

TechDays 2008: Nouveautés Visual C++ 2008

par Patrick OTTAVI MVP Visual C++ ([Site](#)) ([Blog](#))

Date de publication : 05/03/2007

Dernière mise à jour : 05/03/2007

L'autre session importante pour moi lors de ces TechDays était les nouveautés de Visual C++ 2008, une session y était consacrée. voici ce que j'en ai retenu:

On peut dire que cette release améliore notablement le comportement de l'IDE.

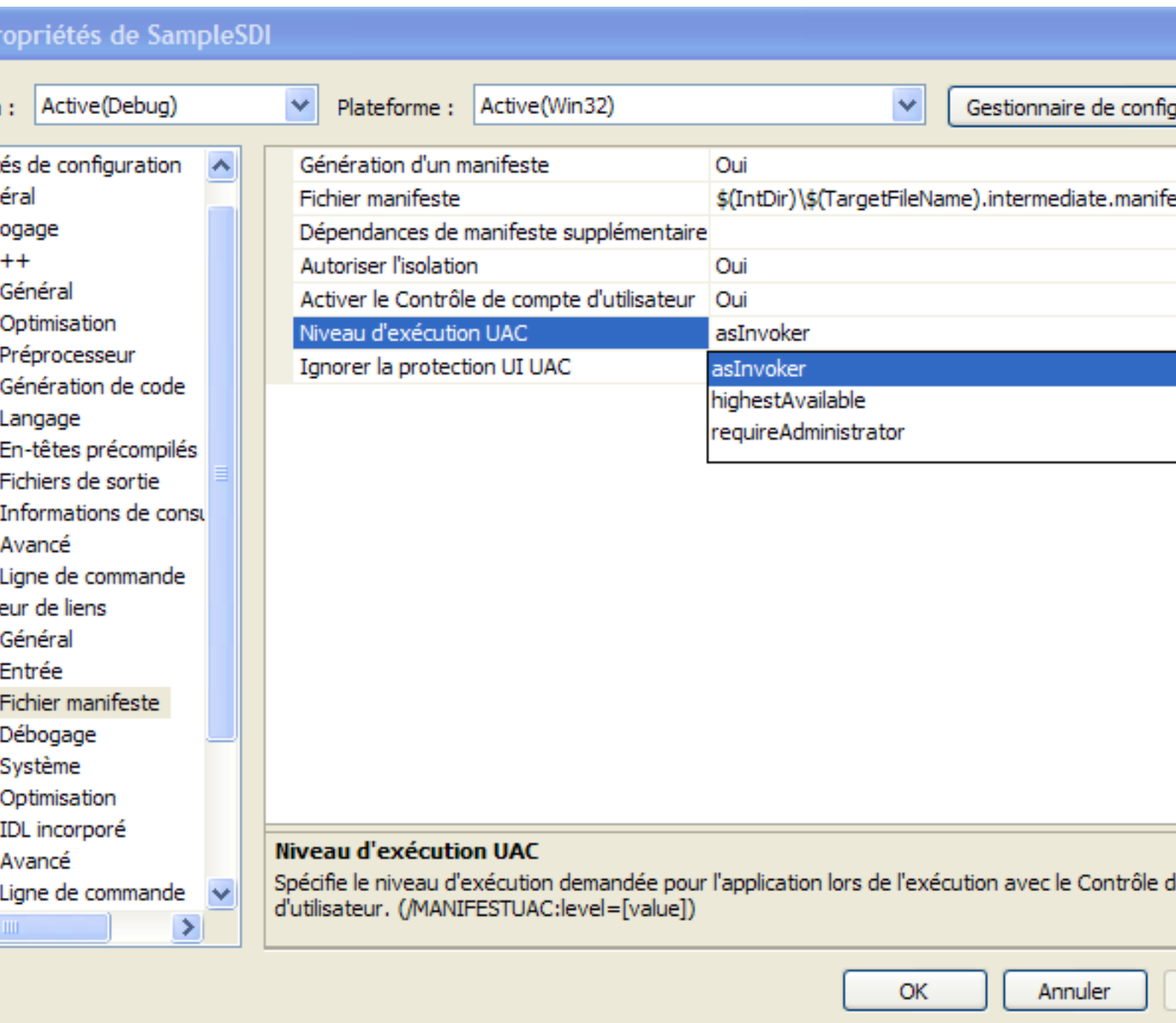
Depuis que je l'utilise en production je n'ai plus constaté de gel en ouverture ou fermeture de projet.

Ces blocages étaient provoqués en grande partie par la gestion d'IntelliSense .

Donc ce point de vue c'est une bonne nouvelle l'ide se comporte beaucoup mieux que Visual 2005.

Au niveau C++ pur dans la version commerciale actuelle pas de grandes nouveautés si ce n'est des améliorations côté environnement :

- Le compilateur prend en charge l'architecture dual core pendant la phase de compilation.**
- Intégration de la gestion du manifest UAC au niveau du projet :**



-Possibilité d'activer la redirection de l'écriture au niveau de la base de registres de la clef HKEY_CLASSES_ROOT vers HKEY_CURRENT_USER automatiquement :

Propriétés de SampleSDI

Configuration : Active(Debug) | Plateforme : Active(Win32) | Gestionnaire de configuration

| | | |
|-------------------|--|---------------------------|
| Général | Fichier de sortie | .\Debug/SampleSDI.exe |
| Entrée | Affichage de l'avancement | Non défini |
| Fichier manifeste | Version | |
| Débogage | Activation des liens incrémentiels | Oui (/INCREMENTAL) |
| Système | Suppression de la bannière de démarrage | Oui (/NOLOGO) |
| Optimisation | Bibliothèque d'importation ignorée | Non |
| IDL incorporé | Inscription de la sortie | Non |
| Avancé | Redirection par utilisateur | Non |
| Ligne de commande | Répertoires de bibliothèques supplémentaires | |
| Manifeste | Dépendances de bibliothèque de liens | Oui |
| Général | Utiliser les entrées de dépendance de bibliothèque | Non |
| Entrée et sortie | Utilisation de fichiers réponse UNICODE | Oui |

Redirection par utilisateur
si l'inscription de la sortie est activée, la redirection par utilisateur force les entrées de Registre dans HKEY_CLASSES_ROOT à être redirigées vers HKEY_CURRENT_USER.

OK Annuler

-L'option /analyze a été renforcée :

Pour rappel, cette option permet de traquer les dépassements de mémoire tampon, le déréréférencement du pointeur null, les fuites de mémoire.

L'autre aspect intéressant est la gestion des annotations permettant d'ajouter des contrôles sur les arguments d'une fonction ou son retour.

Exemple :

Dans une classe boîte de dialogue j'ai une méthode :

```
BOOL GetApplicationFolder(TCHAR *szPath);
```

Je vais rajouter des spécifications pour indiquer que le retour de ma méthode doit être utilisée et que l'argument szPath ne doit pas être nul.

```
#include <CodeAnalysis\SourceAnnotations.h>
using namespace vc_attributes;
// boîte de dialogue CTestDlg
class CTestDlg : public CDialog
{
// Construction
public:
CTestDlg(CWnd* pParent = NULL); // constructeur standard

// Données de boîte de dialogue
enum { IDD = IDD_TEST_DIALOG };

protected:
virtual void DoDataExchange(CDataExchange* pDX); // Prise en charge de DDX/DDV
[returnvalue:Post( MustCheck=SA_Yes )] BOOL GetApplicationFolder([Pre(Null=No)] TCHAR *szPath);
//.....
```

L'implémentation de la méthode :

```
BOOL CTestDlg::GetApplicationFolder(TCHAR *szPath)
{
return SHGetSpecialFolderPath(NULL,szPath,CSIDL_PROGRAM_FILES ,false);
}
```

[returnvalue:Post(MustCheck=SA_Yes)] requiert que l'appelant vérifie la valeur de retour de **GetApplicationFolder**

[Pre(Null=No)] requiert que l'appelant passe un paramètre non nul pour l'argument szPath.

L'utilisation erronée suivante (au sein de la classe):

```
TCHAR path[MAX_PATH];
GetApplicationFolder(NULL);

// de même que le code suivant provoquera les mêmes warnings...
TCHAR *szPath=NULL;
GetApplicationFolder(szPath);
```

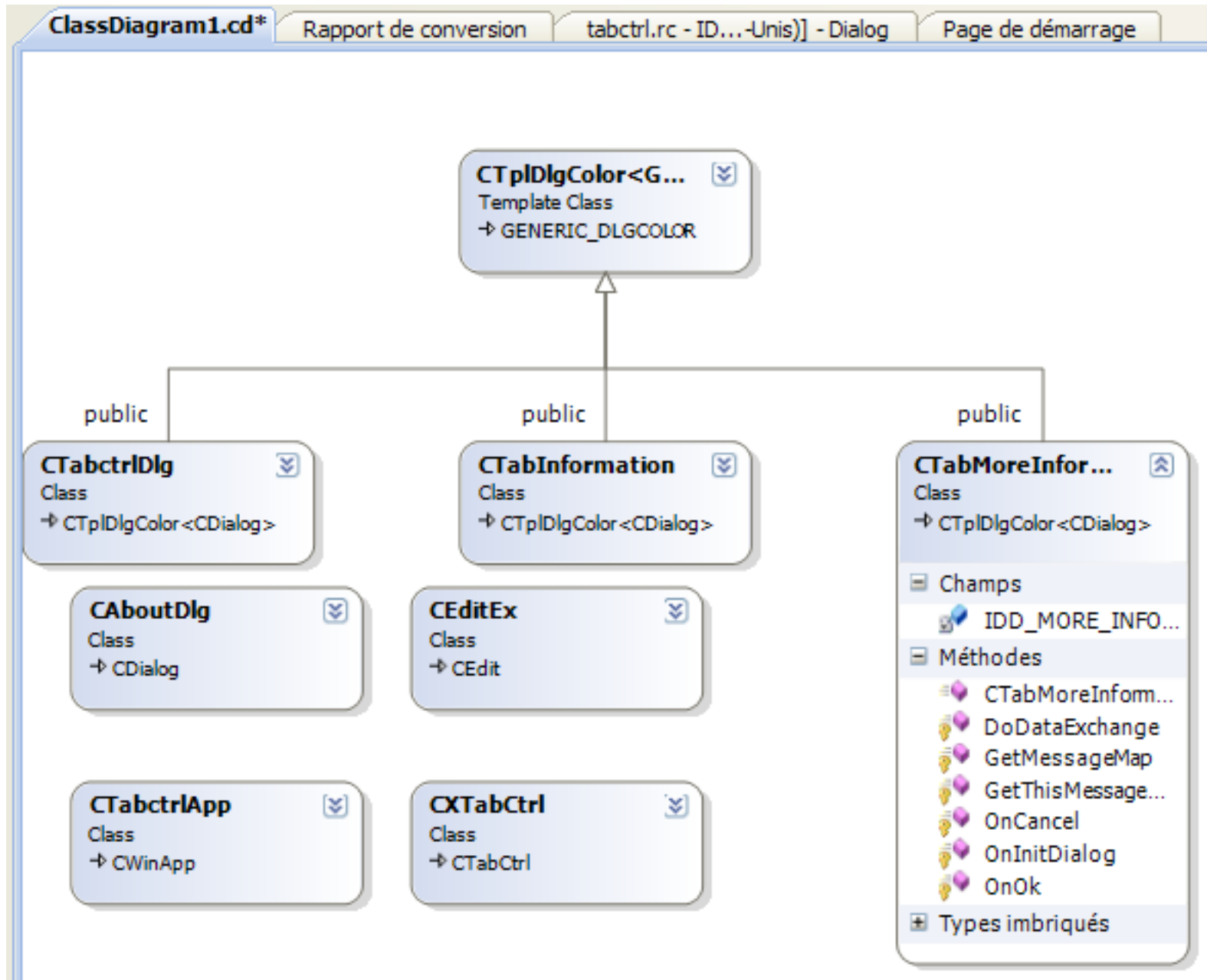
Provoquera les warnings suivants :

```
warning C6031: Valeur de retour ignorée : 'CTestDlg::GetApplicationFolder'
```

```
warning C6309: L'argument '1' est null : ceci n'est pas conforme à la spécification de fonction de
'CTestDlg::GetApplicationFolder'
```

Il existe d'autres options d'annotations pour contrôler les arguments la documentation est disponible sur MSDN.

- Le concepteur de classes est désormais pris en charge pour le code C++ natif en lecture seule uniquement.



Nouveautés sur les bibliothèques :

Comme je l'ai dit plus haut, coté C++ dans la version actuelle rien de nouveau.

Coté C++/CLI :

C++/CLI continue son évolution avec l'intégration d'une **STL/CLR**, et surtout l'ajout d'une nouvelle classe **marshal_as** cette classe permet d'écrire des templates pour simplifier l'écriture de code mixte entre C++ et C++/CLI.


Dans sa version initiale elle permet le marshaling entre chaînes de caractères pour le monde natif et managé.

les modes de conversion supportés sont les suivants:

```
- String^ vers char* / wchar_t* / BSTR / bstr_t / CComBSTR / std::string / std::wstring /  
CString<char> / CString<wchar_t> et vice versa<br/>  
- IntPtr vers HANDLE et vice versa<br/>
```

Le plus c'est que l'on peut l'étendre pour nos besoins.

Vous trouverez un exemple d'implémentation sur le blog de  **Nish**

Un blog est dédié à son enrichissement à cette adresse :  **Marshal-As.NET**

Du coté des MFC :

Une importante mise à jour est en cours, je dis en cours parce que non disponible dans le produit de base .

Elle est encore en version beta et téléchargeable sur [MSDN](#)

Attention tout de même, celle-ci n'est valable que pour la version anglaise.

Il va donc falloir attendre un petit peu pour disposer de ces nouvelles fonctionnalités.

A l'origine Microsoft a racheté la bibliothèque [BCGControlbar](#) , et l'a incluse dans les MFC.

On dispose donc de tout un ensemble de nouvelles classes (une bonne centaine) permettant de donner un look des plus modernes à nos applications.

Voici une liste partielle des possibilités # :

Look Office 2007 : barre ruban (ribbon bar) : avec ses différents éléments, statut bar etc#

Look Office 2003 (look XP): barre d'outils et de menus au style office, barre de raccourcis selon le style Outlook, aperçu d'impression, sélectionneur de fontes et de couleurs etc.

Look Internet Explorer : Rebars et panneaux de tâches (Rebars and task panes).

Look Visual Studio : fonctions d'attachements (docking), fenêtres escamotables (auto hide windows), grilles de propriétés (property grids), onglets MDI, onglets de groupes etc..

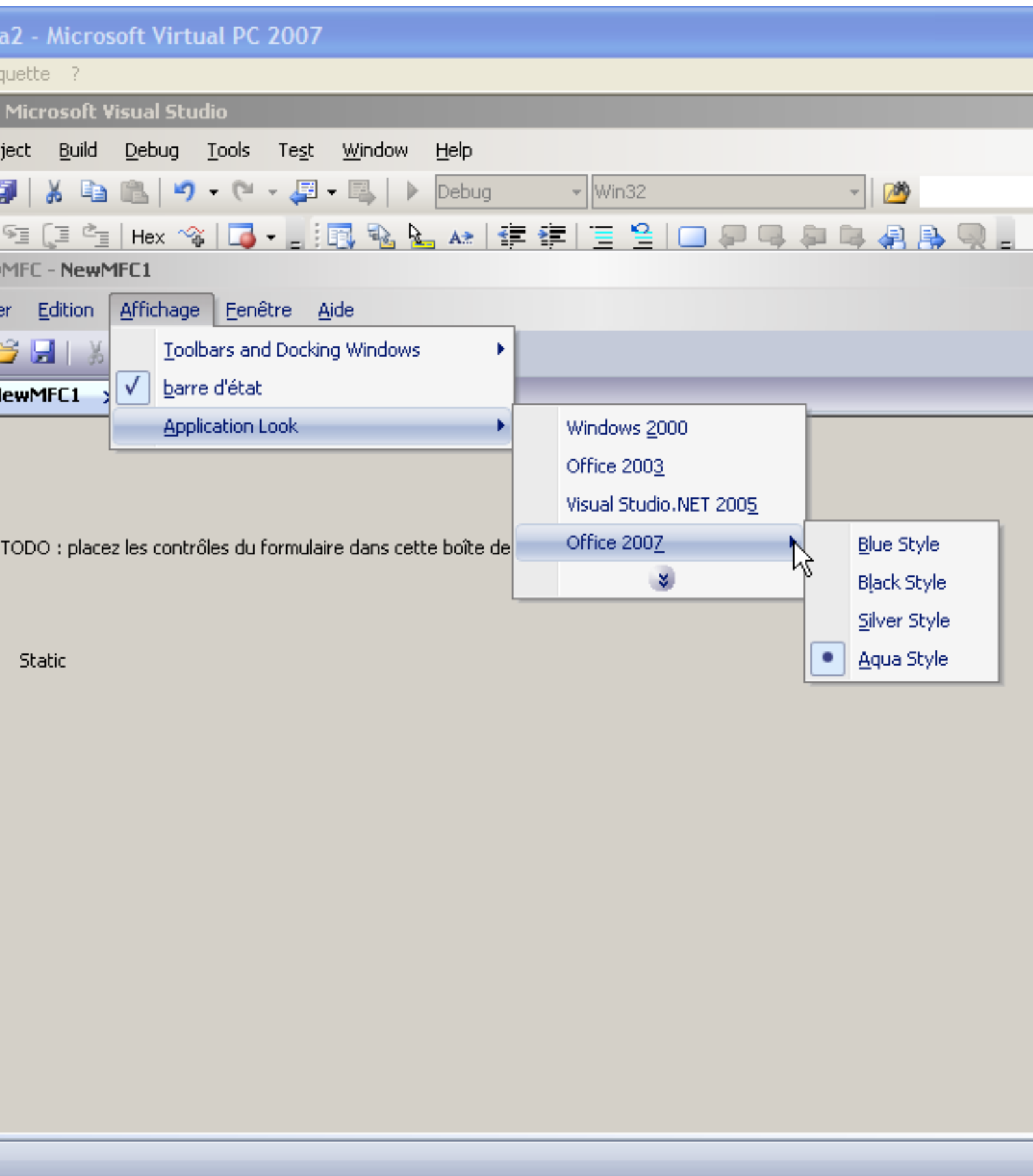
Support des thèmes vista : avec possibilité de passer d'un thème à l'autre


Personnalisation des menus et des barres d'outils par glisser déposer (drag and drop)...

Nombre de classes ont été rajoutées pour le Shell #

Et encore bien d'autres fonctionnalités.

Pour illustrer la chose j'ai généré sans écrire une ligne de code l'application suivante :

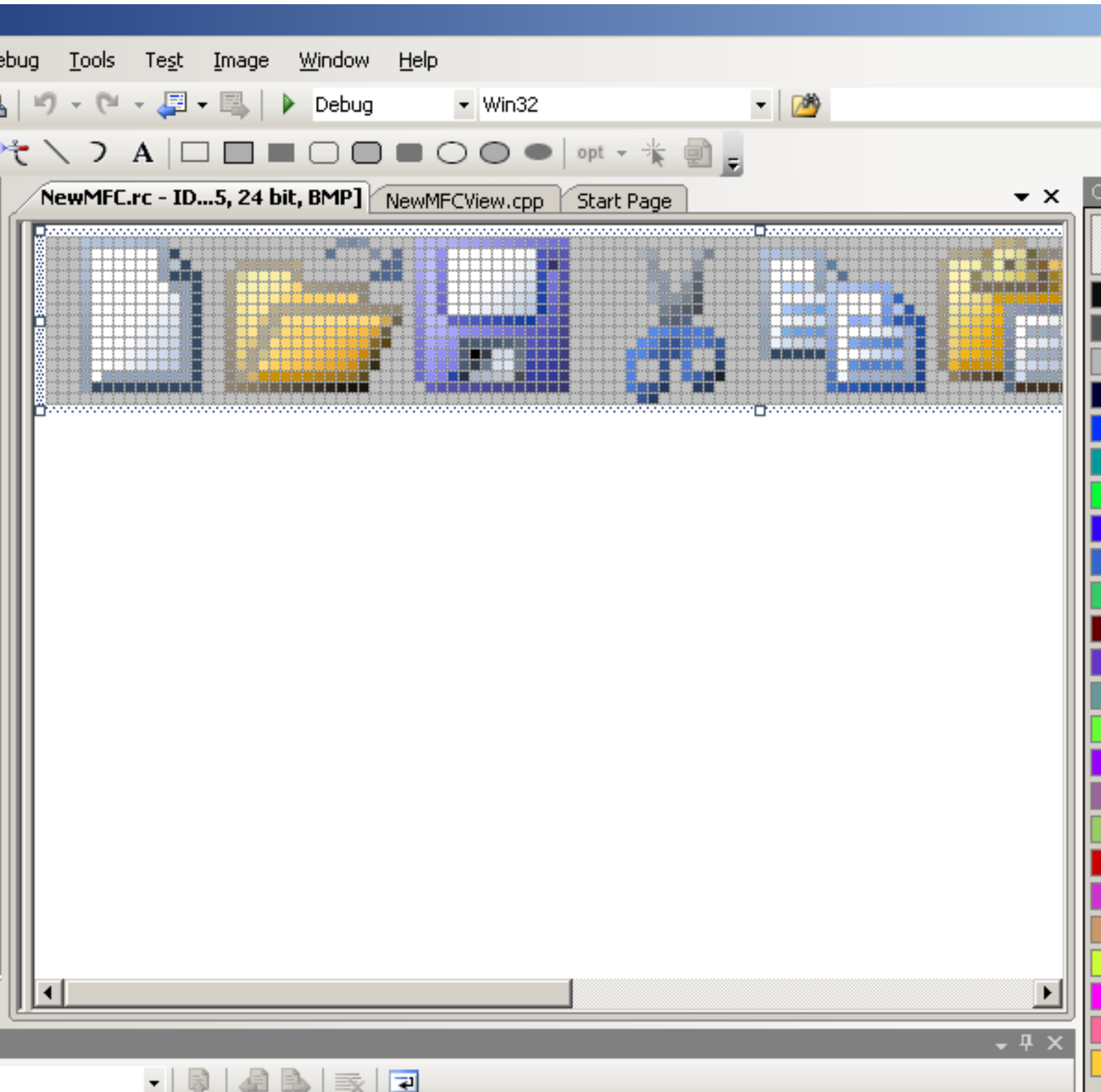


Concernant cette release un pack de  **documentation** à télécharger est disponible où l'on trouve un guide permettant d'adapter une ancienne application MFC vers les nouvelles classes pour modifier le look.

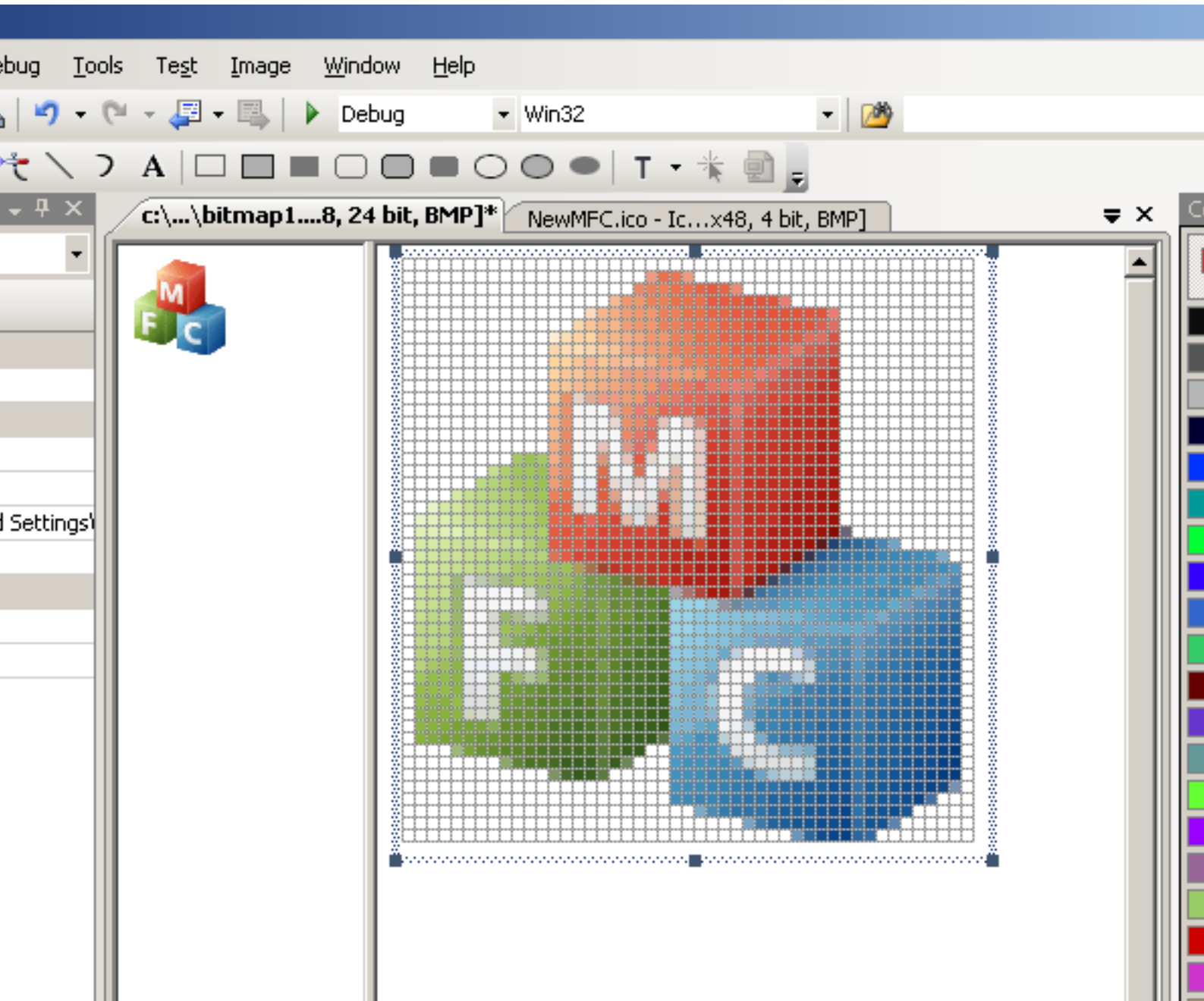
Je reviendrais sur ces différentes classes dans un autre article.

Cette mise à jour a eu aussi des effets sur l'environnement :

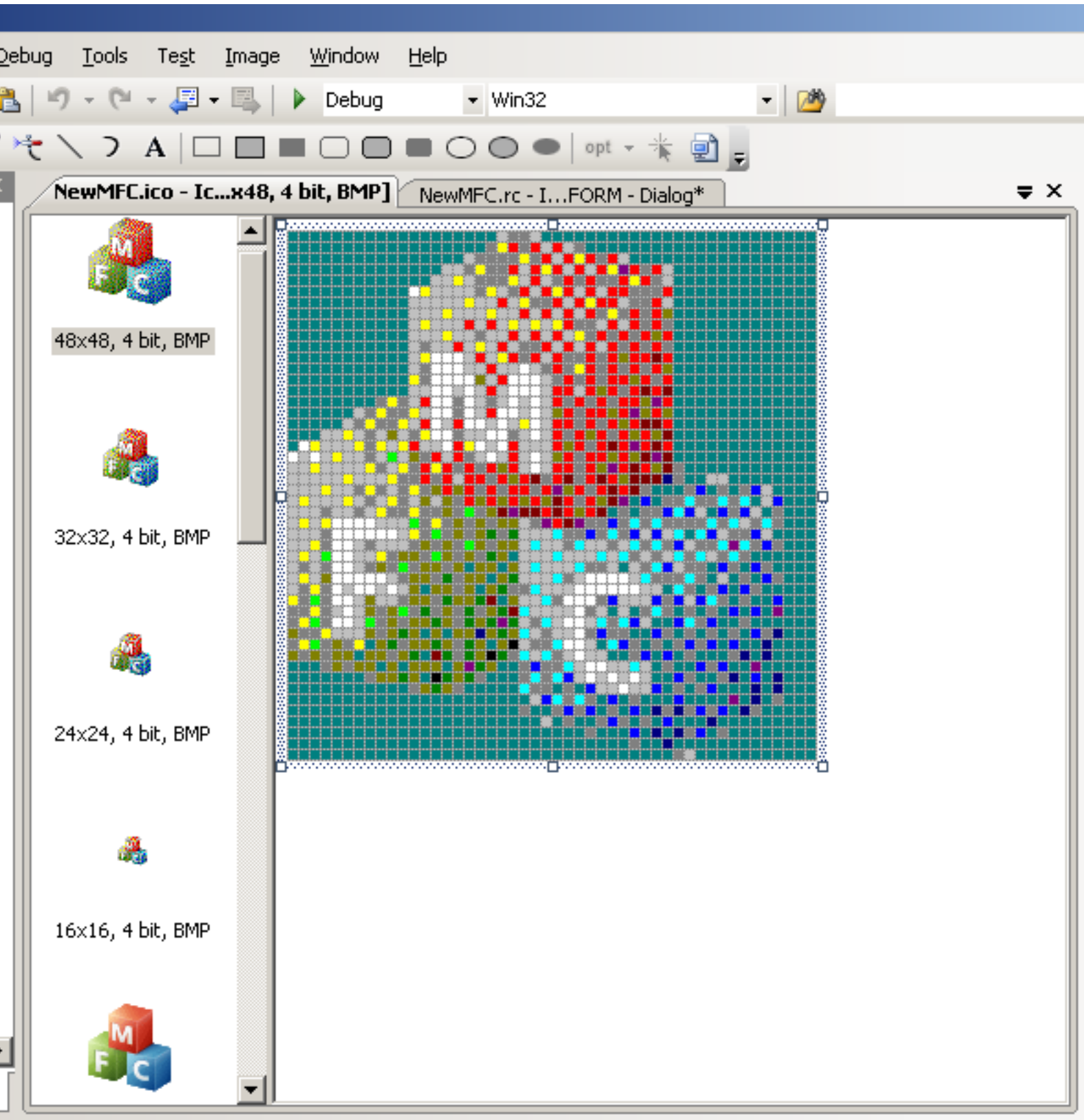
Les barres d'outils supportent maintenant les palettes de couleurs 24 bits (enfin !).



La même possibilité est disponible pour les bitmaps (fonctionne sans le pack MFC).



Enfin l'éditeur d'icônes permet de visualiser les différents formats avec des palettes plus fines.



Conclusions :

Cette version n'est pas une révolution en ce qui concerne le C++ natif mais une évolution des outils avec un éditeur qui se comporte beaucoup mieux que Visual 2005.

Pour les MFC, le pack de mise à jour donne un second souffle aux applications, et vu le niveau de possibilité donné à l'interface cela va peut être donner un regain d'intérêt pour cette plateforme de développement.

Néanmoins j'exprime quelques regrets :

Quand allons nous enfin pouvoir gérer la couleur et soyons fous un fond pour une forme/dialogue directement dans l'éditeur de ressource ?

Il me semble que vu le nombre de modifications apportées à la bibliothèque MFC cette modification n'aurait pas été de trop..

Une simple classe dérivée de **CFormView** ou de **CDialog** nous aurait suffit#

Je ne peux m'empêcher de faire la même remarque au niveau des contrôles.

Sera-t-il possible un jour de proposer des extensions de classe de base ?

Il n'est pas interdit de rêver non ?.

Ca serait vraiment génial, je pense par exemple à la classe **CTabControl** qui est particulièrement imbuvable.

Coté C++/CLI :

Pourquoi ne peut-on pas construire de projet pocket PC avec ce langage ?

C'est vraiment dommage, je ne comprends pas ce choix.

