Implémentation d'une interface dotée de cadres positionnables avec la bibliothèque wxAUI, pour les wxWidgets

Aaron Williams et Ben Williams, Kirix Corporation

Grâce à la bibliothèque wxAUI, les développeurs peuvent créer des applications dotées de cadres natifs, flottants, positionnables, ou déposables ; de barres d'outils à ressort, de sauvegardes et de chargements en perspective et d'effets graphiques spéciaux, tels qu'une convivialité personnalisable ou de fenêtre transparente notamment lors d'un glisser-déposer d'éléments.

Sur l'auteur :

Aaron Williams est scientifique en chef de la Kirix Corporation, et Ben Williams travaille comme architecte en chef des logiciels. Il est également l'auteur de la bibliothèque wxAUI. La corporation Kirix est à l'origine du développement d'un navigateur de données agile et multi-plateformes appelé Kirix Strata, permettant à quiconque d'accéder rapidement, d'analyser et de manipuler une quantité quasi illimitée de données. En 2005, Strata a remporté le prix d'excellence LinuxWorld Product dans la catégorie meilleure application bureau / entreprise / productivité. Vous pouvez contacter les auteurs aux adresses suivantes : *awilliams@kirix.com* ou *bwilliams@kirix.com*

Afin de rester en phase avec l'innovation dans les logiciels, les développeurs d'applications sont soumis à des pressions de plus en plus fortes, et doivent proposer des interfaces d'applications de grande qualité, capables de répondre aux besoins des utilisateurs. Ceux-ci affichent un intérêt croissant dans la réalisation de tâches spécifiques au moyen d'un ensemble de commandes simple et convivial, sans avoir à se soucier des mécanismes sous-jacents de l'application en question.

Alors que les applications elles-mêmes peuvent varier, les interfaces bien conçues ont toujours recours à un ensemble de fonctionnalités qui leur sont communes, telles que les cadres positionnables et la gestion de la barre d'outils. Ces fonctionnalités permettent, en effet, d'exposer les fonctions clés de l'application de manière discrète et facilement accessible. Grâce à ce cadre d'application, les développeurs peuvent proposer aux utilisateurs les outils dont ils ont exactement besoin au moment et à l'emplacement souhaité.

Nous nous efforçons, dans nos applications, de garantir à nos utilisateurs un accès pratique aux outils primordiaux. Tout au long de notre expérience en développement, nous avons compris que de nombreuses fonctionnalités clés des interfaces graphiques utilisateur pouvaient être regroupées dans un ensemble de comportements généralisés. Si nous pouvions créer une bibliothèque particulière capable de résoudre ces problèmes, il serait alors possible de créer plus rapidement des interfaces *sophistiquées* sans avoir à implémenter des éléments un à un de manière improvisée.

C'est ainsi qu'est née la bibliothèque wxAUI (Advanced User Interface), proposant une interface utilisateur sophistiquée pour la boîte à outils wxWidgets. L'objectif de cette bibliothèque consiste à implémenter une interface conviviale do-

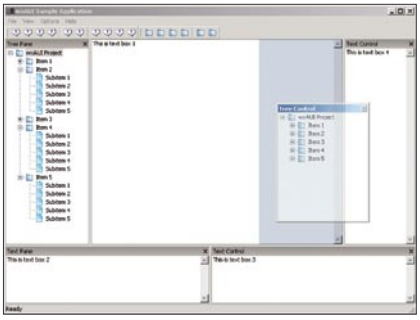

Figure 1. *Exemple d'application avec la bibliothèque wxAUI*

tée de fonctionnalités *à la pointe du progrès* afin de permettre aux développeurs de concevoir rapidement et facilement des interfaces d'applications élégantes et pratiques pouvant fonctionner sur plusieurs plate-formes. Il s'agit d'une bibliothèque libre distribuée sous la licence wxWidgets et téléchargeable à partir du site suivant : *http://www.kirix.com.*

Grâce à la bibliothèque wxAUI, les développeurs peuvent désormais créer des applications dotées de cadres natifs, flottants, positionnables, ou déposables ; de barres d'outils *à ressort*, de sauvegardes et de chargements en perspective et d'effets graphiques spéciaux, tels qu'une convivialité personnalisable ou de fenêtre transparente notamment lors d'un glisser-déposer d'éléments.

Démarrage

Afin de mieux comprendre l'implémentation de ces fonctionnalités, nous vous proposons d'élaborer un exemple d'application chargée d'installer des cadres positionnables, des barres d'outils déplaçables, ainsi que plusieurs options différentes autour de la convivialité.

Avant d'implémenter une de ces fonctionnalités d'interfaces, il faut tout d'abord développer une application classique au moyen de la bibliothèque wxWidgets, boîte à outils graphique puissante permettant aux développeurs d'élaborer facilement des applications multi plateformes dotées d'une convivialité native.

Il faut tout d'abord créer une classe intitulée `MyApp dérivant` de la classe wxApp, qui représente l'application. Cette classe stocke les réglages de l'application, crée le système de messages par fenêtres ou les boucles d'évènements, et permet de passer les arguments en ligne de commande dans l'application.

Il suffit, ensuite, d'implémenter, dans notre classe dérivée `MyApp`, le comportement de l'application en modifiant la fonction `OnInit()`, chargée de définir le point d'entrée dans l'application. Il faut, en effet, créer, dans cette fonction, un cadre, puis ordonner à l'application d'utiliser ce cadre en tant que fenêtre d'application de niveau supérieur, de l'afficher de manière à être visible pour l'utilisateur, et de poursuivre le traitement des évènements jusqu'à la fermeture du cadre.

Afin de créer un cadre, il suffit de dériver d'une autre classe intitulée `MyFrame` issue elle-même de la classe wxFrame, dans laquelle nous allons implémenter notre interface sous forme de fenêtre, comme les menus, les barres d'outils, les fenêtres filles, le comportement positionnable, et d'autres fonctionnalités spécifiques de l'interface. Une fois ce cadre créé, il faut ordonner à l'application de

Listing 1. *Application classique*

```
#include <wx/wx.h>

// -- application --
class MyApp : public wxApp
{
public:
   bool OnInit();
};
DECLARE_APP(MyApp);
IMPLEMENT_APP(MyApp);
// -- cadre --
class MyFrame : public wxFrame
{
public:
   MyFrame(wxWindow* parent,
      wxWindowID id,
      const wxString& title,
      const wxPoint& pos = wxDefaultPosition,
      const wxSize& size = wxDefaultSize,
      long style = wxDEFAULT_FRAME_STYLE | wxSUNKEN_BORDER);
   ~MyFrame();
private:
   DECLARE_EVENT_TABLE();
};
bool MyApp::OnInit()
{
   wxFrame* frame = new MyFrame(NULL,
      wxID_ANY,
      wxT("wxAUI Sample Application"),
      wxDefaultPosition,
      wxSize(800, 600));
   SetTopWindow(frame);
   frame->Show();
   return true;
}
BEGIN_EVENT_TABLE(MyFrame, wxFrame)
END_EVENT_TABLE()
MyFrame::MyFrame(wxWindow* parent,
   wxWindowID id,
   const wxString& title,
   const wxPoint& pos,
   const wxSize& size,
   long style)
   : wxFrame(parent, id, title, pos, size, style)
{
}
MyFrame::~MyFrame()
{
}
```

Listing 2. *Application classique dotée de cadres positionnables*

```
#include <wx/wx.h>
#include <wx/treectrl.h>
#include <wx/artprov.h>
#include "manager.h"
// -- application --
class MyApp : public wxApp
{
public:
   bool OnInit();
};
DECLARE_APP(MyApp);
IMPLEMENT_APP(MyApp);
// -- cadre --
class MyFrame : public wxFrame
{
   enum
   {
      ID_CreateTree = wxID_HIGHEST+1,
      ID_CreateText,
      ID_About,
   };
public:
   MyFrame(wxWindow* parent,
      wxWindowID id,
      const wxString& title,
      const wxPoint& pos = wxDefaultPosition,
      const wxSize& size = wxDefaultSize,
      long style = wxDEFAULT_FRAME_STYLE | wxSUNKEN_BORDER);
   ~MyFrame();
private:
   wxTextCtrl* CreateTextCtrl();
   wxTreeCtrl* CreateTreeCtrl();
private:
   void OnEraseBackground(wxEraseEvent& event);
   void OnSize(wxSizeEvent& event);
private:
   wxFrameManager m_mgr;
   DECLARE_EVENT_TABLE();
};
bool MyApp::OnInit()
{
   wxFrame* frame = new MyFrame(NULL,
      wxID_ANY,
      wxT("wxAUI Sample Application"),
      wxDefaultPosition,
      wxSize(800, 600));
   SetTopWindow(frame);
   frame->Show();
   return true;
}
BEGIN_EVENT_TABLE(MyFrame, wxFrame)
   EVT_ERASE_BACKGROUND(MyFrame::OnEraseBackground)
   EVT_SIZE(MyFrame::OnSize)
END_EVENT_TABLE()
MyFrame::MyFrame(wxWindow* parent,
   wxWindowID id,
   const wxString& title,
   const wxPoint& pos,
   const wxSize& size,
   long style)
   : wxFrame(parent, id, title, pos, size, style)
{
   // indiquer à wxFrameManager de gérer ce cadre
   m_mgr.SetFrame(this);

   // création de l'icône du cadre
```

l'utiliser en tant que fenêtre d'application de niveau supérieur en la passant dans la fonction `SetTopWindow()` . SetTopWindow() est quand à elle appelé lors de l'initialisation qui se fait avec la fonction `MyApp::OnInit()`.

Enfin, il suffit d'indiquer à l'application d'afficher le cadre au moyen de la fonction `Show()`, puis de poursuivre le traitement des évènements de la fenêtre en retournant la valeur *true* à l'issue de la fonction `OnInit()`.

À ce stade, nous avons défini le cadre d'une application classique ainsi qu'une fenêtre de niveau supérieur, comme l'illustre le Listing 1. Nous sommes donc prêts à personnaliser l'interface de l'application en développant des éléments dans la classe `MyFrame`.

Cadres positionnables

C'est ici que le développement devient vraiment intéressant. Afin d'implémenter des barres d'outils positionnables dans des cadres et déplaçables, il faut créer un gestionnaire de cadres wxAUI dans la classe `MyFrame`, chargée de gérer les cadres et les barres d'outils de notre application. En réalité, comme wxAUI implémente cette fonctionnalité dans une classe à part intitulée wxFrameManager, il est possible de l'utiliser avec n'importe quelle classe dérivée de wxFrame, y compris la classe wxMDIParentFrame, afin de disposer d'une meilleure flexibilité dans l'utilisation des fonctionnalités du gestionnaire.

Une fois le gestionnaire de cadres wxAUI créé dans `MyFrame`, il suffit d'ajouter les cadres et les barres de menus que nous souhaitons gérer avec le gestionnaire, puis de préciser les informations dont a besoin le gestionnaire afin de déterminer la gestion des fenêtres et des barres de menus.

Tout d'abord, afin de créer un gestionnaire de cadres wxAUI, il faut ajouter une variable membre de type wxFrameManager à `MyFrame`, on crée ainsi l'objet m_mgr. . Puis dans le constructeur de MyFrame on initialise l'objet m_mgr grâce à la méthode SetFrame de la classe wxFrameManager, MyFrame est alors sous le contrôle du gestionnaire.

Il faut, ensuite, créer les cadres et les barres d'outils que nous souhaitons pouvoir positionner, puis les ajouter au gestionnaire au moyen de la fonction

`Add()` avec certaines informations relatives à la gestion des cadres, mentionnées au moyen de la classe wxPaneInfo de wxAUI.

Enfin, il suffit d'indiquer au gestionnaire de lancer la gestion des toutes nouvelles sous-fenêtres ajoutées en appelant sa fonction `Update()`. Bien évidemment, il faut également vider de son contenu le gestionnaire une fois le cadre détruit, en appelant la fonction `UnInit()` du gestionnaire dans le destructeur `MyFrame`.

Lorsque l'application est lancée, wxFrameManager se charge de gérer les sous-fenêtres qui lui sont associées pour un wxFrame particulier, en ayant recours à une classe remplaçable dock art chargée de réaliser l'ensemble des traits susceptibles d'être personnalisés en fonction des besoins spécifique de l'application.

Le gestionnaire détermine la couche positionnable selon quatre paramètres : la direction, la position, la rangée, et la couche. Le paramètre direction permet de déterminer si un cadre sera positionné en haut, en bas, à gauche, à droite ou au centre de la fenêtre de niveau supérieur. Les paramètres position et rangée permettent de positionner plus d'un cadre dans chacun de ces emplacements. Ainsi, par exemple, avec deux cadres positionnés à gauche, le cadre supérieur situé dans la même fenêtre aura une position de zéro, et le second une position de un. Enfin, le paramètre couche permet de positionner les cadres à de plusieurs niveaux.

Nous avons exposé dans le Listing 2 notre application dotée du nouveau gestionnaire de cadres, ainsi que de quelques barres d'outils et de cadres qu'il doit gérer.

Perspectives et apparences

Une fois les cadres positionnables implémentés, il est possible de sauvegarder les positions des cadres, puis de les recharger ultérieurement afin de créer différentes perspectives de l'application.

Grâce à la bibliothèque wxAUI, la manipulation est très simple : il suffit de configurer les panneaux à votre guise, puis de sauvegarder la perspective grâce à la fonction `SavePerspective()` de wxFrameManager, chargée de créer une chaîne pour stocker les informa-

```
#ifdef __WXMSW__
SetIcon(wxIcon(wxT("mondrian"), wxBITMAP_TYPE_ICO_RESOURCE, 16, 16));
#endif
// création du menu
wxMenuBar* mb = new wxMenuBar;
wxMenu* file_menu = new wxMenu;
file_menu->Append(wxID_EXIT, _("Exit"));
wxMenu* help_menu = new wxMenu;
help_menu->Append(ID_About, _("About..."));
mb->Append(file_menu, _("File"));
mb->Append(help_menu, _("Help"));
SetMenuBar(mb);
CreateStatusBar();
GetStatusBar()->SetStatusText(_("Ready"));
// paramétrer la taille minimum du cadre
SetMinSize(wxSize(400,300));
// création de quelques barres d'outils
wxToolBar* tb1 = new wxToolBar(this, -1, wxDefaultPosition, wxDefaultSize,
   wxTB_FLAT | wxTB_NODIVIDER);
tb1->SetToolBitmapSize(wxSize(16,16));
wxBitmap tb1_bmp1 = wxArtProvider::GetBitmap(wxART_QUESTION, wxART_OTHER,
                    wxSize(16,16));
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddSeparator();
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddSeparator();
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->Realize();
wxToolBar* tb2 = new wxToolBar(this, -1, wxDefaultPosition, wxDefaultSize,
   wxTB_FLAT | wxTB_NODIVIDER);
tb2->SetToolBitmapSize(wxSize(16,16));
wxBitmap tb2_bmp1 = wxArtProvider::GetBitmap(wxART_FOLDER, wxART_OTHER,
                    wxSize(16,16));
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->AddSeparator();
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->Realize();
// ajout d'un panneau central
m_mgr.AddPane(CreateTextCtrl(), wxPaneInfo().Name(wxT("text_content")).
   CenterPane());
// ajout de quelques panneaux positionnés
m_mgr.AddPane(CreateTreeCtrl(), wxPaneInfo().
   Name(wxT("test1")).Caption(wxT("Tree Pane")).
   Left().Layer(1).Position(1));
m_mgr.AddPane(CreateTextCtrl(), wxPaneInfo().
   Name(wxT("test2")).Caption(wxT("Text Pane")).
   Bottom().Layer(1).Position(1));
// ajout de barres d'outils dans le gestionnaire
m_mgr.AddPane(tb1, wxPaneInfo().
   Name(wxT("tb1")).Caption(wxT("Toolbar 1")).
   ToolbarPane().Top().Row(1).
   LeftDockable(false).RightDockable(false));
m_mgr.AddPane(tb2, wxPaneInfo().
   Name(wxT("tb2")).Caption(wxT("Toolbar 1")).
   ToolbarPane().Top().Row(1).Position(1).
   LeftDockable(false).RightDockable(false));
// "exécuter" toutes les modifications réalisées dans wxFrameManager
```

```
    m_mgr.Update();
}
MyFrame::~MyFrame()
{
    m_mgr.UnInit();
}
void MyFrame::OnEraseBackground(wxEraseEvent& event)
{
    event.Skip();
}
void MyFrame::OnSize(wxSizeEvent& event)
{
    event.Skip();
}
wxTextCtrl* MyFrame::CreateTextCtrl()
{
    wxString text;
    static int n = 0;
    text.Printf(wxT("This is text box %d"), ++n);
    return new wxTextCtrl(this,-1, text,
       wxPoint(0,0), wxSize(150,90),
       wxNO_BORDER | wxTE_MULTILINE);
}
wxTreeCtrl* MyFrame::CreateTreeCtrl()
{
    wxTreeCtrl* tree = new wxTreeCtrl(this, -1,
       wxPoint(0,0), wxSize(160,250),
       wxTR_DEFAULT_STYLE | wxNO_BORDER);
    wxTreeItemId root = tree->AddRoot(wxT("wxAUI Project"));
    wxArrayTreeItemIds items;
    wxImageList* imglist = new wxImageList(16, 16, true, 2);
    imglist->Add(wxArtProvider::GetBitmap(wxART_FOLDER, wxART_OTHER,
                      wxSize(16,16)));
    imglist->Add(wxArtProvider::GetBitmap(wxART_NORMAL_FILE, wxART_OTHER,
                      wxSize(16,16)));
    tree->AssignImageList(imglist);
    items.Add(tree->AppendItem(root, wxT("Item 1"), 0));
    items.Add(tree->AppendItem(root, wxT("Item 2"), 0));
    items.Add(tree->AppendItem(root, wxT("Item 3"), 0));
    items.Add(tree->AppendItem(root, wxT("Item 4"), 0));
    items.Add(tree->AppendItem(root, wxT("Item 5"), 0));
    int i, count;
    for (i = 0, count = items.Count(); i < count; ++i)
    {
       wxTreeItemId id = items.Item(i);
       tree->AppendItem(id, wxT("Subitem 1"), 1);
       tree->AppendItem(id, wxT("Subitem 2"), 1);
       tree->AppendItem(id, wxT("Subitem 3"), 1);
       tree->AppendItem(id, wxT("Subitem 4"), 1);
       tree->AppendItem(id, wxT("Subitem 5"), 1);
    }
    tree->Expand(root);
    return tree;
}
```

tions requises à la restauration de votre configuration des positions. Au moment de restituer la perspective de votre choix, il faut passer cette chaîne dans la fonction `LoadPerspective()` de wxFrameManager, afin que le gestionnaire puisse reconfigurer les panneaux dans l'état de la perspective sauvegardée.

Afin d'ajouter un support de perspective à notre application, il faut ajouter un objet menu *View* qui nous permettra de créer de nouveaux panneaux positionnables, ainsi que des perspectives chargeables par la suite. Une fois le menu et les fonctions de création de panneaux ajoutées, il faut créer deux gestionnaires pour les évènements du menu : un chargé de sauvegarder la perspective appelée `OnCreatePerspective()`, et l'autre responsable du chargement de la perspective appelée `OnRestorePerspective()`.

La perspective `OnCreatePerspective()` ouvre une boîte de dialogue dans laquelle nous pouvons taper le nom d'une perspective, puis ajoute cette perspective au menu *View* de sorte à pouvoir être chargée au moment de cliquer sur l'objet menu. La fonction `OnRestorePerspective()` se charge de restituer la perspective associée au nom spécifique classé dans le menu *View*.

Enfin, pour parfaire notre application, nous souhaitons modifier la convivialité de la fonctionnalité de positionnement de notre application. Il suffit, pour ce faire, de paramétrer les drapeaux de wxFrameManager grâce à la fonction `SetFlags()`. En suivant les mêmes étapes mentionnées plus haut pour les perspectives sauvegardées, il faut créer un objet menu *Options* qui nous permettra de régler différentes combinaisons de drapeaux, puis d'utiliser ces objets menu afin d'appeler la fonction `OnManagerFlag()`, chargée de mettre à jour les drapeaux et de modifier la convivialité du positionnement.

Nous avons exposé dans le Listing 3 le programme achevé.

Résumé

Grâce à la bibliothèque wxAUI, les développeurs peuvent désormais créer des applications dotées de cadres natifs, flottants ou positionnables, de barres d'outils déplaçables et *à ressort*, de perspectives pouvant être

sauvegardées et chargées, et d'effets graphiques particuliers, tels qu'une convivialité personnalisables ou de fenêtre transparente lors d'un glisser-déposer.

Grâce à ces fonctionnalités, les développeurs peuvent mieux se concentrer sur la conception d'interfaces élégantes, capables non seulement d'intégrer des fonctionnalités sophistiquées, mais également d'assister l'utilisateur final dans son travail de manière plus efficace. ■

Listing 3. *Application finale*

```
#include <wx/wx.h>
#include <wx/treectrl.h>
#include <wx/artprov.h>
#include "manager.h"
// -- application --
class MyApp : public wxApp
{
public:
   bool OnInit();
};
DECLARE_APP(MyApp);
IMPLEMENT_APP(MyApp);
// -- cadre --
class MyFrame : public wxFrame
{
   enum
   {
      ID_CreateTree = wxID_HIGHEST+1,
      ID_CreateText,
      ID_CreatePerspective,
      ID_AllowFloating,
      ID_AllowActivePane,
      ID_TransparentHint,
      ID_TransparentHintFade,
      ID_TransparentDrag,
      ID_About,
      ID_FirstPerspective = ID_CreatePerspective+1000
   };
public:
   MyFrame(wxWindow* parent,
      wxWindowID id,
      const wxString& title,
      const wxPoint& pos = wxDefaultPosition,
      const wxSize& size = wxDefaultSize,
      long style = wxDEFAULT_FRAME_STYLE | wxSUNKEN_BORDER);
   ~MyFrame();
private:
   wxTextCtrl* CreateTextCtrl();
   wxTreeCtrl* CreateTreeCtrl();
   wxPoint GetStartPosition();
private:
   void OnEraseBackground(wxEraseEvent& event);
   void OnSize(wxSizeEvent& event);
   void OnCreateTree(wxCommandEvent& event);
   void OnCreateText(wxCommandEvent& event);
   void OnCreatePerspective(wxCommandEvent& event);
   void OnRestorePerspective(wxCommandEvent& event);
   void OnManagerFlag(wxCommandEvent& event);
   void OnUpdateUI(wxUpdateUIEvent& event);
private:
   wxFrameManager m_mgr;
   wxArrayString m_perspectives;
   wxMenu* m_view_menu;
   DECLARE_EVENT_TABLE();
};
bool MyApp::OnInit()
{
   wxFrame* frame = new MyFrame(NULL,
      wxID_ANY,
      wxT("wxAUI Sample Application"),
      wxDefaultPosition,
      wxSize(800, 600));
   SetTopWindow(frame);
   frame->Show();
   return true;
}
```

```
BEGIN_EVENT_TABLE(MyFrame, wxFrame)
   EVT_ERASE_BACKGROUND(MyFrame::OnEraseBackground)
   EVT_SIZE(MyFrame::OnSize)
   EVT_MENU(MyFrame::ID_CreateTree, MyFrame::OnCreateTree)
   EVT_MENU(MyFrame::ID_CreateText, MyFrame::OnCreateText)
   EVT_MENU(MyFrame::ID_CreatePerspective, MyFrame::OnCreatePerspective)
   EVT_MENU(ID_AllowFloating, MyFrame::OnManagerFlag)
   EVT_MENU(ID_TransparentHint, MyFrame::OnManagerFlag)
   EVT_MENU(ID_TransparentHintFade, MyFrame::OnManagerFlag)
   EVT_MENU(ID_TransparentDrag, MyFrame::OnManagerFlag)
   EVT_MENU(ID_AllowActivePane, MyFrame::OnManagerFlag)
   EVT_UPDATE_UI(ID_AllowFloating, MyFrame::OnUpdateUI)
   EVT_UPDATE_UI(ID_TransparentHint, MyFrame::OnUpdateUI)
   EVT_UPDATE_UI(ID_TransparentHintFade, MyFrame::OnUpdateUI)
   EVT_UPDATE_UI(ID_TransparentDrag, MyFrame::OnUpdateUI)
   EVT_MENU_RANGE(MyFrame::ID_FirstPerspective, MyFrame::ID_
                       FirstPerspective+1000,
      MyFrame::OnRestorePerspective)
END_EVENT_TABLE()


MyFrame::MyFrame(wxWindow* parent,
   wxWindowID id,
   const wxString& title,
   const wxPoint& pos,
   const wxSize& size,
   long style)
   : wxFrame(parent, id, title, pos, size, style)
{
   // indiquer à wxFrameManager de gérer ce cadre
   m_mgr.SetFrame(this);

   // création de l'icône du cadre
   #ifdef __WXMSW__
   SetIcon(wxIcon(wxT("mondrian"), wxBITMAP_TYPE_ICO_RESOURCE, 16, 16));
   #endif

   // création du menu
   wxMenuBar* mb = new wxMenuBar;

   wxMenu* file_menu = new wxMenu;
   file_menu->Append(wxID_EXIT, _("Exit"));

   m_view_menu = new wxMenu;
   m_view_menu->Append(ID_CreateText, _("Create Text Control"));
   m_view_menu->Append(ID_CreateTree, _("Create Tree"));
   m_view_menu->AppendSeparator();
   m_view_menu->Append(ID_CreatePerspective, _("Create Perspective"));
   m_view_menu->AppendSeparator();
   m_view_menu->Append(ID_FirstPerspective, _("Default View"));

   wxMenu* options_menu = new wxMenu;
   options_menu->AppendCheckItem(ID_AllowFloating, _("Allow Floating"));
   options_menu->AppendCheckItem(ID_TransparentHint, _("Transparent Hint"));
   options_menu->AppendCheckItem(ID_TransparentHintFade, _("Transparent Hint
                       Fade-in"));
   options_menu->AppendCheckItem(ID_TransparentDrag, _("Transparent Drag"));
   options_menu->AppendCheckItem(ID_AllowActivePane, _("Allow Active Pane"));

   wxMenu* help_menu = new wxMenu;
   help_menu->Append(ID_About, _("About..."));

   mb->Append(file_menu, _("File"));
   mb->Append(m_view_menu, _("View"));
   mb->Append(options_menu, _("Options"));
   mb->Append(help_menu, _("Help"));
```

```
SetMenuBar(mb);

CreateStatusBar();
GetStatusBar()->SetStatusText(_("Ready"));

// paramétrer la taille minimum du cadre
SetMinSize(wxSize(400,300));

// création de quelques barres d'outils
wxToolBar* tb1 = new wxToolBar(this, -1, wxDefaultPosition, wxDefaultSize,
   wxTB_FLAT | wxTB_NODIVIDER);
tb1->SetToolBitmapSize(wxSize(16,16));

wxBitmap tb1_bmp1 = wxArtProvider::GetBitmap(wxART_QUESTION, wxART_OTHER,
                    wxSize(16,16));
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddSeparator();
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddSeparator();
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->AddTool(101, wxT("Test"), tb1_bmp1);
tb1->Realize();

wxToolBar* tb2 = new wxToolBar(this, -1, wxDefaultPosition, wxDefaultSize,
   wxTB_FLAT | wxTB_NODIVIDER);
tb2->SetToolBitmapSize(wxSize(16,16));

wxBitmap tb2_bmp1 = wxArtProvider::GetBitmap(wxART_FOLDER, wxART_OTHER,
                    wxSize(16,16));
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->AddSeparator();
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->AddTool(101, wxT("Test"), tb2_bmp1);
tb2->Realize();

// ajout d'un panneau central
m_mgr.AddPane(CreateTextCtrl(), wxPaneInfo().Name(wxT("text_content")).
   CenterPane());

// ajout de quelques panneaux positionnés
m_mgr.AddPane(CreateTreeCtrl(), wxPaneInfo().
   Name(wxT("test1")).Caption(wxT("Tree Pane")).
   Left().Layer(1).Position(1));

m_mgr.AddPane(CreateTextCtrl(), wxPaneInfo().
   Name(wxT("test2")).Caption(wxT("Text Pane")).
   Bottom().Layer(1).Position(1));

// ajout de barres d'outils dans le gestionnaire
m_mgr.AddPane(tb1, wxPaneInfo().
   Name(wxT("tb1")).Caption(wxT("Toolbar 1")).
   ToolbarPane().Top().Row(1).
   LeftDockable(false).RightDockable(false));

m_mgr.AddPane(tb2, wxPaneInfo().
   Name(wxT("tb2")).Caption(wxT("Toolbar 1")).
   ToolbarPane().Top().Row(1).Position(1).
   LeftDockable(false).RightDockable(false));
```

```
   // réalisation de perspectives par défaut
   wxString perspective_default = m_mgr.SavePerspective();
   m_perspectives.Add(perspective_default);

   // "exécuter" toutes les modifications réalisées dans wxFrameManager
   m_mgr.Update();
}

MyFrame::~MyFrame()
{
   m_mgr.UnInit();
}

void MyFrame::OnEraseBackground(wxEraseEvent& event)
{
   event.Skip();
}

void MyFrame::OnSize(wxSizeEvent& event)
{
   event.Skip();
}

void MyFrame::OnManagerFlag(wxCommandEvent& event)
{
   unsigned int flag = 0;

   #ifndef __WXMSW__
   if (event.GetId() == ID_TransparentDrag ||
      event.GetId() == ID_TransparentHint ||
      event.GetId() == ID_TransparentHintFade)
   {
      wxMessageBox(wxT("This option is presently only available on wxMSW"));
      return;
   }
   #endif

   switch (event.GetId())
   {
      case ID_AllowFloating: flag = wxAUI_MGR_ALLOW_FLOATING; break;
      case ID_TransparentDrag: flag = wxAUI_MGR_TRANSPARENT_DRAG; break;
      case ID_TransparentHint: flag = wxAUI_MGR_TRANSPARENT_HINT; break;
      case ID_TransparentHintFade: flag = wxAUI_MGR_TRANSPARENT_HINT_FADE; break;
      case ID_AllowActivePane: flag = wxAUI_MGR_ALLOW_ACTIVE_PANE; break;
   }

   m_mgr.SetFlags(m_mgr.GetFlags() ^ flag);
   m_mgr.Update();
}

void MyFrame::OnUpdateUI(wxUpdateUIEvent& event)
{
   unsigned int flags = m_mgr.GetFlags();

   switch (event.GetId())
   {
      case ID_AllowFloating:
         event.Check(flags & wxAUI_MGR_ALLOW_FLOATING ? true : false);
         break;
      case ID_TransparentDrag:
         event.Check(flags & wxAUI_MGR_TRANSPARENT_DRAG ? true : false);
         break;
      case ID_TransparentHint:
         event.Check(flags & wxAUI_MGR_TRANSPARENT_HINT ? true : false);
         break;
      case ID_TransparentHintFade:
         event.Check(flags & wxAUI_MGR_TRANSPARENT_HINT_FADE ? true : false);
```

```
         break;
   }
}

void MyFrame::OnCreatePerspective(wxCommandEvent& event)
{
   wxTextEntryDialog dlg(this, wxT("Enter a name for the new perspective:"),
      wxT("wxAUI Test"));

   dlg.SetValue(wxString::Format(wxT("Perspective %d"), m_perspectives.GetCount
                        ()+1));
   if (dlg.ShowModal() != wxID_OK)
      return;

   if (m_perspectives.GetCount() == 0)
   {
      m_view_menu->AppendSeparator();
   }

   m_view_menu->Append(ID_FirstPerspective + m_perspectives.GetCount(),
                        dlg.GetValue());
   m_perspectives.Add(m_mgr.SavePerspective());
}

void MyFrame::OnRestorePerspective(wxCommandEvent& event)
{
   m_mgr.LoadPerspective(m_perspectives.Item(event.GetId() - ID_
                        FirstPerspective));
}

wxPoint MyFrame::GetStartPosition()
{
   static int x = 0;
   x += 20;
   wxPoint pt = ClientToScreen(wxPoint(0,0));
   return wxPoint(pt.x + x, pt.y + x);
}

void MyFrame::OnCreateTree(wxCommandEvent& event)
{
   m_mgr.AddPane(CreateTreeCtrl(), wxPaneInfo().
      Name(wxT("Test")).Caption(wxT("Tree Control")).
      Float().FloatingPosition(GetStartPosition()).
      FloatingSize(wxSize(150,300)));
   m_mgr.Update();
}

void MyFrame::OnCreateText(wxCommandEvent& event)
{
   m_mgr.AddPane(CreateTextCtrl(), wxPaneInfo().
      Name(wxT("Test")).Caption(wxT("Text Control")).
      Float().FloatingPosition(GetStartPosition()));
   m_mgr.Update();
}

wxTextCtrl* MyFrame::CreateTextCtrl()
{
   wxString text;
   static int n = 0;

   text.Printf(wxT("This is text box %d"), ++n);

   return new wxTextCtrl(this,-1, text,
      wxPoint(0,0), wxSize(150,90),
      wxNO_BORDER | wxTE_MULTILINE);
}

wxTreeCtrl* MyFrame::CreateTreeCtrl()
```

```
{
   wxTreeCtrl* tree = new wxTreeCtrl(this, -1,
      wxPoint(0,0), wxSize(160,250),
      wxTR_DEFAULT_STYLE | wxNO_BORDER);

   wxTreeItemId root = tree->AddRoot(wxT("wxAUI Project"));
   wxArrayTreeItemIds items;

   wxImageList* imglist = new wxImageList(16, 16, true, 2);
   imglist->Add(wxArtProvider::GetBitmap(wxART_FOLDER, wxART_OTHER,
                     wxSize(16,16)));
   imglist->Add(wxArtProvider::GetBitmap(wxART_NORMAL_FILE, wxART_OTHER,
                     wxSize(16,16)));
   tree->AssignImageList(imglist);

   items.Add(tree->AppendItem(root, wxT("Item 1"), 0));
   items.Add(tree->AppendItem(root, wxT("Item 2"), 0));
   items.Add(tree->AppendItem(root, wxT("Item 3"), 0));
   items.Add(tree->AppendItem(root, wxT("Item 4"), 0));
   items.Add(tree->AppendItem(root, wxT("Item 5"), 0));

   int i, count;
   for (i = 0, count = items.Count(); i < count; ++i)
   {
      wxTreeItemId id = items.Item(i);
      tree->AppendItem(id, wxT("Subitem 1"), 1);
      tree->AppendItem(id, wxT("Subitem 2"), 1);
      tree->AppendItem(id, wxT("Subitem 3"), 1);
      tree->AppendItem(id, wxT("Subitem 4"), 1);
      tree->AppendItem(id, wxT("Subitem 5"), 1);
   }

   tree->Expand(root);

   return tree;
}
```