Introduction

Historiquement, la bibliothèque Qt était destinée à créer des interfaces graphiques riches en utilisant le langage C++. Avec le temps, les besoins ont évolué et la bibliothèque graphique Qt est devenu un framework complet de développement multiplateforme, pouvant prendre en charge également la gestion des fichiers, du réseau ou encore le multimédia. La partie interface graphique de Qt a également évolué pour répondre à des besoins nouveaux et variés, et Qt Quick est le résultat de cette évolution. Pour des projets de petite taille ou des prototypes, Qt Quick permet un développement rapide sans nécessiter de connaissances en C++. Dans le cas de développements pris en charge par des équipes de taille importante regroupant différentes spécialités, en particulier des designers et des développeurs, Qt Quick répond à deux besoins importants : offrir un outil de développement d'interfaces graphiques basé sur un langage simple et facile à prendre en main par des non-développeurs ainsi qu'une amélioration de la communication entre les différents acteurs.

Note *› Cet ouvrage se focalise sur l'utilisation de Qt 5 et donc de Qt Quick 2. Cependant, la majorité des explications et codes d'exemples sont compatibles avec Qt 4 et Qt Quick 1. Pour avoir la liste des nouveautés de Qt Quick 2 par rapport à Qt Quick 1, consultez cette page sur Qt Project.*

1. Qu'est-ce que Qt Quick et le QML ?

Le module Qt Quick permet de développer facilement des interfaces graphiques, sans devoir passer nécessairement par l'apprentissage d'un langage complet et donc complexe comme le C++. Pour réaliser cela, le module Qt Quick se fonde sur deux langages : le QML (pour *Qt Markup Langage*, littéralement *langage de balises pour Qt*) et le JavaScript, bien connu des développeurs de site internet. Le premier est un langage déclaratif permettant de définir la structure d'une interface, les éléments graphiques qui la composent, leurs positions, leurs couleurs, leurs contenus. Le JavaScript est un langage de script permettant d'écrire des fonctions et de paramétrer le comportement des éléments graphiques manipulés.

Les différentes approches des langages utilisés par Qt

En informatique, plusieurs approches (paradigme) sont souvent possibles pour résoudre un même problème. Ces différentes solutions sont implémentées avec différents langages, mais tous les langages ne permettent pas d'implémenter toutes les approches.

- La programmation impérative consiste à décrire les différentes opérations que devra réaliser le processeur. Cette approche sera utilisée par exemple dans les codes JavaScript d'une interface Qt Quick ;
- la programmation objet consiste à décrire une solution en termes d'objets interagissant entre eux. Ce type de programmation sera utilisé par exemple en C++ avec Qt ;
- La programmation déclarative consiste à définir les composants qui sont utilisés et les paramétrer. Le concepteur d'une interface n'a pas à se préoccuper de comment sera affichée cette interface, mais simplement à configurer les éléments. Un exemple bien connu est une page HTML, qui pourra être affichée sur n'importe quel navigateur internet et n'importe quel système d'exploitation. Le langage QML utilise également ce paradigme, ce qui permet de réaliser des interfaces graphiques simplement et de pouvoir les porter sur différents systèmes (ordinateur, tablette, téléphone portable).

Les développeurs connaissant déjà l'utilisation de Qt en C++ peuvent légitimement se poser la question de l'intérêt de Qt Quick. De nos jours, les projets sont rapidement de taille conséquente et nécessitent un travail d'équipe, impliquant des personnes aux compétences multiples. En particulier, dans le cadre de la création d'interfaces graphiques complexes et ergonomiques, il est courant qu'un designer fasse partie de l'équipe de développement pour travailler spécifiquement sur le rendu final de l'application et l'ergonomie de l'interface. Dans de telles équipes, constituées de plusieurs spécialités, il est parfois difficile de communiquer lorsque chacun a ses propres besoins et contraintes et utilisent ses propres outils. Par exemple, le designer dessinera probablement l'interface utilisateur avec un logiciel de graphisme et concevra des maquettes pour schématiser le comportement de l'interface lorsque l'utilisateur réalisera une action particulière. Il lui sera difficile de réaliser correctement son travail s'il doit utiliser les outils des développeurs et un éditeur de code lui sera d'une utilité assez limitée. De même, le développeur aura des difficultés à intégrer directement les graphismes dans l'application, en particulier lorsqu'il devra implémenter correctement les comportements des éléments graphiques. Le va-et-vient continuel entre le designer, qui donne les corrections à appliquer à l'interface, et le développeur, qui réalise l'implémentation effective de cette interface, est une source potentielle de conflit et de perte d'efficacité et donc de rentabilité.

En proposant un langage simple, utilisable directement par les designers et par les développeurs, Qt Quick offre un outil commun facilitant les interactions dans les équipes. Il permet à chacun de se focaliser sur sa spécialité : le designer peut directement modifier l'interface en fonction du résultat obtenu et être indépendant des équipes de dévelop-

pement ; le développeur peut directement intégrer l'interface Qt Quick dans son application, sans devoir convertir des images en code.

Autre avantage important, une interface Qt Quick est constituée de fichiers `.qml` et `.js`, qui sont lus par l'application au moment de l'exécution et non lors de la compilation. De ce fait, il est possible de modifier l'interface sans devoir recompiler, ce qui se traduit par un gain de temps important lors de la conception de l'interface d'une application complexe, laquelle peut être relativement longue à compiler.

2. Organisation de ce module du livre

Attention *› Comme le reste du livre, ce module est d'abord organisé selon une logique thématique. Seuls les chapitres 8 à 12 suivent une progression pédagogique et requièrent les notions abordées au chapitre précédent. Autrement dit si vous démarrez en QML, lisez de préférence ces chapitres dans l'ordre. Après libre à vous de privilégier les thèmes qui vous intéressent. L'infographiste, quant à lui, commencera par le chapitre introduisant Qt Quick Designer, lequel permet de créer des interfaces sans écrire une ligne de code.*

Ce module couvre l'ensemble des notions de base à connaître pour développer des interfaces Qt Quick.

- *Créer des interfaces avec Qt Quick Designer* montre par le biais d'un exemple comment d'une part créer des interfaces graphiques en QML sans écrire une ligne de code, d'autre part exercer un contrôle direct des éléments mis en place à l'aide du Designer. Il s'adresse (au moins dans sa première partie) plus particulièrement aux designers et développeurs débutants, mais il pourra aussi intéresser (en particulier dans sa seconde partie) les développeurs plus avancés soucieux de séparer proprement le code de l'interface utilisateur et d'impliquer davantage les designers dans la conception.
- *Démarrer en QML* aborde les notions importantes du langage QML, les éléments graphiques de base et le positionnement de ces éléments dans une interface.
- *Introduction à JavaScript pour Qt Quick* présente ce qu'il faut connaître du langage JavaScript pour personnaliser le comportement des interfaces graphiques.
- *Gérer l'interactivité* explique comment prendre en charge les événements utilisateurs (souris, clavier, saisie de texte) et introduire de l'intéractivité dans vos applications QML. Pour bien aborder ce chapitre, vous devez avoir acquis les notions de base présentées dans les chapitres Démarrer en QML et Introduction à JavaScript pour Qt Quick.
- *Organiser ses interfaces graphiques* vous montre comment organiser les éléments graphiques les uns par rapport aux autres.

- *Optimiser son développement* explique comment créer des composants réutilisables, introduire des états et transitions, utiliser le modèle de conception modèle-vue.
- *Enrichir l'interface graphique* introduit Canvas2D, le moteur de particule et les animations pour enrichir vos interfaces.
- *Étude d'une première application avec Qt Quick* montre au travers d'un exemple de développement d'une calculatrice comment mettre en œuvre les éléments étudiés dans les chapitres précédents.
- *Qt Quick, QML et le C++*, à destination des développeurs connaissant la programmation en C++ avec Qt, montre comment créer des composants réutilisables en C++ ou intégrer des interfaces graphiques réalisées avec Qt Quick dans une application C++.
- *Gérer l'interactivité* montre comment votre application peut communiquer avec le réseau ou des bases de données.
- *Tests et débogage* aborde les outils de développement pour tester, déboguer, optimiser des interfaces Qt Quick.

3. Premiers pas avec Qt Quick

Une interface Qt Quick est constituée au minimum d'un fichier `.qml` contenant le code QML de l'interface. Pour créer un tel fichier, vous pouvez écrire directement du code QML en utilisant un éditeur de texte. Vous pouvez également concevoir l'interface à l'aide d'un éditeur graphique qui se charge de générer le code QML correspondant à l'interface créée. Le kit de développement de Qt est fourni avec l'éditeur Qt Creator, qui contient un tel éditeur graphique, Qt Quick Designer. (Pour installer Qt Creator, se reporter au chapitre Installation de Qt et introduction à Qt Creator).

Nous allons vous présenter rapidement comment créer un nouveau projet, passer de l'éditeur de code à l'éditeur graphique, et comment les deux interagissent.

Créer un nouveau projet

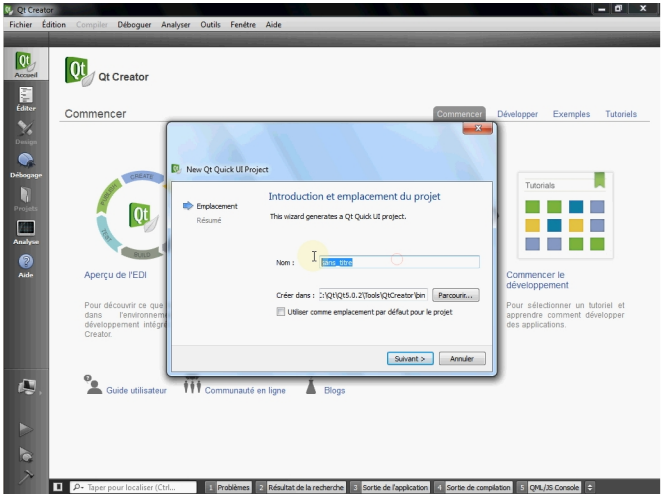
Lancez Qt Creator et créez un nouveau projet en cliquant sur le menu FICHIER puis NOUVEAU FICHIER OU PROJET... (ou directement avec Ctrl+N). Une fenêtre NOUVEAU s'ouvre et permet de choisir le type de projet ou fichier que l'on souhaite créer. Par défaut, l'élément sélectionné est APPLICATIONS dans la liste de PROJETS. Plusieurs types de projets sont proposés, nous allons nous concentrer sur les projets Qt Quick.

Les projets QT QUICK APPLICATION correspondent à des applications C++ contenant une scène Qt Quick, tandis que les projets QT QUICK UI correspondent uniquement à la partie Qt Quick des interfaces laquelle devra être ouverte en utilisant le visualisateur de scène QML. Les projets QT QUICK 1 correspondent à Qt 4, QT QUICK 2 aux versions à partir de Qt 5.

Note *› Les éléments Qt Quick définis dans les fichiers QML sont représentés dans un widget qui est appelé la scène QML. Pour le prototypage rapide, il n'est pas toujours nécessaire d'intégrer cette scène QML dans une application réelle, c'est pour cela que le visualisateur de scène QML permet d'avoir un aperçu rapide du résultat de votre travail. Ce visualisateur QML est bien entendu uniquement à utiliser lors de vos tests et non dans un environnement de production. Pour déployer une application QML dans un environnement de production, vous pouvez utiliser une application C++ comme fourni par les projets QT QUICK APPLICATION ou encore empaqueter le fichier QML dans un module.*

Sélectionnez le type de projet QT QUICK 2 UI puis cliquez sur CHOISIR.... L'assistant de création de projet permet de choisir le nom du projet et son emplacement. Dans la vidéo d'exemple suivante, nous allons nommer le projet `test-qtquick.qml` et le placer sur le bureau. Le nom de projet sera utilisé par Qt Creator pour nommer les fichiers créés. Cliquez sur SUIVANT : à la dernière page, vous avez la possibilité de configurer un gestionnaire de version pour les fichiers. Comme on n'utilise pas de gestionnaire pour cet exemple, cliquez simplement sur TERMINER pour finaliser la création du projet.

Figure 1 : *Création d'un nouveau projet Qt Quick*



Modifier le projet *via* l'éditeur de code

Qt Creator crée un nouveau projet, qui contient deux fichiers. Le premier est le fichier `test-qtquick.qmlproject`, qui contient la description du projet, les fichiers à inclure et les paramètres de génération et d'exécution. Le second est le fichier `test-qtquick.qml`, qui contient le code d'une interface graphique simple, consituée d'un rectangle et du texte *hello world* au centre et qui se ferme lorsque l'on clique dedans.

Note › *Ce code d'exemple est disponible dans le projet* `premier-projet-qtquick`  .

Ouvrez le fichier de projet `test-qtquick.qml` en double-cliquant dessus et examinez le code contenu dans ce fichier (voir listing ci-dessous). Vous noterez tout d'abord que le texte est coloré automatiquement par Qt Creator, en fonction de sa signification. Par exemple, les éléments graphiques du langage QML, comme `Rectangle` ou `Text`, sont en violet. Les propriétés des éléments graphiques, comme `width` ou `height`, sont en rouge.

```
 1. import QtQuick 2.0
 2.
 3. Rectangle {
 4.     width: 360
 5.     height: 360
 6.     Text {
 7.         anchors.centerIn: parent
 8.         text: "Hello World"
 9.     }
10.     MouseArea {
11.         anchors.fill: parent
12.         onClicked: {
13.             Qt.quit();
14.         }
15.     }
16. }
```

Pour visualiser l'interface correspondante, lancez le visualisateur de scène QML. Pour cela, allez dans le menu OUTILS, sélectionnez EXTERNE, QT QUICK puis QT QUICK 2 PREVIEW (QMLSCENE). Une fenêtre contenant le texte *Hello world* au centre s'affiche. Vous pouvez modifier facilement le texte de cette fenêtre. Pour cela, fermez la fenêtre et retournez sur le fichier `premier-projet-qtquick.qml`. Vous pouvez voir que la ligne 8 contient le code suivant : `text: "Hello world"`. Modifiez le texte entre les guillemets, par exemple : `text: "Bonjour tout le monde"`. Relancez le visualisateur QML, la fenêtre qui s'affiche contient maintenant le texte modifié.

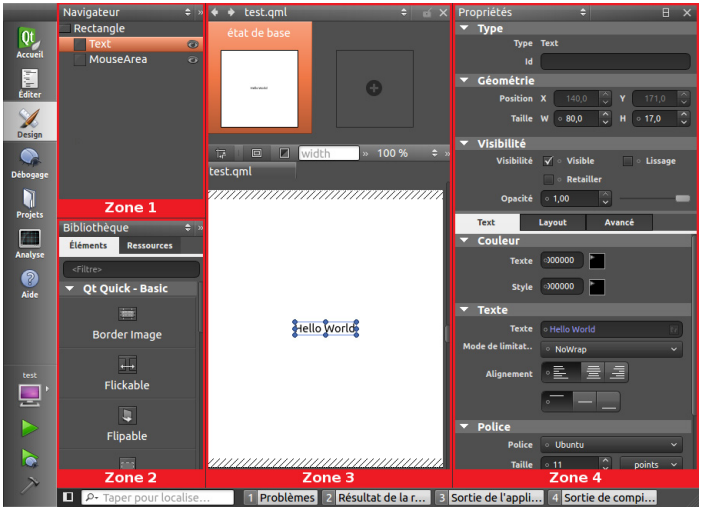
Modifier le projet *via* l'éditeur graphique

Éditez maintenant ce fichier avec l'éditeur graphique Qt Quick Designer. Pour cela, cliquez sur le mode DESIGNER à gauche de la fenêtre.

Attention *› Pour tester Qt Quick 2, il est conseillé d'utiliser la dernière version de Qt Creator, incluse dans le SDK de Qt 5.0.2. En effet, le Designer ne prend pas en charge toutes les versions de Qt Quick et une version plus ancienne pourrait ne pas fonctionner. Pour les versions de Qt Creator antérieure au SDK de Qt 5.0.1, seul Qt Quick 1 est pris en charge. La majorité des exemples vu dans ce module étant compatible avec Qt Quick 1, il vous suffit de remplacer* `import QtQuick 2.0` *par* `import QtQuick 1.0` *pour les faire fonctionner sur des versions antérieures de Qt Creator.*

Par défaut, le mode DESIGN contient plusieurs zones, visibles sur la copie d'écran suivante :

Figure 2 : *Interface graphique de création de projet Qt Quick dans Qt Creator*



- la zone 1 est la fenêtre du Navigateur, qui liste la hiérarchie des éléments contenus dans votre interface. Dans le projet par défaut, l'interface contient un élément `Rectangle`, qui contient lui-même un élément `Text` et un élément `MouseArea`. Ces éléments seront décrits au chapitre Démarrer en QML ;
- la zone 2, juste en dessous, est la fenêtre Bibliothèque, qui liste l'ensemble des éléments graphiques disponibles par défaut dans Qt Quick. En particulier, si vous

parcourez la liste des éléments, vous trouverez les éléments `Rectangle`, `Text` et `MouseArea` ;

- la zone 3, à droite de la zone Navigateur, permet de visualiser les différents états de l'interface ;
- en dessous se trouve la fenêtre de visualisation de l'interface. Dans notre exemple, cette fenêtre contient le texte *Hello world* créé par défaut ;
- tout à droite se trouve la zone Propriétés, qui permet de modifier les propriétés de l'élément sélectionné.

Vous allez maintenant modifier le texte contenu dans l'interface graphique. Pour cela, sélectionnez l'élément `Text` dans la fenêtre Navigateur : il apparaît alors entouré d'un rectangle bleu. Recherchez la propriété TEXTE dans la fenêtre PROPRIÉTÉS et remplacez le texte *Bonjour tout le monde* par *Hola mundo!*. Sauvegardez le fichier puis relancez le visualisateur QML. La fenêtre a été modifiée et contient maintenant le nouveau texte. Si on examine le code QML, on voit que la ligne 8 a été changée en : `text: "Hola mundo!"`.

On voit donc bien la relation entre le mode Design et le mode Éditer : toute modification dans l'un des modes se répercute dans l'autre mode et dans le résultat final. Utiliser le mode texte ou le mode visuel pour modifier un fichier est donc strictement équivalent.

Dans le mode Design, prenez un élément Text dans la zone Bibliothèque et déposez-le sur la zone de visualisation. Vous ajoutez ainsi un nouvel élément dans la fenêtre, qui pourra ensuite être modifié. Dans le code QML, vous pouvez voir que les lignes suivantes qui correspondent au nouveau texte ont été ajoutées :

```
Text {
    id: text1
    x: 168
    y: 106
    text: "text"
    font.pixelSize: 12
}
```

N'hésitez pas à tester différents éléments graphiques et modifier leurs propriétés pour vous familiariser avec les fonctionnalités proposées par Qt Quick.