Code::Blocks et les DevPaks wxWidgets

Introduction

Bonjour à tous.
Ce tutoriel a pour but de vous faire installer la bibliothèque wxWidgets sous Windows, afin de l'utiliser pour programmer vos logiciels sous Code::Blocks.
De plus, nous allons faire en sorte que cette installation soit compatible avec l'assistant nouveau projet de Code::Blocks.
Plusieurs méthodes sont possibles pour réaliser cela :

- Télécharger les sources de wxWidgets et les compiler : cette méthode est un peu longue à mettre en oeuvre et compliquée, surtout pour les débutants.
- Installer wxPack, qui vous permettra d'utiliser wxWidgets sous Code::Blocks ou Ms Visual C++ : cette méthode est pratique, mais le téléchargement (et surtout les fichiers installés) représente plus de 230Mo, et l'assistant projet pour Visual C++ est buggé.
- Utiliser les DevPacks de FredCL.

C'est cette dernière méthode que je vais vous expliquer.
Le seul problème qui existe avec les DevPacks, c'est qu'ils sont initialement prévus pour l'IDE Dev C++.
Code::Blocks intègre bien un plugin permettant de les installer, mais il faudrait pour cela qu'ils soient placés dans un dépôt officiel de DevPacks, ce qui n'est pas le cas.
De plus, les fichiers contenus dans ces DevPacks ne sont pas compatibles avec l'assistant *Nouveau projet wxWidgets* de Code::Blocks, car les noms des fichiers libs (les fichiers ".a") ne suivent pas la norme "officielle" wxWidgets.
Qu'à cela ne tienne, nous allons faire en sorte que tout fonctionne parfaitement bien.

Préparons le terrain

Télécharger les DevPaks

Cette partie est assez simple à réaliser : il vous suffit de vous rendre sur le site de l'auteur (que l'on remercie au passage pour sa réactivité et pour le travail effectué à chaque nouvelle version de wxWidgets) : http://cfred.free.fr , et de vous rendre dans la rubrique « téléchargements ».
Il vous faudra télécharger le fichier "Entête et aide", ainsi qu'au minimum un des 4 autres fichiers, en fonction de la méthode que vous utilisez pour programmer : Ansi, Unicode, Statique, Dynamique.
Je vous conseille de télécharger les 4 fichiers, vous pourrez ensuite choisir votre méthode de compilation lors de la création de votre projet avec Code::Blocks.
Créez le répertoire dans lequel vous voulez installer ces DevPacks. Pour ma part, je vais tout placer dans `C:\wx28`. Ça permettra de limiter les modifications à effectuer lors du passage à une version supérieure.
Pour information, ceux d'entre-vous qui ont déjà suivi ce tutoriel pour les versions 2.8.3, à 2.8.7 ne sont pas obligés de modifier l'arborescence créée à ce moment là. Seuls les fichiers sont à mettre à jour.
Dans ce répertoire, créez un dossier temporaire (je choisi `C:\wx28\devpacks`) dans lequel vous placez vos fichiers téléchargés.

Extraire le contenu des DevPacks

Un DevPack n'est autre qu'une archive *tar* compressée par la méthode *BZip2*, et dont l'extension a été renommée en ".DevPak".
La première action sera donc de les décompresser.
Vous aurez pour cela besoin d'un gestionnaire d'archives permettant la gestion des archives tar et BZip. C'est le cas,

par exemple, de WinRar, et de 7Zip.
Après avoir installé l'un de ces logiciels, et placé vos DevPacks dans le dossier temporaire créé ci-dessus, il faut maintenant les renommer :

- Vous pouvez le faire manuellement pour chaque fichier depuis l'explorateur Windows
- Vous pouvez utiliser la ligne de commande : `rename C:\wx28\devpacks\*.DevPak *.tar.bz2`.

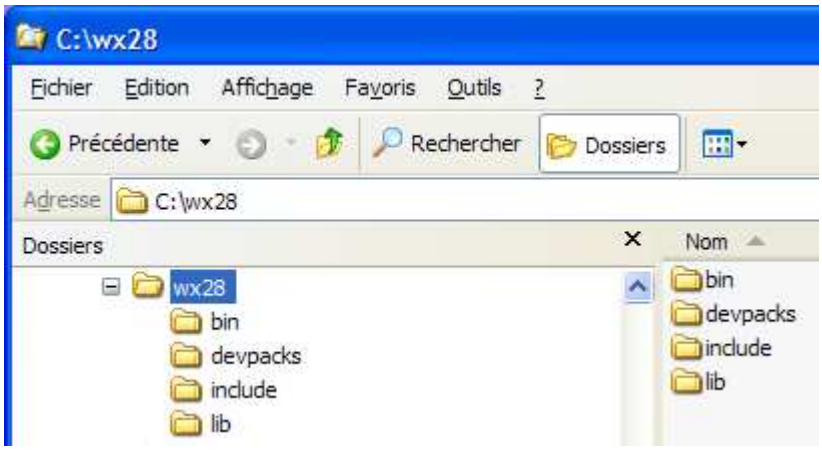
Il faut maintenant extraire le contenu de ces archives :

- Avec WinRar : Clic droit → WinRar → Extraire ici (en répondant oui pour tous quand on vous demande confirmation pour remplacer le fichier *LICENCE.txt*).
- Avec 7Zip : il faut faire 2 manipulations par fichier : le premier “Extraire ici” permet d'obtenir l'archive tar originale, qu'il faut ensuite sélectionnez et désarchiver avec un nouveau “Extraire ici”.

Vous devriez donc au final obtenir 3 répertoires (`packinclude`, `Ansi`, `Unicode`), ainsi qu'une ribambelle de fichiers ”.txt” et ”.DevPackage” que vous pouvez supprimer.

Créer la structure du répertoire d’installation

Nous avons vu précédemment que nous allions installer les DevPacks dans le répertoire `C:\wx28`. Nous y avons créé un sous répertoire temporaire `devpacks` dans lequel nous avons décompressés les fichiers téléchargés.
Il nous faut encore créer un-sous répertoire `bin` dans lequel nous allons placer les dll de wxWidgets, un sous-répertoire `lib` dans lequel nous allons placer … les fichiers libs, et un sous-répertoire `include` dans lequel nous placerons les headers.
Lorsque cette arborescence est créée, nous obtenons quelque-chose comme ceci :



L'installation

Déplacer les fichiers

Tout d'abord, nous allons placer toutes les dll dans le répertoire `bin` créé ci-dessus.
Comme ces fichiers se trouvent dans différents dossiers, et qu'ils sont mélangés à d'autres fichiers, le plus simple est de faire un clic droit sur le répertoire `devpacks` et de sélectionner “Rechercher…” en indiquant ”*.dll*” comme nom de fichier (je vous conseille de mettre les guillemets lors de la saisie, afin de palier à un problème qui arrive si l'option ”*Masquer les extensions des fichiers dont le type est connu*” est activée sur votre système).
Pour information, comme j'ai téléchargé tous les DevPacks, j'ai dans mon arborescence 44 fichiers dll.
Lorsque la recherche est terminée, sélectionnez tous les résultats et déplacez-les dans le répertoire `bin` que nous avons créé. Pour ceux d'entre-vous qui ne disposent pas de cette possibilité, les fichiers à déplacer se trouvent dans les répertoires :

- `C:\wx28\devpacks\Ansi\packdynamic\lib\msw-ansi-2.8.8\Dynamic`

- C:\wx28\devpacks\Unicode\packdynamic\lib\msw-unicode-2.8.8\Dynamic

Ensuite, nous allons copier les fichiers en-tête (les headers). Il faut pour cela déplacer le dossier wx qui se trouve dans C:\wx28\devpacks\packinclude\include\wxWidgets-2.8.8 et le placer dans le répertoire C:\wx28\include.
Il nous reste à mettre en place les fichiers libs (portant l'extension "*.a*"). Il faut tout d'abord créer les répertoires qui vont accueillir ces fichiers.
Dans le dossier C:\wx28\lib, créez les sous-répertoires gcc_lib et gcc_dll*. Ces deux répertoires vont respectivement accueillir les libs statiques et dynamiques.

- Déplacez tous les fichiers "**.a*" se trouvant dans le dossier C:\wx28\devpacks\Ansi\packstatic\lib\msw-ansi-2.8.8\Static pour les mettre dans C:\wx28\lib\gcc_lib.
- Déplacez tous les fichiers "**.a*" se trouvant dans le dossier C:\wx28\devpacks\Ansi\packdynamic\lib\msw-ansi-2.8.8\Dynamic pour les mettre dans C:\wx28\lib\gcc_dll.
- Déplacez tous les fichiers "**.a*" se trouvant dans le dossier C:\wx28\devpacks\Unicode\packstatic\lib\msw-unicode-2.8.8\Static pour les mettre dans C:\wx28\lib\gcc_lib.
- Déplacez tous les fichiers "**.a*" se trouvant dans le dossier C:\wx28\devpacks\Unicode\packdynamic\lib\msw-unicode-2.8.8\Dynamic pour les mettre dans C:\wx28\lib\gcc_dll.

Vous aurez à confirmer l'écrasement de quelques fichiers : c'est normal (certaines libs sont les mêmes en Ansi et en Unicode).
Ensuite, rendez-vous dans le dossier C:\wx28\inlcude\wx. Vous remarquerez deux dossiers : msw-ansi-2.8.8 et msw-unicode-2.8.8. Ils contiennent, dans leur arborescence, le fichier *setup.h* servant à déclarer les options utilisées lors de la compilation.
Le problème est qu'ils ne sont pas à leur place : une installation "standard" de wxWidgets les aurait placés dans le dossier contenant les libs.

- Déplacez ces deux dossiers pour les mettre dans le dossier C:\wx28\lib\gcc_dll.
- Renommez-les : msw_ansi_2.8.8 devient msw et msw_unicode_2.8.8 devient mswu.
- Placez une copie de ces deux dossiers dans C:\wx28\lib\gcc_lib.

Voilà, c'est presque terminé.
Il ne reste plus qu'un tout petit détail, mais qui a son importance : les noms des fichiers libs.
En effet, ils ne respectent pas les noms de fichiers qui se créent automatiquement lors d'une compilation de ces libs. Ils ne sont donc pas compatibles avec l'assistant nouveau projet de Code::Blocks.
Comme je ne veux pas vous faire renommer ces fichiers manuellement (et surtout comme je n'ai pas l'intention de le faire moi-même), je vous livre un petit script VBS de ma composition.
Téléchargez-le (http://x.psoud.free.fr/localfiles/Rename-Libs.vbs) et enregistrez-le dans le dossier C:\wx28\lib.
Exécutez-le une fois (un double-clic devrait faire l'affaire), et tout devrait être OK.

Les variables système

Nous avons terminé l'installation des DevPacks principaux. Il ne reste plus qu'à indiquer au système ce que nous avons fait.
Cela va se faire par l'intermédiaire des Variables système.
Pour modifier et/ou créer ces variables, il faut utiliser un petit utilitaire intégré à Windows. Il se cache dans les "Propriétés systèmes". Pour y avoir accès :

- Clic droit sur "Poste de Travail" et sélectionnez "Propriétés", ou pressez simultanément les touches [Windows] + [Pause].
- Sélectionnez l'onglet "Avancé" : vous y trouverez un bouton donnant l'accès à l'utilitaire de gestion des "Variables d'environnement".

Dans la boite de dialogue qui apparait, vous trouverez deux listes accompagnées de boutons pour créer, modifier, supprimer les variables. La première liste ne concerne que l'utilisateur en cours (en l'occurrence, vous). La deuxième

concerne le système en entier.
Nous allons nous occuper uniquement des variables du système (liste inférieure).

- Créez la variable système **WXWIN** (bouton “Nouveau”) et donnez-lui comme valeur le chemin complet du répertoire d'installation (`C:\wx28`).
- Ensuite, repérez la variable **PATH** (toujours dans la liste inférieure), sélectionnez-la, et pressez le bouton “Modifier”. A la fin de la valeur de cette variable, ajoutez un “**;**”, suivi du chemin vers le répertoire `bin` de notre arborescence. Comme ce répertoire est un sous-répertoire de celui pointé par la variable **WXWIN**, vous pouvez mettre : **%WXWIN%\bin**. Ainsi, quel que soit votre répertoire d'installation, ce chemin sera toujours le bon (et lors d'une éventuelle prochaine installation dans un autre répertoire, il vous suffira de modifier la variable **WXWIN** pour que celle-ci soit correcte).

Et bien voilà, c'est terminé ! Vous pouvez désormais utiliser l'assistant de Code::Blocks pour créer vos projets wxWidgets.
Attention toutefois à sélectionner les options compatibles avec les libs que nous venons d'installer:

- Ne pas cocher l'option ”*wxWidgets is build as a monolithic library*” car ce n'est pas le cas.
- Ne pas cocher l'option ”*Use the debugging libraries*” car les libs n'ont pas été compilées avec l'option **DEBUG**.

Ne tenez pas compte de l'éventuel message vous indiquant qu'aucune configuration “debug” n'a pu être trouvée dans le répertoire wxWidgets spécifié.
Cette installation est également compatible avec l’assistant projet wxWidgets présent sur www.wxdev.fr (accessible aux membres enregistrés via le menu « outils »).

les DevPaks « Add-On »

Vous l’avez sans doute remarqué, FredCL propose également sur son site d’autres DevPaks contenant des composants additionnels :

- **wxPropertyGrid** : une grille à deux colonnes spécialement étudiée pour l’édition des propriétés d’un élément.
- **wxSQLite3** : un composant permettant l’accès aux bases de données SQLite3.
- **wxChart** : un composant pour créer des graphes (barres, lignes, camenberts, …).
- **wxMathPlot** : un composant pour tracer des courbes mathématiques.
- **wxStickyMap** : un composant permettant d’afficher et déplacer des images sur un fond (image ou autre).

Nous allons voir comment intégrer ces composants à notre installation wxWidgets.
Comme la méthode est quasiment la même pour chaque “Add-On” et que vous ne voulez pas forcément tous les installer, je vais redonner à chaque fois la méthode complète. Ne vous étonnez donc pas si vous trouvez un grand nombre de répétitions (vous allez voir : je suis un grand adepte du copier-coller).

wxPropertyGrid

L'installation

Il vous faudra télécharger le DevPak intitulé “wxPropertyGrid Entête et Exemples”, ainsi qu’au minimum un des quatre autres (téléchargez-les et installez-les tous, vous choisirez la version la plus appropriée lors de la création de votre projet).
Placez tous ces fichiers dans le répertoire `C:\wx28\devpacks` que vous aurez au préalable vidé de son contenu afin d’éviter les mélanges avec d’autres libs.
La méthode de décompression est toujours la même :

- Il faut renommer les fichiers ”**.devPak*” en ”**.tar.bz2*”.
- Il faut ensuite en extraire le contenu grâce à WinRar ou 7Zip (pour ceux qui auraient déjà oublié la méthode à suivre, rendez-vous un peu plus haut, dans la partie ”*Extraire le contenu des DevPaks*”).

- Au final, vous obtenez 3 dossiers : `packinc_help`, `Ansi` et `Unicode`.

Nous allons maintenant intégrer les fichiers extraits à notre installation wxWidgets.
Déplacez tous les fichiers "*.dll*" présents dans ces quatre sous-répertoires afin de les mettre dans le dossier `C:\wx28\bin` (ils sont au nombre de 2) :

- `\Ansi\packdynamic\lib\msw-ansi-2.8.8\Dynamic\wxmsw28_propgrid_gcc_custom.dll`
- `\Unicode\packdynamic\lib\msw-unicode-2.8.8\Dynamic\wxmsw28u_propgrid_gcc_custom.dll`

Maintenant, c'est au tour des fichiers headers :

- Déplacez le dossier `propgrid` qui se trouve dans `\packinc_help\include\wxWidgets-2.8.8\wx` et placez-le dans le dossier `C:\wx28\include\wx`.

Ensuite, il reste à déplacer et renommer les ficher libs (on pourrait se passer de les renommer, mais tant qu'à faire, autant avoir une installation homogène) :

- Déplacez le fichier `libwx_msw_propgrid-2.8.dll.a` qui se trouve dans le dossier `\Ansi\packdynamic\lib\msw-ansi-2.8.8\Dynamic`, placez-le dans le dossier `C:\wx28\lib\gcc_dll` et renommez-le en `libwxmsw28_propgrid.a`.
- Déplacez le fichier `libwx_msw_propgrid-2.8.a` qui se trouve dans le dossier `\Ansi\packstatic\lib\msw-ansi-2.8.8\Static`, placez-le dans le dossier `C:\wx28\lib\gcc_lib` et renommez-le en `libwxmsw28_propgrid.a`.
- Déplacez le fichier `libwx_mswu_propgrid-2.8.dll.a` qui se trouve dans le dossier `\Unicode\packdynamic\lib\msw-unicode-2.8.8\Dynamic`, placez-le dans le dossier `C:\wx28\lib\gcc_dll` et renommez-le en `libwxmsw28u_propgrid.a`.
- Déplacez le fichier `libwx_mswu_propgrid-2.8.a` qui se trouve dans le dossier `\Unicode\packstatic\lib\msw-unicode-2.8.8\Static`, placez-le dans le dossier `C:\wx28\lib\gcc_lib` et renommez-le en `libwxmsw28u_propgrid.a`.

Test wxPropertyGrid

Voilà, l'installation est terminée. Il ne nous reste plus qu'à tester si tout fonctionne correctement.
Nous allons faire quelque chose d'assez compliqué, tout en restant simple dans les manipulations à effectuer. Vous n'avez rien compris à ce que j'ai voulu dire ? Ce n'est pas grave : vous allez vite comprendre.
Regardez le nom du premier DevPak que je vous ai fait télécharger pour installer cet Add-On. Il se nomme "wxPropertyGrid Entête et Exemples"". Vous saisissez ? Nous allons tout simplement réutiliser les fichiers de l'exemple qu'il contient.
Tout d'abord, créez un nouveau projet avec Code::Blocks en utilisant l'assistant prévu à cet effet (menu "File" → "New" → "Project", sélectionnez "wxWidgets" dans la boite de dialogue qui apparaît).
Voici les options à sélectionner tout au long de l'assistant :

- Version des libs : **wxWidgets 2.8.x**.
- Titre du projet : **wxPropertyGridTest** (ou ce que vous voudrez).
- Nom de l'auteur, e-mail et site web : c'est vous qui voyez (vous pouvez laisser vide).
- Constructeur d'interface : **None**.
- Type d'application : peu importe.
- Emplacement des libs wxWidgets : **C:\wx28** (si vous avez suivi ce tutoriel à la lettre, sinon, adaptez).
- Compilateur : **GNU GCC Compiler** (c'est avec celui-ci que les libs ont été compilées).
- Vous pouvez cocher la case "*Create 'Debug' configuration*" si vous voulez, mais on ne s'en servira pas spécialement. Par contre, pensez bien à cocher la case "*Create 'Release' configuration*".
- Pour les valeurs de "*Output dir*", j'ai l'habitude de mettre "**.**" afin que l'exécutable soit créé directement dans le dossier du projet.
- Pour les valeurs de "*Objects output dir*", je mets généralement "**obj_d**" pour la configuration "*Debug*" et "**obj_r**" pour la configuration "*Release*".

- Pour les options, je vous laisse choisir en fonction de vos habitudes de programmation, et des fichiers que vous avez téléchargés. Assurez-vous juste de **ne pas cocher** la case "*wxWidgets is built as a Monolithic library*", car ce n'est pas le cas, et pensez à **sélectionner la case** "*Configure Advanced options*" afin d'avoir accès à la page optionnelle qui suit.
- Comme nous allons utiliser les fichiers de l'exemple, il n'est pas nécessaire de laisser l'assistant créer le code pour nous. Vérifiez bien que l'option "*Create empty project*" soit **cochée**.
- Lors du passage à la page suivante, vous devriez normalement obtenir un avertissement vous disant qu'une configuration debug n'a pu être trouvée dans le répertoire wxWidgets spécifié. N'en tenez pas compte, et cliquez sur "*Oui*" pour continuer.
- Ne **sélectionnez pas** la case "*Use WXDEBUG and Debug wxWidgets lib*", car les libs n'ont pas été compilées avec cette option.
- Comme l'Add-On à tester est un contrôle, nous allons obligatoirement créer une application avec interface graphique. Il faut donc sélectionner l'option "**GUI Mode Application**" pour les deux configurations.
- Enfin, sur la dernière page, ne sélectionnez rien, car de toute façon, les libs wxPropertyGrid ne sont pas dans la liste des éléments proposés.

Lorsque l'assistant a terminé la création du projet, nous allons y ajouter les fichiers contenant le code qui va nous permettre de tester notre installation.
Vous trouverez dans dossier `C:\wx28\devpacks\packinc_help\Examples\wx\contrib\propgrid` les 5 fichiers qui nous intéressent (inutile d'aller voir dans les sous-dossiers, ils contiennent les fichiers projets pour DevC++).
Copiez les 5 fichiers ("`propgridsample.h`", "`propgridsample.cpp`", "`propgridsample.rc`", "`sampleprops.h`" et "`sampleprops.cpp`") dans le dossier de votre nouveau projet. Retournez sous Code::Blocks et ajoutez ces fichiers au projet (menu "Project" → "Add files…"). Faites en sorte, quand on vous le demande, que ces fichiers soient ajoutés aux deux configurations "*Debug*" et "*Release*").
Il ne nous reste maintenant plus qu'à ajouter les libs wxPropertyGrid dans les options du linker :

- Ouvrez la boite de dialogue d'options du projet (menu "Project" → "Build options")
- Pour chaque configuration dans la liste de gauche (*debug*, *release*) sélectionnez l'onglet "*linker settings*" et ajoutez, dans la liste "*Link libraries*", la lib `libwxmsw28u_propgrid.a` (si vous travaillez en Unicode) ou `libwxmsw28_propgrid.a` (si vous êtes en Ansi).

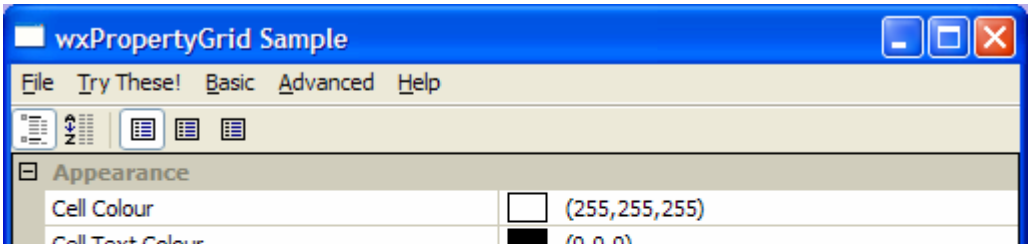
Si vous tentez une compilation maintenant, vous devriez normalement obtenir une erreur relative à une ambigüité au niveau du code (essayez, vous verrez).
La solution à ce problème est tout simplement de modifier la ligne *2261* du fichier `propgridsample.cpp` afin de lever cette ambigüité.
Il faut remplacer la ligne :

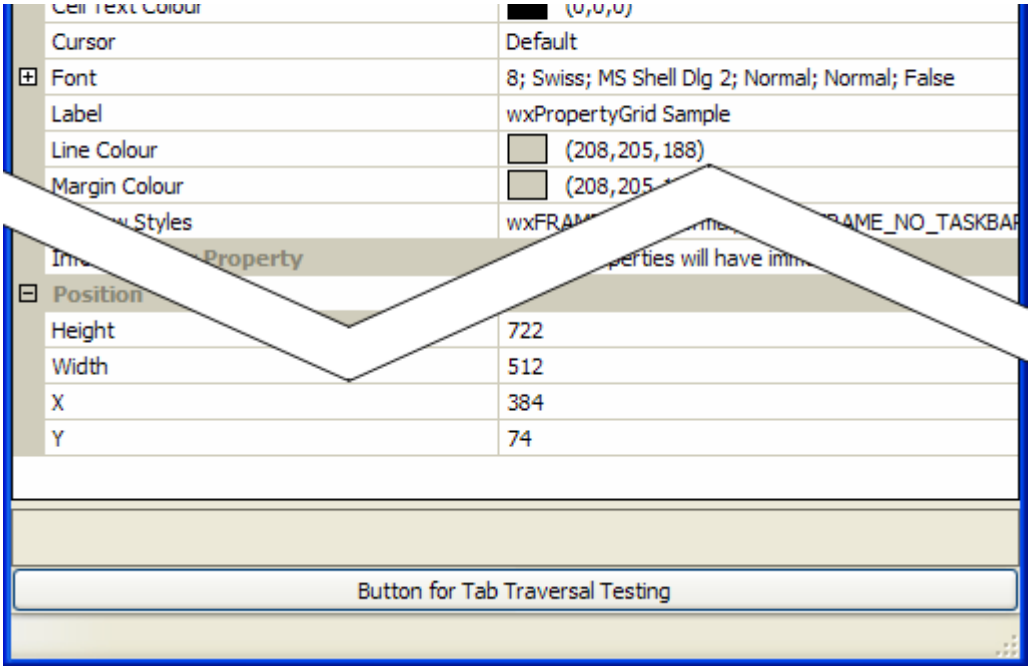
```
2261        id = pg->GetNextSibling(id);
```

par la ligne :

```
2261        id = ((wxPropertyContainerMethods*)pg)->GetNextSibling(id);
```

Voilà, c'est tout. Vous pouvez maintenant compiler et exécuter votre application, tout devrait marcher comme sur des roulettes.
Voici ce que ça donne chez moi :





wxSQLite3

L'installation

Il vous faudra télécharger le DevPak intitulé “wxSQLite3 Entête et Exemples”, ainsi qu’au minimum un des quatre autres (téléchargez-les et installez-les tous, vous choisirez la version la plus appropriée lors de la création de votre projet).
Vous aurez également besoin du DevPak intitulé “SQLite 3.5.9” (que vous pourrez retrouver un peu plus bas sur la page de téléchargement).
Placez tous ces fichiers dans le répertoire `C:\wx28\devpacks` que vous aurez au préalable vidé de son contenu afin d’éviter les mélanges avec d’autres libs.

- La méthode de décompression est toujours le même : il faut renommer les fichiers ”**.devPak*” en ”**.tar.bz2*”.
- Il faut ensuite en extraire le contenu grâce à WinRar ou 7Zip (pour ceux qui auraient déjà oublié la méthode à suivre, rendez-vous un peu plus haut, dans la partie ”*Extraire le contenu des DevPaks*”).
- Au final, vous obtenez 4 dossiers : `pack`, `packinc_help`, `Ansi` et `Unicode`.

Nous allons maintenant intégrer les fichiers extraits à notre installation wxWidgets.
Déplacez tous les fichiers ”**.dll*” présents dans ces quatre sous-répertoires afin de les mettre dans le dossier `C:\wx28\bin` (ils sont au nombre de 4) :

- `\pack\lib\sqlite-3.5.9\fts3-3.5.9.dll`
- `\pack\lib\sqlite-3.5.9\sqlite3.5.9.dll`
- `\Ansi\packdynamic\lib\msw-ansi-2.8.8\Dynamic\wxmsw28_sqlite_gcc_custom.dll`
- `\Unicode\packdynamic\lib\msw-unicode-2.8.8\Dynamic\wxmsw28u_sqlite_gcc_custom.dll`

Maintenant, c’est au tour des fichiers headers :

- Déplacez le dossier `sqlite3` qui se trouve dans `\packinc_help\include\wxWidgets-2.8.8\wx` et placez-le dans le dossier `C:\wx28\include\wx`.
- Déplacez le fichier `\pack\include\sqlite-3.5.9\sqlite3.h` et placez-le dans le dossier `C:\wx28\include`.

Ensuite, il reste à déplacer et renommer les ficher libs (on pourrait se passer de les renommer, mais tant qu’à faire, autant avoir une installation homogène) :

- Déplacez le fichier `libwx_msw_sqlite-2.8.dll.a` qui se trouve dans le dossier `\Ansi\packdynamic\lib\msw-ansi-2.8.8\Dynamic`, placez-le dans le dossier `C:\wx28\lib\gcc_dll` et renommez-le en `libwxmsw28_sqlite.a`.
- Déplacez le fichier `libwx_msw_sqlite-2.8.a` qui se trouve dans le dossier `\Ansi\packstatic\lib\msw-ansi-2.8.8\Static`, placez-le dans le dossier `C:\wx28\lib\gcc_lib` et renommez-le en `libwxmsw28_sqlite.a`.
- Déplacez le fichier `libwx_mswu_sqlite-2.8.dll.a` qui se trouve dans le dossier `\Unicode\packdynamic\lib\msw-unicode-2.8.8\Dynamic`, placez-le dans le dossier `C:\wx28\lib\gcc_dll` et renommez-le en `libwxmsw28u_sqlite.a`.
- Déplacez le fichier `libwx_mswu_sqlite-2.8.a` qui se trouve dans le dossier `\Unicode\packstatic\lib\msw-unicode-2.8.8\Static`, placez-le dans le dossier `C:\wx28\lib\gcc_lib` et renommez-le en `libwxmsw28u_sqlite.a`.
- Déplacez le fichier `libsqlite3.5.9-dy.a` qui se trouve dans le dossier `\pack\lib\sqlite-3.5.9`, placez-le dans le dossier `C:\wx28\lib\gcc_dll` et renommez-le en `libsqlite3.5.9.a` (supprimez le "*-dy*").
- Déplacez le fichier `libsqlite3.5.9-st.a` qui se trouve dans le dossier `\pack\lib\sqlite-3.5.9`, placez-le dans le dossier `C:\wx28\lib\gcc_lib` et renommez-le en `libsqlite3.5.9.a` (supprimez le "*-st*").
- Déplacez le fichier `libfts3-3.5.9.a` qui se trouve dans le dossier `\pack\lib\sqlite-3.5.9`, placez-le dans le dossier `C:\wx28\lib\gcc_lib` et placez-en une copie dans le dossier `C:\wx28\lib\gcc_dll`.

Voilà, l'installation est terminée. Il ne nous reste plus qu'à tester si tout fonctionne correctement.

Test wxSQLite

Nous allons faire une petite application qui va créer une base de données contenant 2 tables :

- une table "*auteurs*" contenant un ID et un nom.
- une table "*livres*" contenant un ID, un titre, et l'ID de l'auteur.

Ensuite, cette application va placer quelques valeurs dans ces deux tables et enfin, faire le listing des livres en y associant leurs auteurs.
Lancez Code::Blocks, démarrez l'assistant "Nouveau projet", et sélectionnez un projet wxWidgets. Voici les options à sélectionner :

- Version des libs : **wxWidgets 2.8.x**.
- Titre du projet : **wxSQLiteTest** (ou ce que vous voudrez).
- Nom de l'auteur, e-mail et site web : c'est vous qui voyez (vous pouvez laisser vide).
- Constructeur d'interface : **None** (l'interface sera très simple, vous verrez).
- Type d'application : peu importe
- Emplacement des libs wxWidgets : **C:\wx28** (si vous avez suivi ce tutoriel à la lettre).
- Compilateur : **GNU GCC Compiler** (c'est avec celui-ci que les libs ont été compilées).
- Vous pouvez cocher la case "*Create 'Debug' configuration*" si vous voulez, mais on ne s'en servira pas spécialement. Par contre, pensez bien à cocher la case "*Create 'Release' configuration*".
- Pour les valeurs de "*Output dir*", j'ai l'habitude de mettre "**.**" afin que l'exécutable soit créé directement dans le dossier du projet.
- Pour les valeurs "*Objects output dir*", je mets généralement "**obj_d**" pour la configuration "*Debug*" et "**obj_r**" pour la configuration "*Release*".
- Pour les options, je vous laisse choisir en fonction de vos habitudes de programmation, et des fichiers que vous avez téléchargés. Assurez-vous juste de **ne pas cocher** la case "*wxWidgets is built as a Monolithic library*", car ce n'est pas le cas, et pensez à **sélectionner la case** "*Configure Advanced options*" afin d'avoir accès à la page optionnelle qui suit.
- Nous allons en fait créer une simple application "*console*" : il n'est donc pas nécessaire de laisser l'assistant créer le code pour nous. Vérifiez bien que l'option "*Create empty project*" soit **cochée**.

- Lors du passage à la page suivante, vous devriez normalement obtenir un avertissement vous disant qu'une configuration debug n'a pu être trouvée dans le répertoire wxWidgets spécifié. N'en tenez pas compte, et cliquez sur "*Oui*" pour continuer.
- Ne **sélectionnez pas** la case "*Use WXDEBUG and Debug wxWidgets lib*", car les libs n'ont pas été compilées avec cette option.
- Comme je vous l'ai dit, nous allons faire une simple application console. Il faut donc **sélectionner** l'option "*Console Mode Application*" pour les deux configurations.
- Enfin, sur la dernière page, ne sélectionnez rien, car de toute façon, les libs wxSQLite ne sont pas dans la liste des éléments proposés.

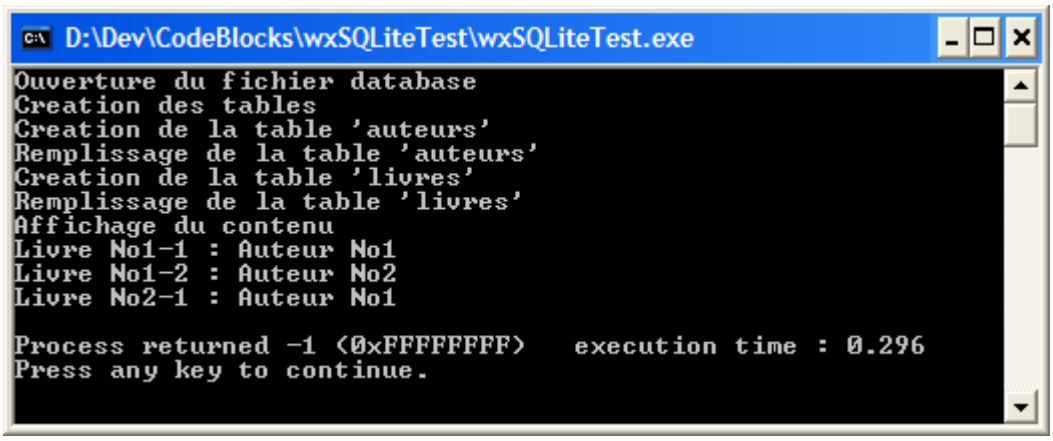
Lorsque l'assistant a terminé la création du projet, nous allons y ajouter l'unique fichier contenant le code qui va nous permettre de tester notre installation.
Ajoutez un fichier vide (menu "*File*", "*New*", "*Empty file*") en répondant "*Oui*" quand il vous est demandé si vous voulez l'ajouter au projet courant, enregistrez-le (je l'appelle `main.cpp`) et faites en sorte qu'il soit ajouté à toutes les configurations disponibles.
Voici le code à insérer dans ce fichier. Je pense qu'il est suffisamment commenté, mais si vous estimez que ce n'est pas le cas, n'hésitez pas à pousser une gueulante sur le forum de www.wxdev.fr .

```
#include <wx/wx.h>
#include "wx/sqlite3/wxsqlite3.h"
class MyApp : public wxApp
{
public:
    virtual bool OnInit();
private:
    wxSQLite3Database mydb;
    bool OpenDatabase(const wxString &filename);
    bool CreateTables();
    bool ShowList();
};

IMPLEMENT_APP(MyApp);

bool MyApp::OnInit()
{
    // On essaye d'ouvrir ou de créer le fichier "database.db"
    wxPuts(_T("Ouverture du fichier database"));
    if (!OpenDatabase(_T("database.db"))){ wxPuts(_T("Erreur 1")); return false;}
    // Les tables existent-t'elles (dans le cas où le fichier existait déjà)
    if (!mydb.TableExists(_T("auteurs")))
    {
        wxPuts(_T("Creation des tables"));
        if (!CreateTables()) { wxPuts(_T("Erreur 2")); return false;}
    }
    // Maintenant, on liste les livres et leurs auteurs
    wxPuts(_T("Affichage du contenu"));
    if (!ShowList()) { wxPuts(_T("Erreur 3")); return false;}
    // On renvoie false sinon, l'application boucle malgré l'absence de fenêtre
    return false;
}
bool MyApp::OpenDatabase(const wxString &filename)
{
    try
    {
        // Ouverture (ou création) de la base de données
        mydb.Open(filename);
        return true;
    }
    catch (wxSQLite3Exception& e)
    {
        wxPuts(_T("Erreur lors de l'ouverture du fichier database."));
        wxPrintf(_T("t%d : %s"),e.GetErrorCode(),e.GetMessage().c_str());
        return false;
    }
}

bool MyApp::CreateTables()
{
```

Il ne nous reste maintenant plus qu'à ajouter les libs wxSQLite3 dans les options du linker :

- Ouvrez la boite de dialogue d'options du projet (menu "*Project*", "*Build options*")
- Pour chaque configuration dans la liste de gauche (debug, release) sélectionnez l'onglet "*linker settings*" et ajoutez, dans la liste "*Link libraries*", la lib `libwxmsw28u_sqlite.a` (si vous travaillez en Unicode) ou `libwxmsw28_sqlite.a` (si vous êtes en Ansi).

Voilà, c'est tout. Vous pouvez maintenant compiler et exécuter votre application, tout devrait marcher comme sur des roulettes.
Voici ce que ça donne chez moi:



Il se peut que la fenêtre *console* disparaisse instantanément (sans que vous ayez le temps de voir le résultat). Pour remédier à ce problème (et indiquer à Code::Blocks d'ajouter une commande "*pause*" à la fin de l'exécution), procédez de la façon suivante :

- Ouvrez la boite de dialogue de propriétés du projet (menu "*Project*", "*Properties*").
- Sélectionnez l'onglet "*Build targets*".
- Pour chaque entrée dans la liste de gauche (debug, release), assurez-vous que l'option "*Pause when execution ends*" soit cochée.

Pour information, le code de retour (-1) est normalement un code d'erreur (et est considéré comme tel par Code::Blocks).
Il est dû au fait que la fonction `MyApp::OnInit()` renvoie `false` (les commentaires du code vous expliquent pourquoi).

Conclusion

Voilà, ce tutoriel est terminé.
Vous pouvez supprimer le dossier `devpacks` ainsi que tout son contenu, et profiter de votre nouvelle version des libs wxWidgets.
Encore une fois (on ne le répètera jamais assez), merci à FredCL pour ses DevPacks ! N'hésitez pas à laisser un petit mot sur le forum ou dans le livre d'or de son site : http://cfred.free.fr
Bonne prog, et @ +
Xav'