La programmation Windows, le retour
Partie 1/5 : Le ribbon Windows

Nous vous proposons une série d'articles sur la pratique de la programmation Windows en C++. Cette nouvelle série se veut accessible à tous dans le but de faire des applications modernes. Ce premier article est consacrée au « Ribbon » alias le Ruban Windows que vous trouvez dans les applications comme MSPaint ou la suite Office.

Christophe PICHAUD | .NET Rangers by Sogeti
Consultant sur les technologies Microsoft
christophepichaud@hotmail.com | www.windowscpp.net

Le Ruban est apparu avec Microsoft Office 2007 et c'est une nouvelle expérience utilisateur qui permet de changer la présentation des anciens menus et des barres d'outils (toolbars). Avec le temps, Microsoft a constaté que les menus devenaient de plus en plus gros et que trouver une fonctionnalité devenait parfois difficile. Ils ont donc introduit ce nouveau paradigme sur l'interface UI. Et là, vous vous dites, est-ce que je peux mettre un ruban dans mon application ? Oui c'est possible et en natif. Il existe même deux solutions natives pour faire un ruban. Il y a la façon Win32 avec des API COM et il y a la version C/C++ disponible dans les MFC aka Microsoft Foundation Classes. Fig.1 et 2.

En effet depuis Visual Studio 2008 SP1, Microsoft a introduit le MFC Feature Pack qui contient de nombreuses classes C++ pour les MFC qui permettent de faire des applications exceptionnelles : support du Ruban, support des Docking Panes, Properties Windows, Options Windows. Bref, toutes ces fonctionnalités qui permettent de faire des applications modernes sont disponibles à portées de main avec leur code source car les MFC contiennent les headers .h et leur code source .cpp pratique pour le debugging avancé. On sait ce que fait le code. Pas de mystères. Pour montrer à quel point les API MFC sont poussées au maximum au niveau des détails, voici le ruban de la démo de l'application MSOffice2007Demo. Voici un extrait du ruban. On ne remarquera pas la différence avec le vrai. Fig.3.

Avant de voir le code, il faut parler de l'architecture du ruban. C'est plus qu'une grosse toolbar. C'est un élement contextuel qui permet de tester des opérations sans applications définitives. Je m'explique : vous passez la souris au dessus d'une couleur, et votre objet graphique sélectionné change automatiquement de couleur rien qu'avec le déplacement de la souris. Et si vous clickez, le changement s'opère. Fig.4 et 5.

Regardez comme ces panneaux graphiques apportent de multiples fonctionnalités. Il existe même une fonctionnalité « Plus de Couleurs... » pour afficher une boîte dédiée à la sélection de couleur. Fig.6.

Tout est concentré dans des éléments graphiques simples à visualiser mais complexes à implémenter sans un léger coup de main. Et c'est là que la magie du C++ entre en scène. Il existe une classe C++ par fonctionnalité de ruban. Eh oui, il est possible de coder tout cela en C++ sans avoir besoin de développer des montagnes de code. Comment ?

Commençons par lancer Visual Studio et demandons un nouveau projet MFC. Au niveau des options, il faut cocher un projet de type Office avec un Ruban. Fig.7. Une fois l'assistant fini, il suffit de compiler le code qui a été produit et voilà le résultat : Fig.8.

Ouvrons le fichier de ressources RC et admirons le ruban en mode édition : Fig.9. Et là vous me dites : « cool c'est simple y a un éditeur ! ». Je vous arrête de suite... Ce n'est pas comme cela que l'on fera le ruban de Word,

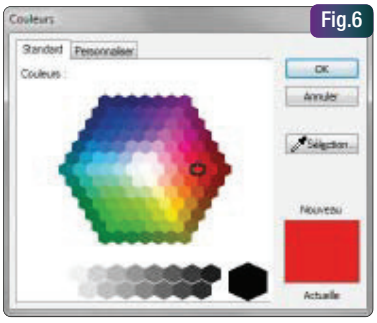

Fig.6

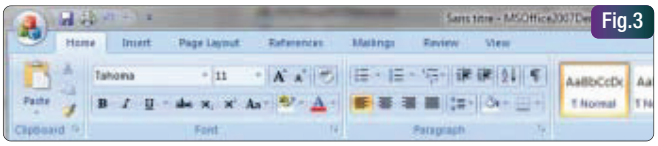

Fig.3

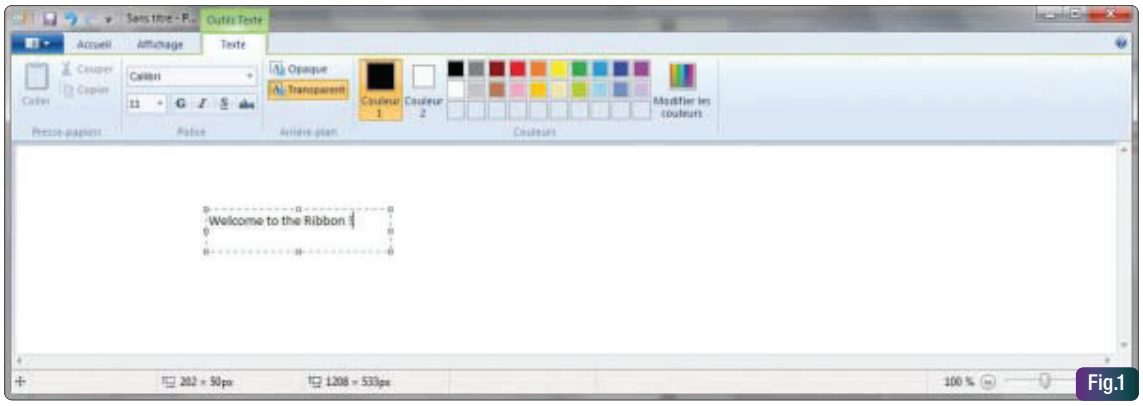

Fig.1

Fig.4

Fig.5


Fig.2

via l'éditeur. Nous allons construire le ruban tout en programmation pour supporter l'ensemble des gabarits disponibles. On va faire du code ! Je vous avais prévenu, le code Level 100 ce n'est pas aujourd'hui ! Et puis avouez que c'est plus marrant de faire du code non ?

Coder le ruban

Voici le ruban que nous allons créer : c'est une application existante que j'ai écrite pour que ma fille joue à la souris en faisant du dessin. Fig.10.
On commence par où ? Eh bien par le seul item qui ressemble de près ou de loin à un menu. Fig.11.
La première chose à faire c'est de créer l'objet ribbon : c'est le parent de tous les objets que nous allons créer.

```
CMFCRibbonBar    m_wndRibbonBar;
if (!m_wndRibbonBar.Create(this))
{
    return FALSE;
}
```

Ensuite, pour chaque élément CMFCRibbonButton, on fait un CMFCRibbonMainPanel::Add.

```
CMFCRibbonMainPanel* pMainPanel = m_wndRibbonBar.AddMainCategory(_T("File"), IDB_RIBBON_FILESMALL, IDB_RIBBON_FILELARGE);
  pMainPanel->Add(new CMFCRibbonButton(ID_FILE_NEW, _T("&New"), 0, 0));
  pMainPanel->Add(new CMFCRibbonButton(ID_FILE_OPEN, _T("&Open..."), 1, 1));
  pMainPanel->Add(new CMFCRibbonButton(ID_FILE_OPEN_AS_XML, _T("Open As XML..."), 1, 1));
  pMainPanel->Add(new CMFCRibbonButton(ID_FILE_SAVE, _T("&Save"), 2, 2));
  pMainPanel->Add(new CMFCRibbonButton(ID_FILE_SAVE_AS, _T("Save &As..."), 3, 3));
  pMainPanel->Add(new CMFCRibbonButton(ID_FILE_SAVE_AS_XML, _T("Save As & XML..."), 3, 3));
  pMainPanel->AddSeparator();
  pMainPanel->Add(new CMFCRibbonButton(ID_FILE_CLOSE, _T("&Close"), 8, 8));
  pMainPanel->AddRecentFilesList(_T("Recent Documents"));
  pMainPanel->AddToBottom(new CMFCRibbonMainPanelButton(ID_APP_EXIT, _T("E&xit"), 27));
```

Les numéros indiquent l'icône à utiliser par rapport à la ressource graphique IDB_RIBBON_FILESMALL et IDB_RIBBON_FILLARGE. Voilà pour le pave carré en haut à gauche du ruban.

Exemple du Ruban

Maintenant, il faut coder chaque bloc avec le détail nécessaire à l'architecture du ruban. Fig.12.
Dans cet exemple, « Home » est une catégorie de type CMFCRibbonCategory.
« Design » est un panel de type CMFCRibbonPanel.

Fig.12

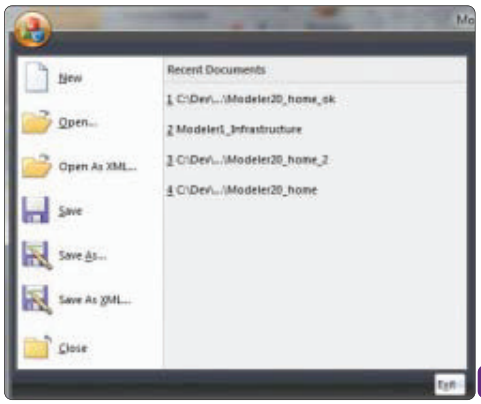

Fig.11

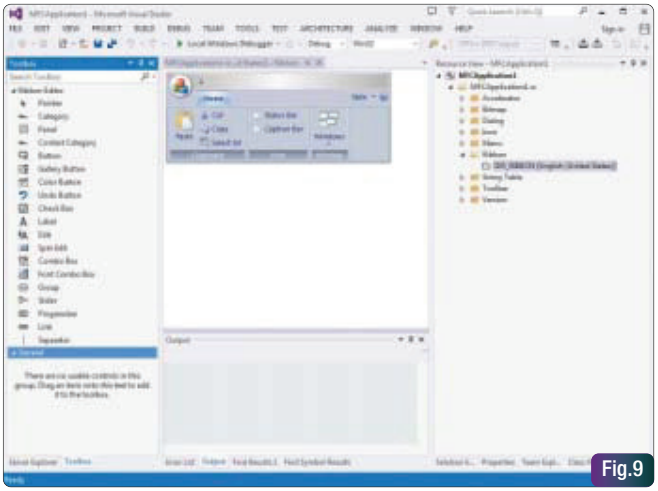
Fig.9

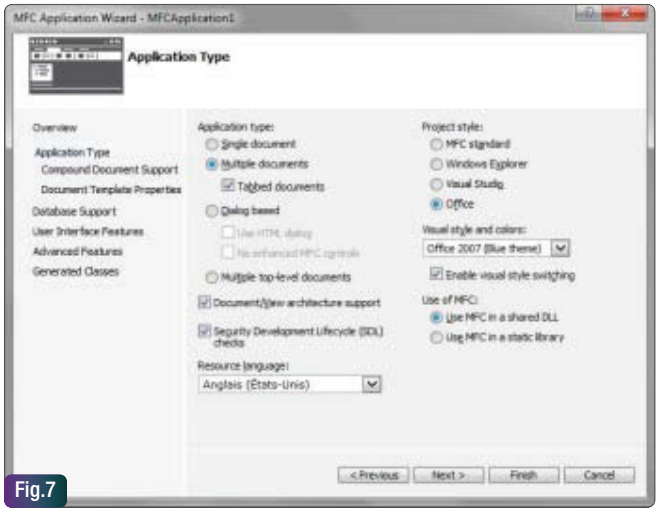

Fig.7

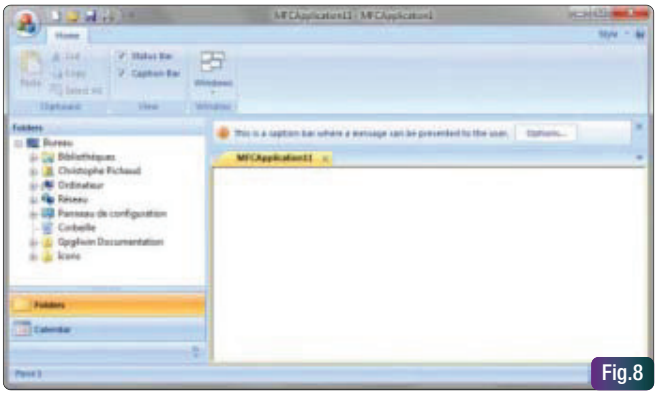
Fig.8

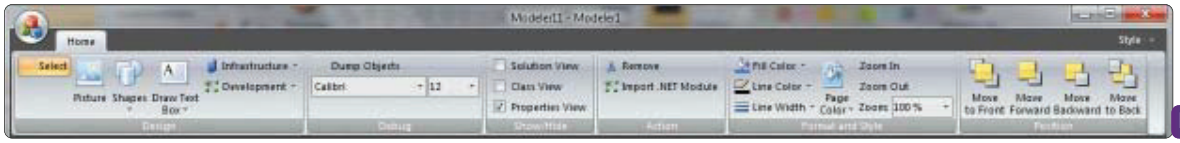

Fig.10

```
CMFCRibbonCategory* pCategory = m_wndRibbonBar.AddCategory(_T("&Home"), IDB
_RIBBON_WRITESMALL, IDB_RIBBON_WRITELARGE);
CMFCRibbonPanel* pPanelDesign = pCategory->AddPanel(_T("Design\nzd"), m_PanelI
mages.ExtractIcon(2));
```

Dans le panel “Design” on va ajouter des éléments. Par exemple, le bouton « Picture » :

```
CMFCRibbonButton* pInsertPictureBtn = new CMFCRibbonButton(ID_DESIGN_IMAGE, _T
("Picture"), 4, 4);
pPanelDesign->Add(pInsertPictureBtn);
```

L’icône associée au bouton provient de l’icône n°4 (index 0-based) de la ressource graphique IDB_RIBBON_WRITELARGE associée à writelarge.png dans le fichier .RC :

```
IDB_RIBBON_WRITELARGE  PNG              "res\\writelarge.png"
```

Fig.13.

Ce sont des images 32x32 pixels dans le même fichier. Un conseil : manipulez GIMP avec ce genre de chose car MSPaint devient très vite limité quand cela se joue au pixel prêt... Un conseil...

Ajoutons le bouton Shapes qui est une galerie. Fig.14.

C’est une galerie de type CMFCRibbonGallery.

```
CMFCRibbonGallery *pInsertShapesBtn = new CMFCRibbonGallery(ID_DESIGN_SHAPES,
_T("Shapes"), 6, 6);
pInsertShapesBtn->SetButtonMode();
pInsertShapesBtn->SetIconsInRow(12);
pInsertShapesBtn->AddGroup(_T("Recently Used Shapes"), IDB_SHAPE1, 20);
pInsertShapesBtn->AddGroup(_T("Lines"), IDB_SHAPE2, 20);
pInsertShapesBtn->AddGroup(_T("Basic Shapes"), IDB_SHAPE3, 20);
pInsertShapesBtn->AddGroup(_T("Block Arrows"), IDB_SHAPE4, 20);
pInsertShapesBtn->AddGroup(_T("Flowchart"), IDB_SHAPE5, 20);
pInsertShapesBtn->AddGroup(_T("Callouts"), IDB_SHAPE6, 20);
pInsertShapesBtn->AddGroup(_T("Stars and Banners"), IDB_SHAPE7, 20);
pInsertShapesBtn->AddSubItem(new CMFCRibbonButton(ID_DESIGN_SHAPES_NEW,
_T("&New Drawing Canvas"), 29, -1));
pPanelDesign->Add(pInsertShapesBtn);
```

Chaque ligne est une série de bitmaps enregistrés dans les ressources. Ex : IDB_SHAPE1 :

```
IDB_SHAPE1      BITMAP DISCARDABLE   "res\\shape1.bmp"
```

Ajoutons un menu sous forme de bouton : Fig.15.

Il suffit d’utiliser la classes CMFCRibbonButton et de faire appel à la méthode SetMenu().

```
CMFCRibbonButton* pBtnDrawText = new CMFCRibbonButton(ID_DESIGN_TEXTBOX,
_T("Draw Text Box\nx"), 15, 15);
pBtnDrawText->SetMenu(IDR_DRAW_TEXT_MENU);
pPanelDesign->Add(pBtnDrawText);
```

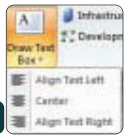

Fig.15

Fig.13

IDR_DRAW_TEXT_MENU est un menu défini dans le fichier de ressource RC. Ajoutons une liste de gabarits : Fig.16.

Il suffit d’utiliser la class CRibbonListButton et d’y ajouter une image et une liste des chînes de caractères. Fig.17.

```
CStringArray m_arInfraShapes;
m_arInfraShapes.Add(_T("AD Server"));
m_arInfraShapes.Add(_T("Server"));
m_arInfraShapes.Add(_T("Web Server "));
m_arInfraShapes.Add(_T("Database Server"));
m_arInfraShapes.Add(_T("Workstation"));
m_arInfraShapes.Add(_T("Laptop"));
m_arInfraShapes.Add(_T("Firewall"));
m_arInfraShapes.Add(_T("Network"));
m_arInfraShapes.Add(_T("Virtual Server"));
m_arInfraShapes.Add(_T("Virtual Web Server"));
m_arInfraShapes.Add(_T("Virtual Database Server"));
m_arInfraShapes.Add(_T("Virtualization Server"));
m_arInfraShapes.Add(_T("AD Server"));
m_arInfraShapes.Add(_T("Server"));
m_arInfraShapes.Add(_T("Database Server"));
m_arInfraShapes.Add(_T("Server Farm"));
m_arInfraShapes.Add(_T("Workstation"));
m_arInfraShapes.Add(_T("laptop"));

CRibbonListButton *pListBtnInfra = new CRibbonListButton(ID_DESIGN_SHAPESINFRA,
_T("Infrastructure\nti"), 20, -1, FALSE);
pListBtnInfra->AddGroup(_T("Built-In"), IDB_SHAPES_INFRA, 64, m_arInfraShapes);
pListBtnInfra->SetIconsInRow(4);
pListBtnInfra->EnableMenuResize();
pPanelDesign->Add(pListBtnInfra);
```

L’élément suivant du ruban est aussi un ensemble de gabarits. Il suffit de garder le même code mais de changer la liste de chaînes de caractères et l’image. Fig.18 et 19. Ajoutons la combobox des polices et la combobox de taille : Fig.20. Il suffit d’utiliser la classe CMFCRibbonFontComboBox pour afficher les polices. La combobox pour les tailles est une simple utilisation de la classe CMFCRibbonComboBox.

Fig.16

Fig.17

Fig.14

```
// Create "Debug" panel
CMFCRibbonPanel* pPanelDebug = pCategory->AddPanel(_T("Debug\nzd"), m_PanelImages.
ExtractIcon(2));

CMFCRibbonButtonsGroup * apFontGroup = new CMFCRibbonButtonsGroup();

CMFCRibbonFontComboBox::m_bDrawUsingFont = TRUE;

m_pFontCombo = new CMFCRibbonFontComboBox(ID_FONT_FONT,TRUETYPE_FONTTYPE );
m_pFontCombo->SetWidth(55, TRUE); // Width in "floaty" mode
m_pFontCombo->SelectItem(_T("Arial"));
apFontGroup->AddButton(m_pFontCombo);

m_pFontSizeCombo = new CMFCRibbonComboBox(ID_FONT_FONTSIZE, FALSE, 39);
m_pFontSizeCombo->AddItem(_T("8"));
m_pFontSizeCombo->AddItem(_T("9"));
m_pFontSizeCombo->AddItem(_T("10"));
m_pFontSizeCombo->AddItem(_T("11"));
m_pFontSizeCombo->AddItem(_T("12"));
m_pFontSizeCombo->AddItem(_T("14"));
m_pFontSizeCombo->AddItem(_T("16"));
m_pFontSizeCombo->AddItem(_T("18"));
m_pFontSizeCombo->AddItem(_T("20"));
m_pFontSizeCombo->SetWidth(20, TRUE); // Width in "floaty" mode
m_pFontSizeCombo->SelectItem(3);
apFontGroup->AddButton(m_pFontSizeCombo);

pPanelDebug->Add(apFontGroup);
```

Ajoutons une combobox : Fig.21.
Il suffit d'utiliser la classe CMFCRibbonCheckBox.

```
// Create "Show/Hide" panel:
CMFCRibbonPanel* pPanelShow = pCategory->AddPanel(_T("Show/Hide\nzs"), m_PanelImages.
ExtractIcon(4));
pPanelShow->Add(new CMFCRibbonCheckBox(ID_VIEW_FILE_VIEW, _T("Solution View\nc")));
pPanelShow->Add(new CMFCRibbonCheckBox(ID_VIEW_CLASS_VIEW, _T("Class View\nc")));
pPanelShow->Add(new CMFCRibbonCheckBox(ID_VIEW_PROPERTIES, _T("Properties View\np")));
```

Ajoutons un bouton pour choisir la couleur de fond de l'écran : Fig.22.
Il suffit d'utiliser la classe CMFCRibbonColorButton et d'y passer en paramètres des couleurs de palette. Il y a 3 palettes mais le code ici n'en présente qu'une. Le code est identique sauf qu'il y a 50 couleurs en plus...

```
CList<COLORREF,COLORREF> m_lstMainColors;

static ColorTableEntry colorsMain [] =
{
    { RGB(255, 255, 255), _T("White, Background 1") },
    { RGB(0, 0, 0), _T("Black, Text 1") },
    { RGB(238, 236, 225), _T("Tan, Background 2") },
    { RGB(31, 73, 125), _T("Dark Blue, Text 2") },
    { RGB(79, 129, 189), _T("Blue, Accent 1") },
    { RGB(192, 80, 77), _T("Red, Accent 2") },
    { RGB(155, 187, 89), _T("Olive Green, Accent 3") },
    { RGB(128, 100, 162), _T("Purple, Accent 4") },
    { RGB(75, 172, 198), _T("Aqua, Accent 5") },
    { RGB(245, 150, 70), _T("Orange, Accent 6") }
};

nNumColours = sizeof(colorsMain) / sizeof(ColorTableEntry);

for (i = 0; i < nNumColours; i++)
{
    m_lstMainColors.AddTail(colorsMain [i].color);
    CMFCRibbonColorButton::SetColorName(colorsMain [i].color, colorsMain [i].szName);
};

CMFCRibbonColorButton * pBtnPageColor = new CMFCRibbonColorButton(ID_FORMAT_PAGECOLOR,
_T("Page Color\npo"), TRUE, -1, 26);
pBtnPageColor->SetDefaultCommand(FALSE);
pBtnPageColor->EnableAutomaticButton(_T("&Automatic"), RGB(0, 0, 0));
pBtnPageColor->EnableOtherButton(_T("&More Line Colors..."), _T("More Line Colors"));
pBtnPageColor->SetColumns(10);
pBtnPageColor->SetColorBoxSize(CSize(17, 17));
pBtnPageColor->AddColorsGroup(_T("Theme Colors"), m_lstMainColors);
pBtnPageColor->AddColorsGroup(_T(""), m_lstAdditionalColors, TRUE);
pBtnPageColor->AddColorsGroup(_T("Standard Colors"), m_lstStandardColors);
pPanelFormat->Add(pBtnPageColor);
```

Ajoutons les deux boutons « ZoomIn » et « ZoomOut » :
Il suffit d'utiliser la class CMFCRibbonButton.

```
pPanelFormat->Add(new CMFCRibbonButton(ID_FORMAT_ZOOM_IN, _T("Zoom In\ni"), -1));
pPanelFormat->Add(new CMFCRibbonButton(ID_FORMAT_ZOOM_OUT, _T("Zoom Out\no"), -1));
```

Vous remarquerez que le -1 indique ne pas prendre d'icônes.

Conclusion

Construire un Ruban par programmation n'est pas si compliqué que cela car nous disposons de classes MFC qui font le job. La mécanique est toujours la même : on passe un identifiant d'image ou un numéro d'index sur un bitmap multiple et le tour est joué. Le codage d'un Ribbon comme celui de Office Word 2007 par exemple est disponible dans « Microsoft Visual C++ 2008 SP1 Sample Library »:
http://www.microsoft.com/fr-fr/download/details.aspx?id=5718
Vous télécharger vc_samples qui fait 8 MB et une fois installé, vous allez dans le répertoire C++\MFC\Visual C++ 2008 Feature Pack et vous tomberez sur les exemples suivants : MSOffice2007Demo, VisualStudioDemo, OutlookDemo, etc. Le code complet du Ribbon de cet article est accessible sur codeplex à l'URL suivante : http://ultrafluid.codeplex.com/