
Créer un projet wxWidgets a l'aide de Bloodshed Dev-C++ sous Microsoft Windows

Nowicki Christophe <nowick_c@epita.fr>

Copyright © 2003 Nowicki Christophe

Permission est accordée de copier, distribuer et/ou modifier ce document selon les termes de la Licence de Documentation Libre GNU (GNU Free Documentation License), version 1.1 ou toute version ultérieure publiée par la Free Software Foundation. Pas de section inaltérable.

	Historique des versions
Version 1.1	26/02/2004
Version 1.0	s/wxWindows/wxWidgets/g ;(27/10/2003 Version initiale

Table des matières

Télécharger les outils	1
Bloodshed Dev-C++	1
wxWidgets	2
Installation	3
Bloodshed Dev-C++	3
Les DevPacks wxWidgets	3
Création d'un projet d'interface graphique basique	3
"Hello wxWidgets!"	3
Analyse du code source	5
Conclusion	8

Résumé

Cet article décrit étape par étape l'installation, la configuration et la compilation d'un projet Hello World! à l'aide de l'environnement de développement intégré (ou IDE) Bloodshed Dev-C++ et de l'interface de programmation d'applications (ou API) multiplateforme wxWidgets sous Microsoft Windows.

Télécharger les outils

Bloodshed Dev-C++

Bloodshed Dev-C++ est libre et écrit en Delphi pour Microsoft Windows. Il utilise le portage Mingw des outils GNU dont le fameux GCC (GNU Compiler Collection). Il dispose des fonctionnalités suivantes :

- Débbugger intégré
- Gestion de projet
- Colorisation syntaxique personnalisable
- Navigateur de classe C++
- Profiler
- Création de fichiers Makefile
- Création de modèles pour ses propres projets
- Editer et compiler les ressources
- Support de CVS (Concurrent Versions System)

Il est possible de le télécharger librement sur le site de Bloodshed Software [<http://www.bloodshed.net/devcpp.html>]. La version actuelle est la 5.0 beta 8 (4.9.8.0).

wxWidgets

wxWidgets est une API C++ libre multiplateforme (Unix, Windows et Mac). Elle a l'énorme avantage d'être sous License LGPL (acronyme de Library General Public Licence) et donc permettre la redistribution de binaires. De plus, elle utilise l'interface graphique native sur chaque système d'exploitation.

Voila la liste des avantages offerts par wxWidgets :

- Libre
- Orienté objet
- Portable (Windows, Unix et Mac)
- Fonctionne avec la plupart des compilateurs C++
- Une interface Python (wxPython)
- Bien documenté

Si nous voulez avoir un aperçu de toutes les possibilités de cette API je vous propose de vous référer aux documents officiels :

- FAQ Général [<http://www.wxWidgets.org/faqgen.htm>]
- FAQ Microsoft Windows [<http://www.wxWidgets.org/faqmsw.htm>]
- Documentation [<http://www.wxWidgets.org/docs.htm>]

Pour l'installation de wxWidgets vous avez le choix entre : recompiler les sources a l'aide de Dev-C++

ou bien télécharger les DevPacks. Les DevPacks sont des modules pour Dev-C++ qui contiennent les bibliothèques wxWidgets compilées et les modèles des projets d'applications qui utilisent wxWidgets. Les DevPacks wxWidgets pour Dev-C++ sont téléchargeables sur la page personnelle de Michel Weinachter :

Les "DevPacks" wxWidgets pour Dev-C++ [<http://michel.weinachter.free.fr/>]

Vous devez télécharger les trois DevPacks suivants :

- wxWidgets 2.4.0
- imagelib
- wxWidgets 2.4.0 contribs

Installation

Bloodshed Dev-C++

L'installation de Dev-C++ ne pose aucun problème. Il suffit de choisir une installation dite Typical et choisir le répertoire d'installation. Une fois l'installation terminée vous pouvez choisir votre thème et la langue. Vous pouvez choisir anglais ou bien français.

Les DevPacks wxWidgets

Pour installer les DevPacks il suffit de cliquer sur :

Anglais	Français
Tools->Package Manager	Outils->Package Manager

Puis d'installer les DevPacks dans l'ordre suivant : imagelib, wxWidgets, wxWidgets-contrib.

Création d'un projet d'interface graphique basique

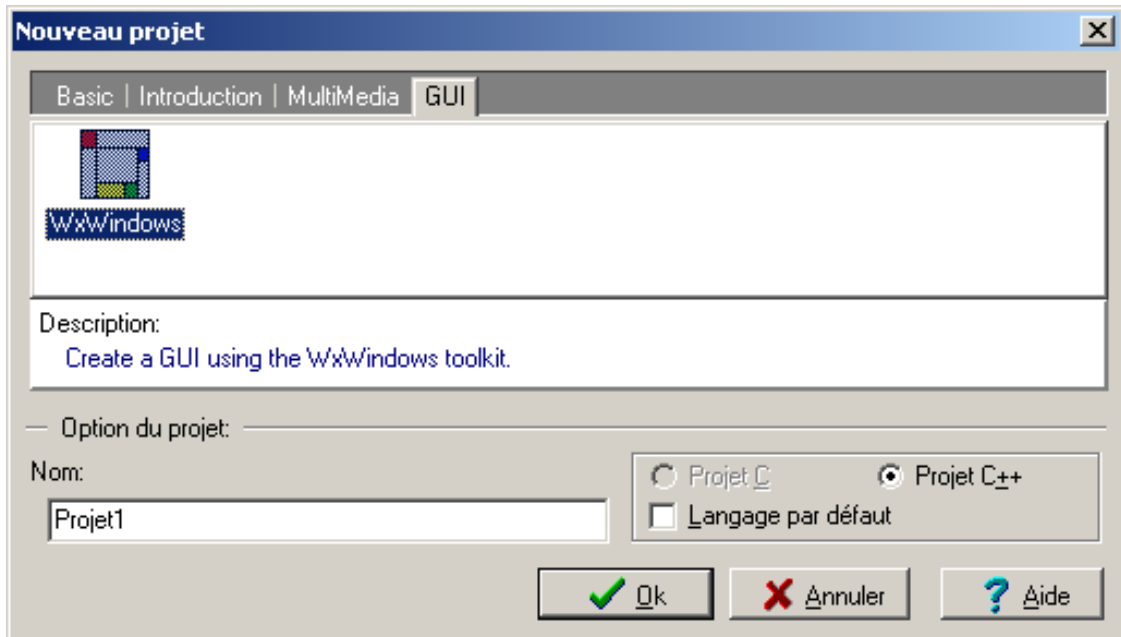
"Hello wxWidgets!"

Une fois les trois DevPacks installés il suffit de cliquer sur :

Anglais	Français
File->New->Project...	Fichier->Nouveau->Projet...

Puis sur :

GUI->wxWidgets



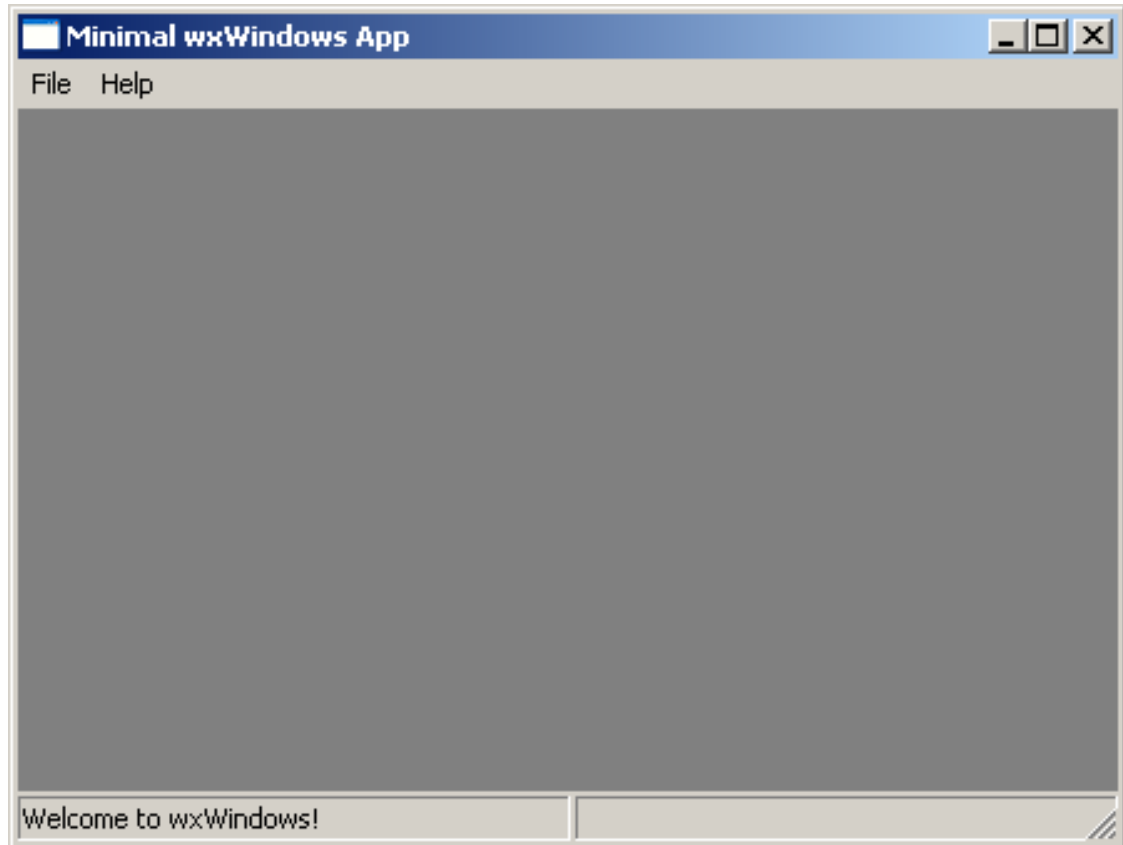
Il ne reste plus qu'à choisir le nom de votre nouveau projet et de le sauvegarder dans un répertoire de travail. Une fois le projet créé vous devez voir un fichier main.cpp avec le code d'une application wxWidgets minimal. Il faut sauvegarder le fichier :

Anglais	Français
File->Save (Ctrl-S)	Fichier->Sauvegarder (Ctrl-S)

Il ne vous reste plus qu'à compiler et exécuter celle-ci :

Anglais	Français
Execute->Compile & Run (F9)	Executer->Compiler & Executer (F9)

Si la compilation s'est bien passée vous devez voir apparaître la fenêtre suivante :



Analyse du code source

Nous allons analyser le code généré automatiquement par Dev-C++. Tout d'abord il faut inclure les fichiers d'entête de wxWidgets. Vous avez le choix entre inclure un gros fichier qui contient toutes les class wxWidgets (`#include "wx/wx.h"`) ou bien inclure seulement les class dont votre application a besoin (ex : `#include "wx/window.h"` et `#include "wx/menu.h"`). Pour les compilateurs qui supportent les fichiers d'entête précompilés il est possible d'utiliser la version précompilé. (`#include "wx/wxprec.h"`)

```
// For compilers that support precompilation, includes "wx/wx.h".
#include "wx/wxprec.h"

#ifdef __BORLANDC__
    #pragma hdrstop
#endif

#ifndef WX_PRECOMP
    #include "wx/wx.h"
#endif
```

Chaque application définit une classe dérivée de la classe wxApp. En utilisant la méthode virtuelle OnInit() de la classe wxApp, on peut initialiser notre application. Créer la fenêtre principale par exemple.

```
class MyApp: public wxApp
{
    virtual bool OnInit();
};
```

Pour créer la fenêtre principale de l'application il suffit d'utiliser la même méthode : dériver une classe de la classe wxFrame. Dans le constructeur de cette nouvelle classe vous pouvez créer un menu et une barre des statuts. Cette fenêtre doit pouvoir répondre aux évènements envoyés par l'utilisateur (comme le bouton de la souris, les menus ou les boutons). Pour répondre à ces évènements, la classe doit contenir une table d'évènement (DECLARE_EVENT_TABLE).

```
class MyFrame: public wxFrame
{
public:
    MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size);

    void OnQuit(wxCommandEvent& event);
    void OnAbout(wxCommandEvent& event);

private:
    DECLARE_EVENT_TABLE()
};
```

Pour chaque évènement il faut donner un identifiant unique, à l'aide d'une constante ou bien d'un enum.

```
enum
{
    ID_Quit = 1,
    ID_About,
};
```

Il faut implémenter les handler des différents évènements de la classe wxFrame à l'aide de macros. Il n'a pas besoin de gérer tous les évènements que peut recevoir la classe car il y a un handler d'évènement générique par défaut pour chaque évènement.

```
BEGIN_EVENT_TABLE(MyFrame, wxFrame)
    EVT_MENU(ID_Quit, MyFrame::OnQuit)
    EVT_MENU(ID_About, MyFrame::OnAbout)
END_EVENT_TABLE()
```

Tous les programmes doivent avoir une fonction d'entrée principale (main). Celle-ci est implémentée à l'aide d'une macro qui crée une instance de l'objet qui représente votre application.

```
IMPLEMENT_APP(MyApp)
```

Après la création de l'instance de wxApp la méthode OnInit() est appelée. Vous pouvez l'utiliser cette méthode pour initialiser le programme, afficher un "splash screen" et créer la ou les fenêtres principales.

```
bool MyApp::OnInit()
{
    MyFrame *frame = new MyFrame( "Hello World", wxPoint(50,50), wxSize(450,340) )
    frame->Show( TRUE );
    SetTopWindow( frame );
    return TRUE;
}
```

Dans ce constructeur de la fenêtre principale nous pouvons créer un menu et une barre de statuts.

```
MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size)
    : wxFrame((wxFrame *)NULL, -1, title, pos, size)
{
    wxMenu *menuFile = new wxMenu;

    menuFile->Append( ID_About, "&About..." );
    menuFile->AppendSeparator();
    menuFile->Append( ID_Quit, "E&xit" );

    wxMenuBar *menuBar = new wxMenuBar;
    menuBar->Append( menuFile, "&File" );

    SetMenuBar( menuBar );

    CreateStatusBar();
    SetStatusText( "Welcome to wxWidgets!" );
}
```

Il nous reste plus qu'à écrire les méthodes pour la gestion des événements. MyFrame::OnQuit() ferme la fenêtre, s'il n'y a pas d'autres fenêtres ouvertes par l'application, celle-ci se termine automatiquement.

```
void MyFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
{
    Close(TRUE);
}
```

MyFrame::OnAbout() va afficher un boîte de dialog avec un message dedans.

```
void MyFrame::OnAbout(wxCommandEvent& WXUNUSED(event))
{
    wxMessageBox( "This is a wxWidgets' Hello world sample",
                  "About Hello World", wxOK | wxICON_INFORMATION );
}
```

Conclusion

Vous avez réussi a faire une application wxWidgets basique. Pour mieux comprendre le fonctionnement de l'API et aller plus loin il faut se plonger dans les manuels de référence de l'API [<http://www.wxWidgets.org/manuals/>]. Mais aussi sur le forum dédié [<http://g.yi.org/Forum/list.php?f=13>] à la programmation en wxWidgets avec Dev-C++.