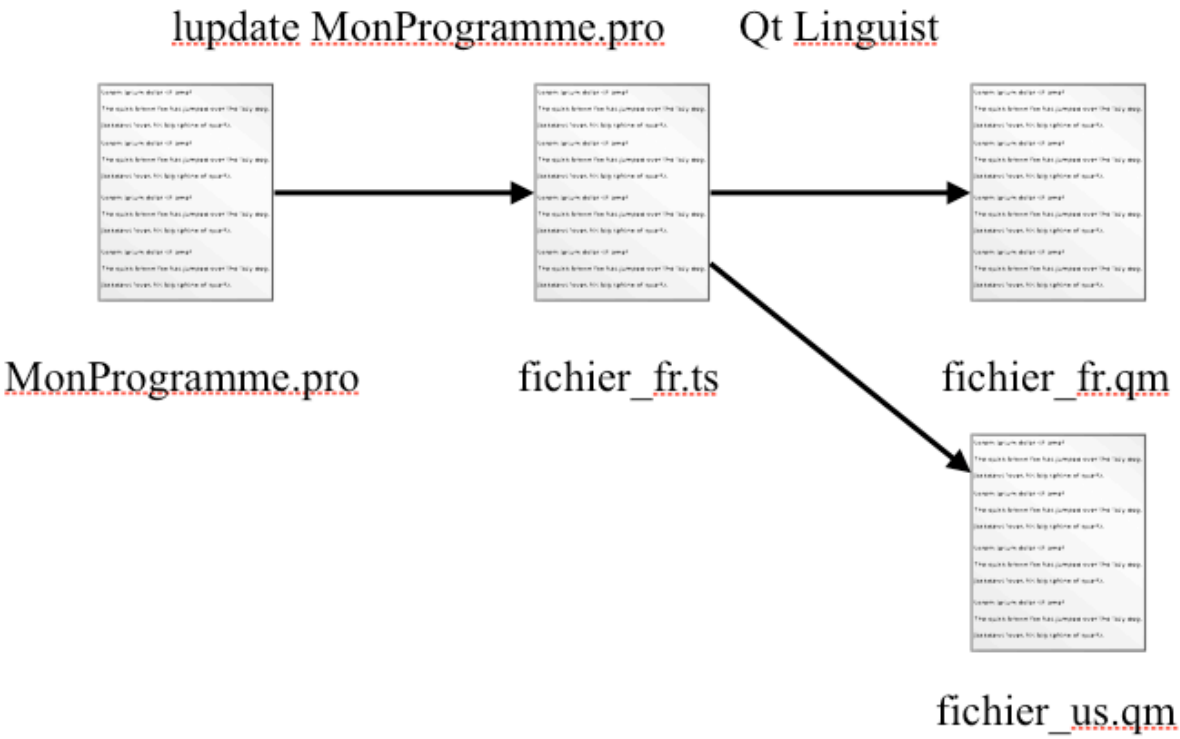
```
QPushButton(tr("mon texte", "commentaire"));

QPushButton(tr("%n nouveau(x) message(s)", "message arrivé", messages.count()));
```

La chaîne «mon texte» sera suivi du commentaire commentaire qui ne sera visible qu'à l'intérieur de l'application Qt Linguist afin de guider la personne en charge de la traduction du logiciel.

Nous pouvons voir dans la seconde ligne qu'il est possible de mettre des données tels que des nombres à l'intérieur des chaîne à traduire.

Voici les différentes étapes de la création des fichiers de langue pour des applications Qt :



Qt Creator

Qt Creator est un outils fourni avec le framework Qt, il est au C++ ce qu'Eclipse est au Java. C'est un IDE complet qui permet de coder des application Qt. Il fournit tous les outils permettant le développement aisé d'applications.

Il est disponible pour toutes les plate-formes. Il gère l'autocomplétion ainsi qu'un nombre de projet quasi illimité. Il intègre aussi un debugger qui permet de debugger son application en temps réel.

Dans sa dernière version Qt Designer est intégré ce qui permet à partir d'une seule et même application de créer des application Qt riches en fonctionnalités.

DÉMONSTRATION

Scénario

Le but de la démonstartion est de créer un convertisseur Qt qui converti très simplement un chiffre en US Dollar vers l'Euro.

Cette petite application est composée des éléments graphiques suivants :

- un SpinBox contenant la valeur à convertir
- une ComboBox contenant la monnaie de départ
- un PushBouton qui exécute le calcul
- un Label indiquant le résultat
- un Label contenant l'unité d'arrivée (Euro)

Lors de cette démonstration, j'ai utilisé Qt Creator avec Qt Designer intégré.

BIBLIOGRAPHIE

www.qtsoftware.com/developer

www.developpez.com

http://oreilly.com/catalog/9780596000646/toc.html

http://vrlab.epfl.ch/~bhbn/qt_presentation/presentation.html

http://www.javaworld.com/javaworld/jw-08-2007/jw-08-qtjambi.html?page=1

http://www.digitalfanatics.org/projects/qt_tutorial/fr/chapter02.html

https://www.ldc.usb.ve/docs/qt/examples.html