7

Interface graphique avec les widgets

Les premières interfaces graphiques de Qt reposaient sur le module Qt GUI. Ce module a accompagné le framework depuis ses débuts. Il contenait l'ensemble des classes permettant de réaliser une interface graphique. Depuis Qt 5, tous les widgets de Qt GUI ont été transférés dans le nouveau module Qt Widgets. Qt 6 a conservé cette logique. Ainsi, Qt GUI contient les classes de rendus graphiques, de gestion d'événements mais aussi les deux classes QGuiApplication et QWindow. Le module Qt Widgets contient les classes définissant les widgets classiques associés aux interfaces graphiques (bouton, slider, label, texte éditable, group box, etc.), les classes de layout permettant d'organiser leur disposition dans l'application ainsi que des classes permettant de leur attribuer un style particulier.

N'importe quel développeur ayant commencé à développer avec Qt avant l'apparition des smartphones a probablement utilisé ces modules et donc les widgets pour créer les interfaces utilisateurs. Cette technologie est éprouvée et fournit un résultat fiable et multiplateforme. Rappelons que le multiplateforme de l'époque était essentiellement basé sur du bureau : Windows, Mac ou Linux. Au moment du développement de QML, la technologie de l'embarqué et des objets connectés avec des interfaces graphiques n'en était encore qu'à ses débuts. Il est important de noter que Qt Widgets se compile et peut techniquement s'utiliser sur Android ou iOS. Mais en le faisant, on découvre rapidement que la taille des widgets n'est pas adaptée à la taille des écrans de smartphones/tablettes et que l'utilisation d'un logiciel avec une souris est différente d'une utilisation avec des écrans tactiles. Par exemple, appuyer sur un QToolButton avec un doigt n'est pas aussi simple à faire qu'avec une souris, tant le bouton semble petit. On pourrait croire que déployer une même application sur différents écrans n'est qu'une histoire de mise à l'échelle, mais il n'en est rien. La taille d'un bouton sur un écran tactile doit être adaptée à la taille d'un pouce ou d'un index. Par ailleurs, déployer une application Qt Widgets sur tablette provoque des artefacts graphiques regrettables. Le QML a vraiment permis de régler tout cela

Cet extrait provient du livre Qt 6 en pratique
écrit par Mériadeg Perrinel - © 2025 Éditions D-BookeR

d'un coup et il évident que la technologie à choisir pour faire avec Qt du multiplateforme au sens actuel (bureau, mobile, embarqué et web) implique l'utilisation du QML au lieu de l'historique Qt Widgets.

Alors pourquoi continuer à maintenir ce module et l'utiliser ? Tout simplement car de nombreuses entreprises ne recherchent pas nécessairement ce multiplateforme au sens large et ne ciblent que des applications de bureau, là où l'on sait que ce module est éprouvé et efficace. En outre, si l'esthétique de l'application n'est pas une priorité pour ces entreprises et que l'objectif est plutôt une application scientifique ou industrielle fiable, robuste et efficace, alors il devient tout à fait correct de la coder en C++/Qt, épargnant par la même des coûts d'apprentissage du QML aux développeurs (attention, ce n'est pas une corvée de l'apprendre, seulement dans ce cadre-là, c'est inutile). J'ai eu moi-même à développer de nombreuses applications scientifiques en pur C++/Qt et sans une once de QML.

Ce livre comporte trois chapitres sur le QML et un seul sur Qt Widgets. En effet, les besoins applicatifs d'aujourd'hui ont changé prenant une direction nécessitant le QML avec des écrans dans des véhicules de tout genre (voitures, tracteurs, avions, etc.) ou des écrans embarqués pour de l'assistance médicale (tablettes dans les salles d'opérations avec une application accompagnant le traitement de patients), ou encore dans les maisons avec de l'électroménager connecté possédant des interfaces graphiques. L'application modulaire Qt6BookApp accompagnant ce livre a été déclinée en deux versions utilisant le même backend C++ mais présentant une interface graphique différente mais équivalente, l'une en QML et l'autre en Qt Widgets classique. Ce chapitre présente le développement d'interfaces graphiques basées sur Qt Widgets à l'aide de la déclinaison associée de l'application Qt6BookApp, que l'on nomme Qt6BookWidgetApp.

7.1. Conception d'une interface graphique pour une application de bureau

J'ai eu l'opportunité de développer de nombreuses applications avec les widgets, et certains éléments de conception sont récurrents. De manière générale, il y a une fenêtre principale (QMainWindow), sur laquelle sont placés différents panneaux à l'aide de widgets (QWidget ou ses classes enfants). Une barre de menus (QMenuBar) et une barre d'outils (QToolBar) sont également présentes et permettent d'activer des fonctions affectant l'application. Enfin, une barre d'état permet d'afficher en premier plan des informations importantes relatives à l'application. Bien sûr, il existe

Cet extrait provient du livre Qt 6 en pratique
écrit par Mériadeg Perrinel – © 2025 Éditions D-BookeR

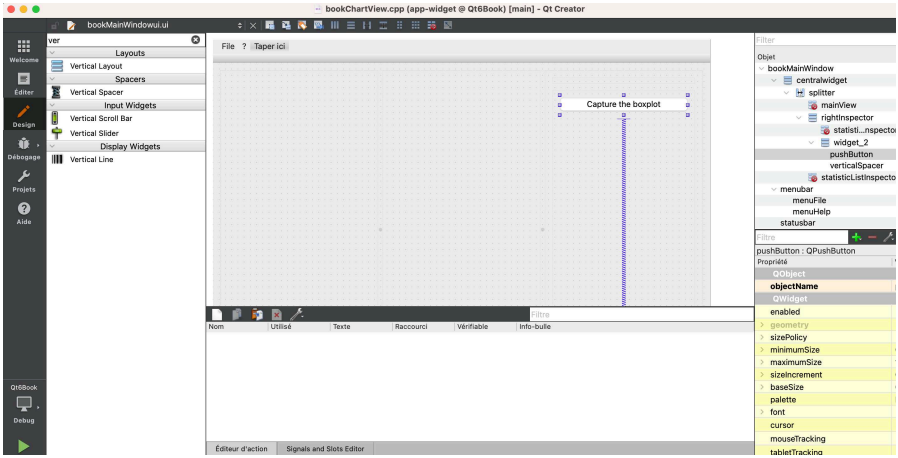
autant de façons de combiner et de personnaliser ces éléments qu'il y a d'applications différentes.

Les widgets sont vraiment l'essence même des applications utilisant le module Qt Widgets. Ce sont sur ces widgets de base que viendront se greffer les différents composants classiques d'une application (bouton, label, textEdit, etc.).

Ébaucher son application à l'aide du designer

Lors du développement d'une application, il est généralement intéressant de faire des maquettes pour voir à quoi l'application peut ressembler. En entreprise, lorsque l'on répond à un appel d'offres, faire une maquette de l'application demandée permet au prestataire de montrer au client un bon aperçu de ce qu'il se propose de faire s'il remporte le contrat. Qt Creator permet de faire cela. Cet outil est parfaitement adapté à l'écriture d'applications C++/Qt/QML. Personnellement, je code essentiellement avec celui-ci, même si d'autres sont disponibles, comme Visual Studio ou encore CLion. Mais je trouve Qt Creator parfaitement convivial, efficace et simple d'utilisation. Par ailleurs, pour en revenir aux maquettes, Qt Creator propose un designer d'interface graphique adapté au développement avec Qt Widgets. Il permet de générer très rapidement des maquettes avec Qt Widgets.

Figure 7.1 : *Maquette Qt 6 avec l'éditeur de design intégré à Qt Creator*



Cet extrait provient du livre Qt 6 en pratique
écrit par Mériadeg Perrinel – © 2025 Éditions D-BookeR