Centre Informatique pour les Lettres et les Sciences Humaines

TD 1 : Première rencontre avec Visual C++ et Qt Designer

Les documents décrivant les TD mêlent des *considérations générales* qui vous aident à comprendre la logique des actions décrites et des *instructions* qui exigent que vous fassiez quelque chose. Pour vous permettre de distinguer facilement ces deux types d'informations (et, éventuellement, de bien repérer où vous en êtes), les instructions sont suivies d'une "case à cocher" vierge. Si vous travaillez avec une version imprimée du TD, vous pouvez donc cocher ces cases à mesure que vous effectuez les actions requises.

Lorsqu'une commande doit être exécutée, elle est surlignée en mauve, alors que les indications relatives à l'endroit où elle se trouve sont surlignées en bleu. Par exemple :

Choisissez la commande Quitter dans le menu Fichier ☐.

Comme tous les suivants, ce premier TD a pour objectif de vous permettre de créer un petit programme illustrant l'utilisation concrète des notions introduites dans la Leçon correspondante. Un problème semble toutefois se poser :

1 - Créer un programme sans savoir programmer ?

Nous avons, dans la Leçon 1, donné de la création d'un programme une description qui est résumée dans le tableau suivant :

Nom de l'opération	*Produit*	*Responsable*	*Objectif*
Rédaction	Texte source	Programmeur (être humain)	Décrire la procédure à suivre pour parvenir au résultat souhaité
Compilation	Code objet	Compilateur (programme)	Traduire le texte source en une suite d'instructions exécutables par le processeur visé.
Edition de liens	Programme exécutable	Linker (programme)	Greffer sur le code objet tout ce qui lui manque pour devenir un programme exécutable dans le contexte visé.

Les deux dernières opérations (compilation et édition de liens) sont effectuées par des programmes, et il nous suffira donc d'apprendre à lancer ceux-ci. La rédaction du texte source, en revanche, semble poser un problème actuellement insurmontable : c'est vous qui devriez en être responsable, et la Leçon 1 n'a pas même commencé à vous présenter le moindre rudiment de C++, langage dans lequel ce texte doit être écrit.

La complexité des interfaces graphiques

Pour comprendre pourquoi ce problème ne se pose pas réellement, il vous faut réaliser que les "petits" programmes que nous allons créer au cours des différents TD sont en fait d'une complexité monstrueuse. A l'exception du TD2, ils se présentent en effet sous la forme de fenêtres affichées à l'écran et laissent à l'utilisateur différentes possibilités d'interaction, telles que le déplacement de la fenêtre en question sur l'écran, l'utilisation simultanée d'un autre programme, le choix arbitraire de l'ordre d'utilisation des éléments proposés dans la fenêtre (boutons, menus, etc.), ou la copie d'une information présentée dans la fenêtre en vue d'un collage dans un document géré par une autre application. De plus, ces programmes respectent les choix visuels généraux faits par l'utilisateur (couleur des fenêtres, taille de la police utilisée dans les menus, etc.).

Même en s'en tenant aux quelques fonctionnalités citées ci-dessus, la réalisation du moindre programme exigerait des efforts considérables (et quelques années d'études préalables) si nous devions assurer nous-mêmes intégralement la rédaction du texte source.

Aides à la création d'applications

Pour contourner cette difficulté, les programmeurs utilisent différents types d'artifices :

- Les librairies d'interface graphique sont des collections de séquences de code exécutable qui assurent non seulement la représentation à l'écran des éléments d'interface communément utilisés (fenêtres, boutons, zones d'édition, etc.), mais également le fonctionnement élémentaire de ces éléments (déroulement des menus, détection du fait que l'utilisateur clique sur un bouton, etc.). Un programmeur disposant d'une telle librairie n'a donc pas à rédiger cette partie du programme : il lui suffit, pour pouvoir disposer des fragments de code contenus dans la librairie en question, de préciser au linker que celle-ci doit être liée (c'est-à-dire incluse dans le programme exécutable).
- Les éditeurs d'interfaces se présentent comme des sortes de logiciel de dessin permettant de disposer à l'aide de la souris les différents éléments composant, par exemple, une fenêtre de dialogue. Ce qui les rend précieux est leur capacité à générer le texte source correspondant au dessin réalisé par le programmeur (ce texte source fait habituellement appel à une librairie graphique). Un programmeur disposant d'un éditeur d'interface n'a donc pas à rédiger la partie du programme qui crée une fenêtre et y dispose les divers boutons, zones d'édition et autres menus nécessaires. Il lui suffit de compiler le texte généré par l'éditeur d'interface et d'indiquer au linker que le code objet correspondant doit être lié.

- Les générateurs d'applications sont des logiciels capables, à des degrés divers, de créer des portions de texte source correspondant à des parties du programme allant au-delà de la gestion des éléments de l'interface utilisateur. Dans sa version la plus rudimentaire, ce type d'outils n'est pas à proprement parler un *générateur* d'applications, mais un simple dispositif de *choix* entre différents squelettes d'applications préfabriqués[1] assurant certaines fonctionnalités standard élémentaires. Le rôle du programmeur est ensuite d'habiller ce squelette, pour donner au programme les caractéristiques qui justifient son existence.

Notre environnement de travail

Le compilateur et le linker que nous allons utiliser sont ceux inclus dans l'environnement de développement Microsoft Visual C++ 6.0.

En plus de ces programmes, nous allons utiliser Qt Designer, qui est à la fois un éditeur d'interfaces (utilisant la librairie Qt) et un petit générateur d'applications. Qt Designer et la librairie Qt sont des produits (développés par la société Trolltech) qui s'intègrent suffisamment dans l'environnement Microsoft pour que l'ensemble ainsi constitué soit tout à fait utilisable par des débutants. C'est l'utilisation de Qt Designer qui va vous permettre de réaliser ce premier TD sans écrire vous mêmes la moindre ligne de C++.

2 - Préparatifs

La procédure décrite ci-dessous suppose que Qt Designer ait été intégré dans Visual C++. 6.0

> Cette condition est normalement remplie sur les machines mises à votre disposition par le Centre Informatique pour les Lettres et les Sciences Humaines. Si vous souhaitez pouvoir travailler dans un autre contexte, contactez votre enseignant.

Notion de projet

A la différence d'autres activités utilisant un ordinateur (le traitement de textes, par exemple), la création d'un programme implique la mise en œuvre de plusieurs programmes différents (éditeurs, compilateur, linker, etc.) travaillant sur un ensemble de fichiers (textes sources, librairies, etc.). L'objet sur lequel nous allons travailler n'est donc pas un simple fichier (comme celui représentant le document en cours de rédaction, dans le cas du traitement de texte) mais un *projet*. Concrètement, un projet se présente comme une collection de fichiers habituellement regroupés dans un dossier portant le même nom que le projet. L'un de ces fichiers porte l'extension .dsw et permet à Visual C++ de "savoir quoi faire" des différents autres fichiers du projet. C'est l'ouverture de ce fichier qui correspond à l'ouverture du projet.

> Si vous souhaitez modifier une lettre que vous avez écrite avec Word, par exemple, vous allez ouvrir le fichier maLettre.doc dans lequel vous avez sauvé votre texte. Si vous souhaitez modifier un programme que vous avez créé avec Visual C++, il n'est en revanche pas question d'ouvrir le fichier monProgramme.exe qui contient ce programme. Cette ouverture déclencherait en effet l'*exécution* du programme en question, ce qui ne vous donnerait aucun moyen de le modifier. Pour améliorer un programme, il faut au contraire ouvrir le projet qui a débouché sur sa création, modifier le texte source, puis relancer le compilateur et le linker pour fabriquer un nouveau fichier monProgramme.exe (remarquez qu'il n'est donc pas nécessaire de disposer du fichier .exe contenant l'ancienne version).

Choix d'un nom de projet

Pour des raisons techniques inavouables, les projets que nous allons créer doivent porter des noms ne comportant ni espaces ni minuscules accentuées.

Dans le cas présent, je vous suggère de choisir le nom TD01.

Choix de l'emplacement où vont être créés les fichiers du projet

Le dossier portant le nom du projet et contenant les fichiers correspondants sera créé automatiquement lors de la création du projet. Il nous faut cependant choisir un emplacement convenable pour la création de ce dossier, et deux facteurs importants doivent être pris en compte :

[1] Ce qu'on appelle, en anglais, "an application framework".

- le compilateur peut être amené à créer des fichiers temporaires volumineux (plusieurs méga-octets) dans le dossier projet ;
- les différents programmes utilisés (éditeurs, compilateurs, linker, etc.) ont fréquemment besoin d'accéder aux différents fichiers du projet.

Le premier facteur INTERDIT d'envisager la création d'un projet sur une disquette. Le second suggère fortement de choisir un disque dur local plutôt qu'une unité accessible via un réseau ou un disque local peu rapide (disque zip, par exemple).

Quel que soit votre choix, il faut savoir que le dossier dans lequel le dossier projet va être créé doit déjà exister (il ne peut pas être créé automatiquement en même temps que le dossier projet) et qu'il ne doit pas déjà contenir un dossier portant le nom choisi pour le projet (ce qui rendrait impossible la création automatique du dossier projet).

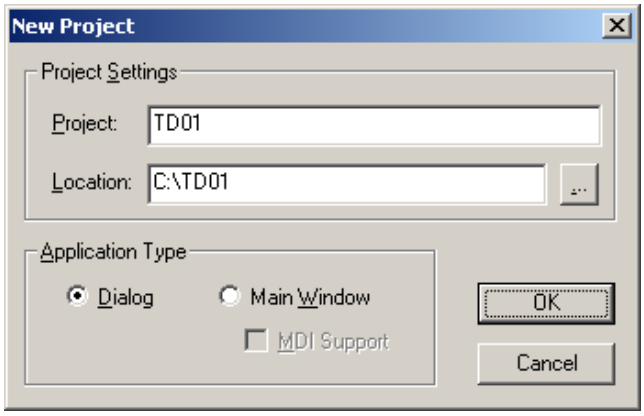
> Mon expérience me conduit également à déconseiller aux étudiants d'utiliser des dossiers projets placés directement sur le bureau, qui me semblent courir des risques de dysfonctionnement inexplicables plus importants que les projets résidant dans l'arborescence normale des dossiers. (Notez cependant qu'il ne s'agit là que d'une impression personnelle que je ne saurais étayer d'aucun argument technique ou statistique sérieux.)

Création du projet

Après avoir lancé Visual C++ ☐, la création d'un nouveau projet est obtenue simplement en cliquant ☐ sur le bouton  (New Qt Project). La boîte de dialogue "New Project" s'ouvre alors (cf. ci-contre).

Il faut ensuite indiquer le nom du projet ☐ et l'emplacement ☐ où le dossier qui va recevoir les fichiers relatifs à ce projet doit être créé.

Le "type d'application" créé par défaut est un "Dialog", ce qui convient parfaitement à notre propos. Il suffit donc maintenant de cliquer sur le bouton "OK" ☐.



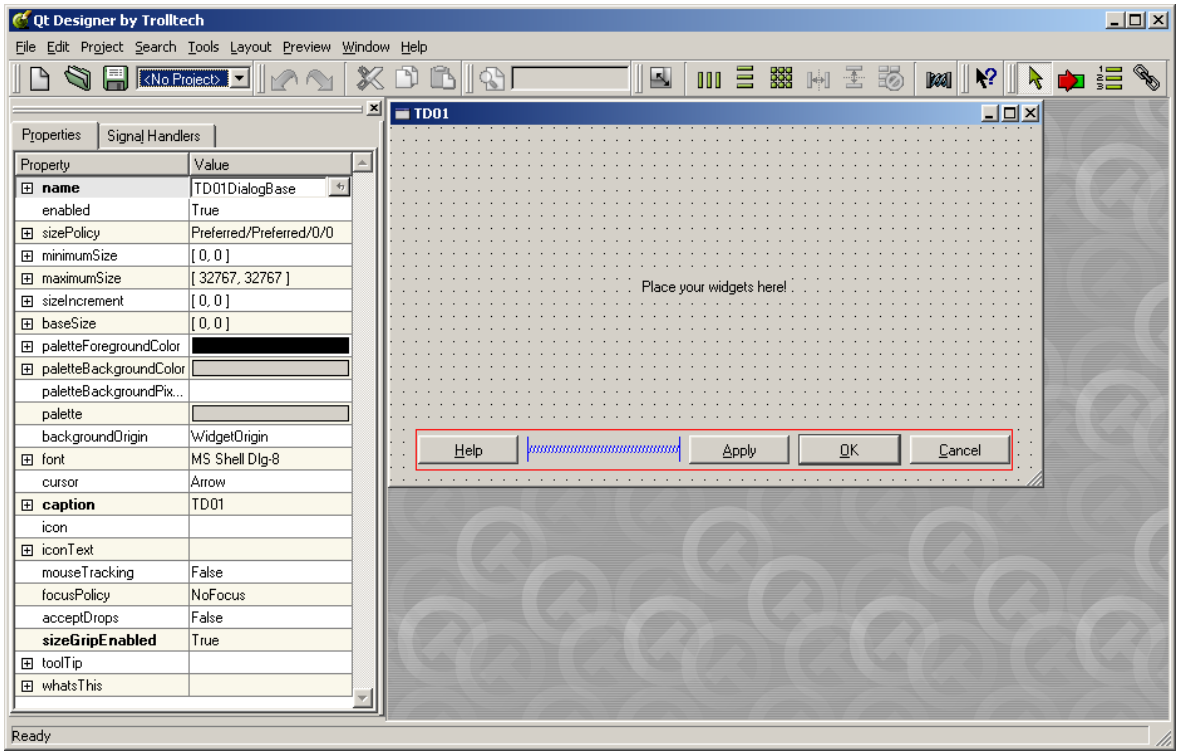
Le dialogue "Nouveau projet Qt"

> Le dialogue de création d'un projet QT comporte plusieurs imperfections. Tout d'abord, la zone "Location" affiche un chemin qui n'existe pas encore, puisqu'il comporte le nom du dossier projet qui va être créé. Par ailleurs, l'utilisation du bouton "..." conduit à l'ouverture d'un dialogue *non modal*, ce qui signifie qu'il est possible d'oublier de le valider avant de refermer le dialogue de création de projet. Cet oubli conduit à une situation incohérente qu'il est difficile de rectifier.
>
> Pour finir, même l'utilisation normale du bouton "..." conduit parfois à une "Location" syntaxiquement incorrecte (redoublements de barres obliques), ce qui se traduit, lorsqu'on valide le dialogue de création de projet, par l'apparition du merveilleux message d'erreur représenté ci-contre.
>
> 
>
>
> En cas de problème, refermez Visual C++ et QT Designer, effacez le dossier éventuellement créé et recommencez la procédure à partir du début...

3 - Création du programme

La création d'un "projet Qt" provoque le lancement de Qt Designer, qui propose un certain nombre de fenêtres et outils divers. Pour simplifier la situation et rendre moins intimidante notre première rencontre avec Qt Designer, utilisez la commande Views du menu Window pour désélectionner toutes les propositions, à l'exception de "Property Editor/Signal Handler" ☐.

La scène qui vous est offerte devrait alors ressembler plus ou moins à l'image suivante :



L'interface utilisateur de Qt Designer

La fenêtre intitulée "TD01" est une représentation de l'interface que le programme que nous allons créer proposera à ses utilisateurs. C'est donc dans cette fenêtre que nous allons disposer les différents éléments d'interface que nous souhaitons utiliser.

Les éléments composant une interface graphique (boutons, menus, zones d'éditions, etc.) sont appelés des widgets, ou dans la terminologie propre à Windows, des contrôles.

Lorsque l'un des widgets présents dans notre projet d'interface est sélectionné, la fenêtre "Properties/Signal Handlers" affiche des données le concernant, ce qui va nous permettre de donner à notre interface les caractéristiques exactes que nous souhaitons.

Première version : affichage d'un message

Dans la fenêtre "TD01", cliquez sur la zone de texte "Place your widgets here!" ☐

Dans l'onglet "Properties", repérez la ligne intitulée "text" et cliquez dans la colonne "Value", de façon à pouvoir modifier ce texte ☐

Effacez le texte proposé par Qt Designer et tapez un message de votre cru (cf. exemple ci-contre) ☐

Le texte que vous avez tapé devrait maintenant apparaître dans la fenêtre "TD01", en lieu et place du message originel. Ajustez, si nécessaire, la taille de cette zone de texte, de façon à ce que votre message apparaisse intégralement ☐.

Il est maintenant temps de générer le programme correspondant aux spécifications que nous venons de donner. Comme cette tâche est du ressort de Visual C++ et non de Qt Designer, il nous faut faire en sorte que les informations collectées par ce dernier soient rendues disponibles pour le compilateur. Il suffit pour cela d'enregistrer le résultat de vos efforts (Menu File, commande Save) ☐.

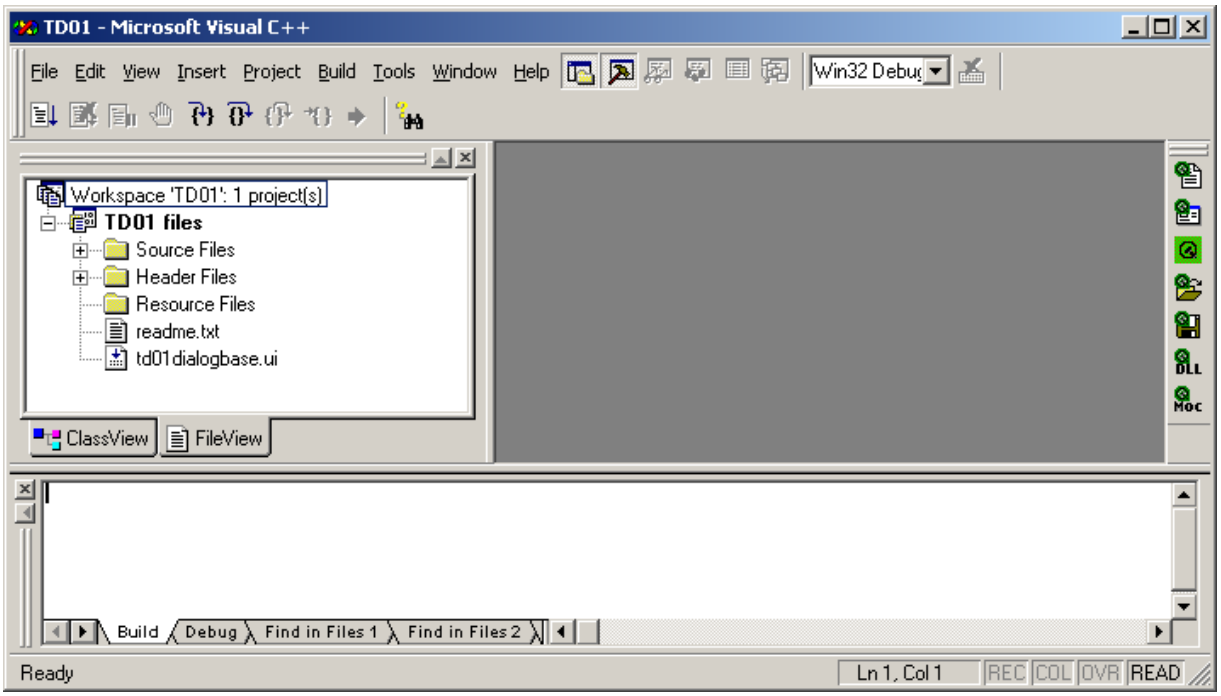
Avant de compiler un programme, il ne faut pas oublier de sauvegarder les modifications faites à son interface à l'aide de Qt Designer.

Dans la barre de tâches, cliquez sur l'icône représentant un anneau de Moebius coloré, de façon à retourner dans Visual C++ ☐.

L'icône de Visual C++

Il est inutile de refermer Qt Designer, nous allons encore en avoir besoin.

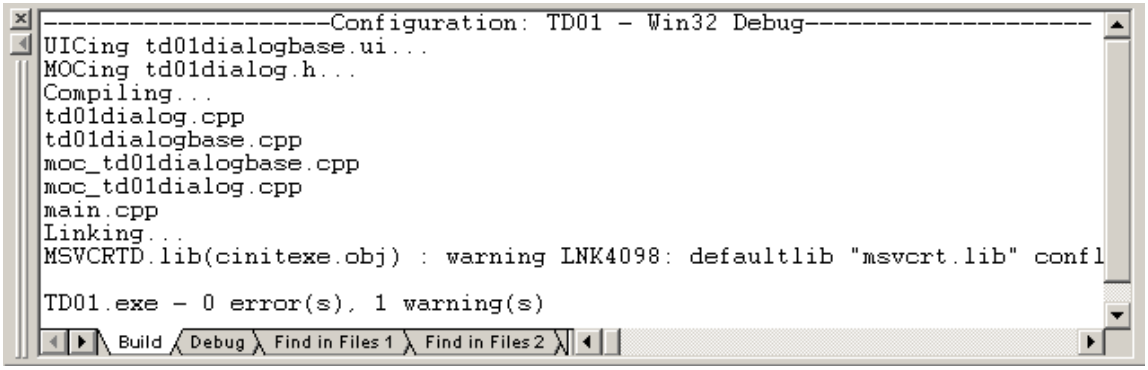
Vous devez donc vous retrouver face à Visual C++ :



L'interface utilisateur de Visual C++

La compilation de tous les fichiers source et l'édition des liens peuvent être déclenchés par une commande unique : dans le menu Build, choisissez la commande "Build TD01.exe" (ou, si vous êtes paresseux, contentez-vous d'appuyer sur la touche F7) .

Après quelques instants de traitement, la fabrication de notre programme s'achève et Visual C++ nous propose un petit rapport d'activité :



Un "rapport d'activité" de Visual C++

Ce rapport indique que des traitements préalables (UIC et MOC) ont tout d'abord été effectués sur les fichiers créés par Qt Designer (td01dialogbase.ui et td01dialog.h). Cinq fichiers contenant du texte source en C++ ont ensuite été compilés (ce sont les fichiers portant l'extension .cpp), et la fabrication du programme s'est achevée par l'invocation du linker. Ce dernier a émis un message d'avertissement (warning) concernant un conflit entre deux librairies, mais ce problème peut être considéré comme sans conséquences[2] et, en définitive, le programme TD01.exe a été construit sans qu'aucune erreur n'ait été détectée.

Pour essayer le programme qui vient d'être créé, pressez la touche F5 .

Deuxième version : affichage numérique de la position d'un curseur

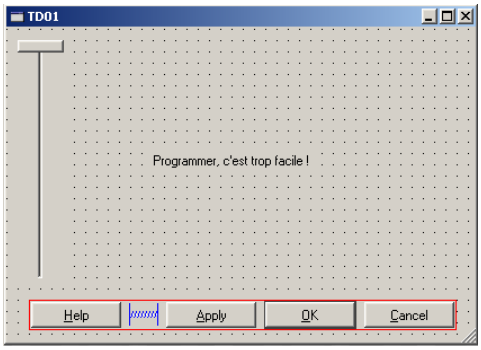
Maintenant que nous avons une bonne idée du processus général de création d'un programme, nous pouvons envisager de donner à notre projet des fonctionnalités un peu plus attrayantes que le simple affichage d'un message, aussi encourageant soit-il.

[2] Si vous préférez le résoudre, allez dans le menu "Project" et choisissez la commande "Settings". Dans l'onglet "Link" du dialogue qui apparaît alors, choisissez la catégorie "Input". Dans la zone de texte "Ignore libraries", tapez "msvcrt.lib". Les prochaines tentatives d'édition de liens de ce projet ne donneront plus naissance à ce warning.

Dans la barre de tâches, cliquez sur l'icône , de façon à retourner dans Qt Designer .

> Si vous avez refermé Qt Designer, son icône n'apparaît plus dans la barre de tâches. Vous pouvez rétablir la situation de la façon suivante : dans l'onglet "File View" de la fenêtre "Workspace" de Visual C++, double-cliquez sur le fichier td01dialogbase.ui. (si la fenêtre Workspace est elle-même invisible, vous pouvez la faire apparaître à l'aide de la commande du même nom proposée dans le menu View de Visual C++).

Dans le menu "Tools" de Qt Designer, choisissez l'option "Input" et, dans le sous-menu qui apparaît alors, la commande "Slider" .

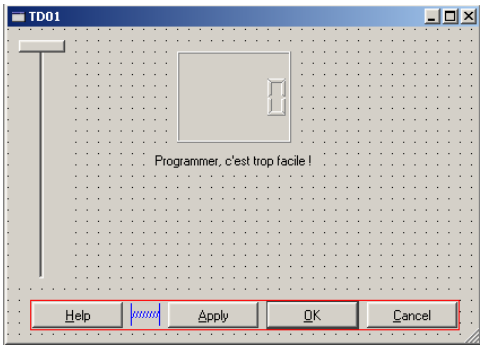


Placez votre pointeur de souris quelque-part dans la région supérieure gauche de notre projet d'interface, enfoncez le bouton gauche de la souris et, tout en maintenant ce bouton enfoncé, faites glisser la souris vers le bas et vers la droite . Vous venez d'ajouter un widget de type "slider" à notre projet d'interface, qui doit maintenant ressembler à l'image présentée ci-contre.

> Après avoir créé un widget, vous pouvez en changer la taille et la position, ou même le supprimer (sélectionnez le widget et appuyez sur la touche "d'effacement à droite").

Les "sliders" permettent à l'utilisateur d'un programme d'effectuer certains réglages, un peu comme un potentiomètre linéaire peut permettre à l'utilisateur d'un amplificateur d'ajuster le volume sonore. Du point de vue du programmeur, un slider se caractérise par une valeur entière, qui indique dans quelle position l'utilisateur a placé le curseur.

L'objectif de notre deuxième version du projet est simplement d'afficher cette valeur. Pour cela, nous devons introduire un second widget dans notre interface.



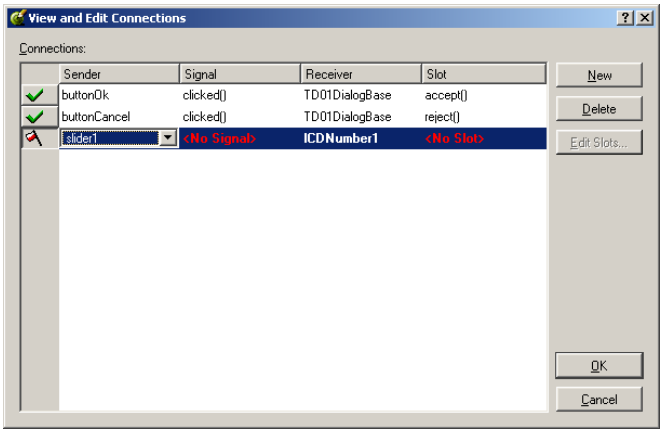
Dans le menu "Tools" de Qt Designer, choisissez l'option "Display" et, dans le sous-menu qui apparaît alors, la commande "LCDNumber" .

Positionnez ce widget au-dessus du message affiché dans notre projet d'interface, qui doit maintenant ressembler à l'image présentée ci-contre .

Il nous reste maintenant à exprimer l'idée que, lorsque le slider est déplacé, il doit signaler au LCDNumber que la valeur à afficher n'est plus la même.

Dans le menu "Tools", sélectionnez la commande "Connect Signal/Slots" .

Amenez le curseur de la souris (qui est maintenant une croix plutôt qu'une flèche) sur le slider, enfoncez le bouton gauche et, et, tout en maintenant ce bouton enfoncé, faites glisser la souris vers le LCDNumber. Lorsque vous relâchez le bouton de la souris dans le LCDNumber , le dialogue de connexion signal/slot apparaît (cf. ci-contre).



Le dialogue de connexion signal/slot

Ce dialogue propose un tableau décrivant les connexions existant entre les différents éléments de notre interface. La connexion que nous venons de créer figure sur la dernière ligne, où un petit drapeau rouge et blanc signale qu'elle n'est encore que partiellement spécifiée.

La première colonne indique que le widget nommé "slider1" doit envoyer une information (c'est lui le *sender*, c'est à dire l'émetteur), alors que la troisième indique que c'est le widget nommé LCDNumber1 qui recevra cette information (c'est lui le *receiver*, c'est à dire le récepteur).

La deuxième colonne devrait indiquer à la fois quand l'émission doit avoir lieu et quelle information doit être émise. Cliquez sur le texte <No Signal> et, dans le menu qui apparaît alors, choisissez l'option "`valueChanged(int)`" ☐.

> Nous souhaitons, en effet, que le slider signale tout changement de la valeur correspondant à la position de son curseur.

La quatrième colonne devrait, pour sa part, indiquer ce que le LCDNumber est censé faire du message qu'il va recevoir. Cliquez sur le texte <No Slot> et, dans le menu qui apparaît alors, choisissez l'option "`display(int)`" ☐.

> Nous souhaitons, en effet, que le LCDNumber affiche (en anglais : to display) la valeur reçue.

Le petit drapeau rouge et blanc est maintenant remplacé par une marque verte qui indique que notre connexion est en état de marche. Nous pouvons donc cliquer sur le bouton "OK" pour refermer le dialogue d'établissement d'une connexion ☐.

En vous inspirant des instructions suivies lors de la mise au point de la première version, générez ☐ et essayez ☐ cette seconde version de notre programme.

Troisième version : introduction d'un quadrant

Modifiez notre projet d'interface pour qu'il ressemble à l'image ci-contre ☐.

> Le widget situé à droite de la fenêtre est de type "dial" et est proposé sous la rubrique "Input" du menu "Tools".
>
> Pour améliorer la lisibilité, vous pouvez modifier la propriété "font" du widget affichant votre texte, ainsi que les propriétés "numDigits" et "segmentStyle" du LCDNumber.

Créez une connexion pour que les mouvements du curseur se traduisent non seulement par l'affichage de la valeur numérique correspondant à sa nouvelle position, mais aussi par un déplacement de l'aiguille du quadrant ☐.

> Le "slot" du dial qui permet de fixer la position de l'aiguille s'appelle "`setValue(int)`".

En vous inspirant des instructions suivies lors de la mise au point de la première version, générez ☐ et essayez ☐ cette troisième version de notre programme.

Que se passe-t-il lorsque vous utilisez la souris pour déplacer directement l'aiguille du quadrant ?

4 - Interrompre et reprendre le travail sur un projet

Refermez Visual C++ ☐ et Qt Designer ☐.

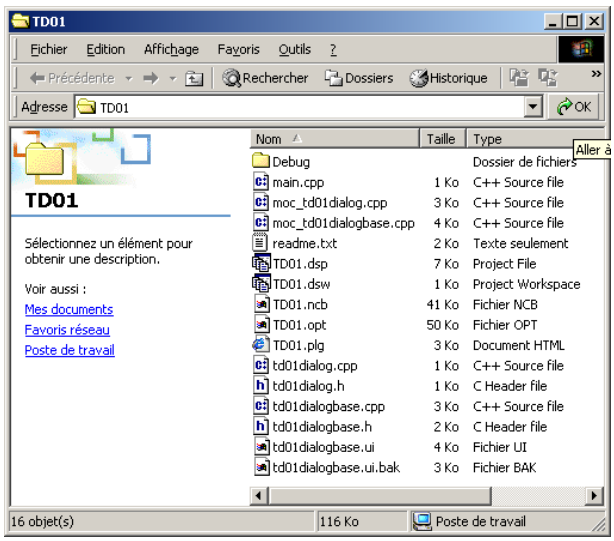
Contenu du dossier projet

Retrouvez votre dossier projet et ouvrez-le ☐.

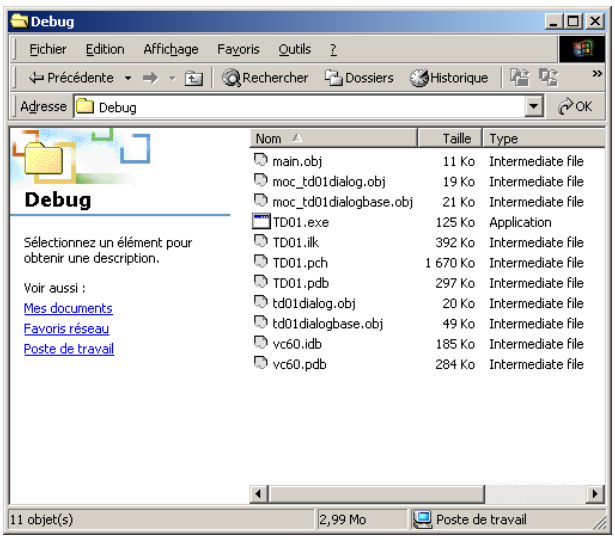
Outre une quinzaine de fichiers nécessaires à la construction de notre programme, ce dossier contient un sous-dossier nommé Debug, qui a été créé pour recevoir les fichiers créés pendant la construction de notre programme.

Ouvrez le sous-dossier Debug ☐.

Deux des fichiers contenus dans ce sous-dossier sont particulièrement remarquables. L'un, qui porte l'extension .pch, est d'une taille considérable (supérieure à la somme des tailles de tous les autres fichiers du dossier projet !). L'autre, qui porte l'extension .exe, contient le programme que nous venons de créer, comme vous pouvez le constater en double-cliquant dessus ☐.



Le contenu du dossier projet



Le contenu du sous-dossier Debug

Refermez notre programme et jetez le sous-dossier Debug à la poubelle.

> Le dossier Debug ne contient que des fichiers créés automatiquement lorsque nous lançons la compilation et l'édition de liens. Dans la mesure ou nous conservons les autres fichiers du dossier projet, le contenu du sous-dossier Debug n'a rien d'indispensable : si nous souhaitons recréer ces fichiers, il nous suffit de presser F7 dans Visual C++....

La taille du dossier projet est maintenant des plus raisonnables (de l'ordre d'une centaine de kilo-octets, dans le cas présent) et, en cas de besoin, vous pouvez parfaitement le copier sur une disquette.

Reprendre le travail

Pour reprendre un projet interrompu, le plus simple est d'ouvrir le dossier projet et de double-cliquer sur le fichier .dsw qu'il contient.

Cette action a pour effet d'ouvrir le projet avec Visual C++ et, comme nous l'avons vu, un double-clic sur le fichier portant l'extension .ui (onglet FileView de la fenêtre Workspace) ordonne à QT Designer de reprendre le travail sur notre projet d'interface.

5 - Exercice

La troisième version de notre programme n'a pas, nous l'avons vu, un fonctionnement très satisfaisant lorsque l'utilisateur manipule directement l'aiguille du quadrant : ni la valeur affichée par le LCDNum ni le curseur du slider ne suivent le mouvement, ce qui provoque un saut de l'aiguille fort disgracieux lorsqu'on déplace ensuite le curseur du slider.

Corrigez ces problèmes.

6 - Qu'avons-nous appris ?

1 - Les projets Qt sont mis au point à l'aide de deux logiciels complémentaires : Qt Designer permet de dessiner l'interface graphique et génère le code C++ correspondant. Ce code (ainsi que celui que nous rédigerons nous-mêmes) est ensuite compilé par Visual C++.

2 - Quand on passe de Qt Designer à Visual C++, il ne faut pas oublier d'enregistrer le travail accompli (faute de quoi ce travail ne serait pas pris en compte lors de la compilation).

3 - Pour reprendre le travail sur un projet, il faut ouvrir le fichier .dsw correspondant.

4 - Pour transporter un projet, on peut effacer le sous-dossier Debug, ce qui réduit le dossier correspondant au projet à une taille permettant de le copier sur une disquette.