```cpp

// MainFrm.cpp : implementation of the CMainFrame class
//

#include "pch.h"
#include "framework.h"
#include "DesignArk.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWndEx)

const int  iMaxUserToolbars = 10;
const UINT uiFirstUserToolBarId = AFX_IDW_CONTROLBAR_FIRST + 40;
const UINT uiLastUserToolBarId = uiFirstUserToolBarId +
  iMaxUserToolbars - 1;

BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWndEx)
    ON_WM_CREATE()
    ON_COMMAND(ID_WINDOW_MANAGER, &CMainFrame::OnWindowManager)
    ON_COMMAND(ID_VIEW_CUSTOMIZE, &CMainFrame::OnViewCustomize)
    ON_REGISTERED_MESSAGE(AFX_WM_CREATETOOLBAR,
      &CMainFrame::OnToolbarCreateNew)
    ON_COMMAND_RANGE(ID_VIEW_APPLOOK_WIN_2000,
      ID_VIEW_APPLOOK_WINDOWS_7, &CMainFrame::OnApplicationLook)
    ON_UPDATE_COMMAND_UI_RANGE(ID_VIEW_APPLOOK_WIN_2000,
      ID_VIEW_APPLOOK_WINDOWS_7,
      &CMainFrame::OnUpdateApplicationLook)
END_MESSAGE_MAP()

static UINT indicators[] =
{
    ID_SEPARATOR,           // status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM
};

// CMainFrame construction/destruction

CMainFrame::CMainFrame() noexcept
{
    // TODO: add member initialization code here
    theApp.m_nAppLook = theApp.GetInt(_T("ApplicationLook"),
      ID_VIEW_APPLOOK_VS_2008);
}
CMainFrame::~CMainFrame()
{
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
```

```
51  {
52      if( !CMDIFrameWndEx::PreCreateWindow(cs) )
53          return FALSE;
54      // TODO: Modify the Window class or styles here by modifying
55      //  the CREATESTRUCT cs
56
57      return TRUE;
58  }
59  BOOL CMainFrame::LoadFrame(UINT nIDResource, DWORD dwDefaultStyle,
      CWnd* pParentWnd, CCreateContext* pContext)
60  {
61      // base class does the real work
62
63      if (!CMDIFrameWndEx::LoadFrame(nIDResource, dwDefaultStyle,
          pParentWnd, pContext))
64      {
65          return FALSE;
66      }
67
68
69      // enable customization button for all user toolbars
70      BOOL bNameValid;
71      CString strCustomize;
72      bNameValid = strCustomize.LoadString(IDS_TOOLBAR_CUSTOMIZE);
73      ASSERT(bNameValid);
74
75      for (int i = 0; i < iMaxUserToolbars; i ++)
76      {
77          CMFCToolBar* pUserToolbar = GetUserToolBarByIndex(i);
78          if (pUserToolbar != nullptr)
79          {
80              pUserToolbar->EnableCustomizeButton(TRUE,
                  ID_VIEW_CUSTOMIZE, strCustomize);
81          }
82      }
83
84      return TRUE;
85  }
86
87  #ifdef _DEBUG
88  void CMainFrame::AssertValid() const
89  {
90      CMDIFrameWndEx::AssertValid();
91  }
92  void CMainFrame::Dump(CDumpContext& dc) const
93  {
94      CMDIFrameWndEx::Dump(dc);
95  }
96  #endif //_DEBUG
97
98  int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
99  {
100     if (CMDIFrameWndEx::OnCreate(lpCreateStruct) == -1)
101         return -1;
102
103     BOOL bNameValid;
```

```
104     CString strCustomize;
105
106     init_tabs();
107     init_menubar();
108     init_toolbar();
109     init_statusbar();
110     init_dockingwindows();
111     init_customisation();
112     init_userimages();
113     init_commands();
114
115     return 0;
116 }
117 void CMainFrame::OnWindowManager()
118 {
119     ShowWindowsDialog();
120 }
121 void CMainFrame::OnViewCustomize()
122 {
123     CMFCToolBarsCustomizeDialog* pDlgCust = new
        CMFCToolBarsCustomizeDialog(this, TRUE /* scan menus */);
124     pDlgCust->EnableUserDefinedToolbars();
125     pDlgCust->Create();
126 }
127 LRESULT CMainFrame::OnToolbarCreateNew(WPARAM wp,LPARAM lp)
128 {
129     LRESULT lres = CMDIFrameWndEx::OnToolbarCreateNew(wp,lp);
130     if (lres == 0)
131     {
132         return 0;
133     }
134
135     CMFCToolBar* pUserToolbar = (CMFCToolBar*)lres;
136     ASSERT_VALID(pUserToolbar);
137
138     BOOL bNameValid;
139     CString strCustomize;
140     bNameValid = strCustomize.LoadString(IDS_TOOLBAR_CUSTOMIZE);
141     ASSERT(bNameValid);
142
143     pUserToolbar->EnableCustomizeButton(TRUE, ID_VIEW_CUSTOMIZE,
        strCustomize);
144     return lres;
145 }
146 void CMainFrame::OnApplicationLook(UINT id)
147 {
148     CWaitCursor wait;
149
150     theApp.m_nAppLook = id;
151
152     switch (theApp.m_nAppLook)
153     {
154     case ID_VIEW_APPLOOK_WIN_2000:
155         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
            (CMFCVisualManager));
156         break;
```

```
157
158     case ID_VIEW_APPLOOK_OFF_XP:
159         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
              (CMFCVisualManagerOfficeXP));
160         break;
161
162     case ID_VIEW_APPLOOK_WIN_XP:
163         CMFCVisualManagerWindows::m_b3DTabsXPTheme = TRUE;
164         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
              (CMFCVisualManagerWindows));
165         break;
166
167     case ID_VIEW_APPLOOK_OFF_2003:
168         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
              (CMFCVisualManagerOffice2003));
169         CDockingManager::SetDockingMode(DT_SMART);
170         break;
171
172     case ID_VIEW_APPLOOK_VS_2005:
173         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
              (CMFCVisualManagerVS2005));
174         CDockingManager::SetDockingMode(DT_SMART);
175         break;
176
177     case ID_VIEW_APPLOOK_VS_2008:
178         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
              (CMFCVisualManagerVS2008));
179         CDockingManager::SetDockingMode(DT_SMART);
180         break;
181
182     case ID_VIEW_APPLOOK_WINDOWS_7:
183         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
              (CMFCVisualManagerWindows7));
184         CDockingManager::SetDockingMode(DT_SMART);
185         break;
186
187     default:
188         switch (theApp.m_nAppLook)
189         {
190         case ID_VIEW_APPLOOK_OFF_2007_BLUE:
191             CMFCVisualManagerOffice2007::SetStyle
                  (CMFCVisualManagerOffice2007::Office2007_LunaBlue);
192             break;
193
194         case ID_VIEW_APPLOOK_OFF_2007_BLACK:
195             CMFCVisualManagerOffice2007::SetStyle
                  (CMFCVisualManagerOffice2007::Office2007_ObsidianBlack
                  );
196             break;
197
198         case ID_VIEW_APPLOOK_OFF_2007_SILVER:
199             CMFCVisualManagerOffice2007::SetStyle
                  (CMFCVisualManagerOffice2007::Office2007_Silver);
200             break;
201
202         case ID_VIEW_APPLOOK_OFF_2007_AQUA:
```

```
203             CMFCVisualManagerOffice2007::SetStyle
                  (CMFCVisualManagerOffice2007::Office2007_Aqua);
204             break;
205         }
206
207         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
              (CMFCVisualManagerOffice2007));
208         CDockingManager::SetDockingMode(DT_SMART);
209     }
210
211     RedrawWindow(nullptr, nullptr, RDW_ALLCHILDREN | RDW_INVALIDATE
          | RDW_UPDATENOW | RDW_FRAME | RDW_ERASE);
212
213     theApp.WriteInt(_T("ApplicationLook"), theApp.m_nAppLook);
214 }
215 void CMainFrame::OnUpdateApplicationLook(CCmdUI* pCmdUI)
216 {
217     pCmdUI->SetRadio(theApp.m_nAppLook == pCmdUI->m_nID);
218 }
219
220 void CMainFrame::OnSetActiveView(CView* pViewNew, BOOL bNotify)
221 {
222     CFrameWnd::SetActiveView(pViewNew, bNotify);
223 }
224
225 // Private Initialisers
226 int CMainFrame::init_tabs()
227 {
228     CMDITabInfo mdiTabParams;
229     mdiTabParams.m_style = CMFCTabCtrl::STYLE_3D_ONENOTE;
230     mdiTabParams.m_bActiveTabCloseButton = TRUE;
231     mdiTabParams.m_bDocumentMenu = TRUE;
232     mdiTabParams.m_bEnableTabSwap = TRUE;
233     mdiTabParams.m_bFlatFrame = TRUE;
234     mdiTabParams.m_bTabCloseButton = TRUE;
235     mdiTabParams.m_bTabCustomTooltips = FALSE;
236     mdiTabParams.m_bTabIcons = FALSE;
237     //mdiTabParams.m_nTabBorderSize =
          CMFCVisualManager::GetMDITabsBordersSize;
238     mdiTabParams.m_tabLocation =
          CMFCBaseTabCtrl::Location::LOCATION_TOP;
239     this->EnableMDITabbedGroups(TRUE, mdiTabParams);
240
241     return 0;
242 }
243 int CMainFrame::init_menubar()
244 {
245     if (!this->m_wndMenuBar.Create(this))
246     {
247         TRACE0("Failed to create menubar\n");
248         return -1;      // fail to create
249     }
250
251     this->m_wndMenuBar.SetPaneStyle(this->m_wndMenuBar.GetPaneStyle
          () | CBRS_SIZE_DYNAMIC | CBRS_TOOLTIPS | CBRS_FLYBY);
252
```

```
253     // prevent the menu bar from taking the focus on activation
254     CMFCPopupMenu::SetForceMenuFocus(FALSE);
255
256     return 0;
257 }
258 int CMainFrame::init_toolbar()
259 {
260     BOOL bNameValid;
261     if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD |
          WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER | CBRS_TOOLTIPS |
          CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
262         !m_wndToolBar.LoadToolBar(theApp.m_bHiColorIcons ?
            IDR_MAINFRAME_256 : IDR_MAINFRAME))
263     {
264         TRACE0("Failed to create toolbar\n");
265         return -1;      // fail to create
266     }
267
268     CString strToolBarName;
269     bNameValid = strToolBarName.LoadString(IDS_TOOLBAR_STANDARD);
270     ASSERT(bNameValid);
271     m_wndToolBar.SetWindowText(strToolBarName);
272
273     CString strCustomize;
274     bNameValid = strCustomize.LoadString(IDS_TOOLBAR_CUSTOMIZE);
275     ASSERT(bNameValid);
276     m_wndToolBar.EnableCustomizeButton(TRUE, ID_VIEW_CUSTOMIZE,
          strCustomize);
277
278     // Allow user-defined toolbars operations:
279     InitUserToolbars(nullptr, uiFirstUserToolBarId,
          uiLastUserToolBarId);
280
281     return 0;
282 }
283 int CMainFrame::init_statusbar()
284 {
285     if (!this->m_wndStatusBar.Create(this))
286     {
287         TRACE0("Failed to create status bar\n");
288         return -1;      // fail to create
289     }
290     this->m_wndStatusBar.SetIndicators(indicators, sizeof
          (indicators) / sizeof(UINT));
291
292     return 0;
293 }
294 int CMainFrame::init_dockingwindows()
295 {
296     // TODO: Delete these five lines if you don't want the toolbar
          and menubar to be dockable
297     this->DockPane(&this->m_wndMenuBar);
298     this->DockPane(&this->m_wndToolBar);
299     //this->m_wndMenuBar.EnableDocking(CBRS_ALIGN_ANY);
300     //this->m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
301     //this->EnableDocking(CBRS_ALIGN_ANY);
```

```
302
303     // enable Visual Studio 2005 style docking window behavior
304     CDockingManager::SetDockingMode(DT_SMART);
305     // enable Visual Studio 2005 style docking window auto-hide
          behavior
306     this->EnableAutoHidePanes(CBRS_ALIGN_ANY);
307
308     return 0;
309 }
310 int CMainFrame::init_customisation()
311 {
312     BOOL bNameValid;
313     CString strCustomize;
314     bNameValid = strCustomize.LoadString(IDS_TOOLBAR_CUSTOMIZE);
315     ASSERT(bNameValid);
316
317     // set the visual manager and style based on persisted value
318     this->OnApplicationLook(theApp.m_nAppLook);
319
320     // Enable enhanced windows management dialog
321     this->EnableWindowsDialog(ID_WINDOW_MANAGER, ID_WINDOW_MANAGER,
          TRUE);
322
323     // Enable toolbar and docking window menu replacement
324     this->EnablePaneMenu(TRUE, ID_VIEW_CUSTOMIZE, strCustomize,
          ID_VIEW_TOOLBAR);
325
326     // enable quick (Alt+drag) toolbar customization
327     CMFCToolBar::EnableQuickCustomization();
328
329     // Switch the order of document name and application name on the
         window title bar. This
330     // improves the usability of the taskbar because the document
          name is visible with the thumbnail.
331     this->ModifyStyle(0, FWS_PREFIXTITLE);
332
333     return 0;
334 }
335 int CMainFrame::init_userimages()
336 {
337     // Load menu item image (not placed on any standard toolbars):
338     CMFCToolBar::AddToolBarForImageCollection(IDR_MENU_IMAGES,
          theApp.m_bHiColorIcons ? IDB_MENU_IMAGES_24 : 0);
339
340     if (CMFCToolBar::GetUserImages() == nullptr)
341     {
342         // load user-defined toolbar images
343         if (this->m_UserImages.Load(_T(".\\UserImages.bmp")))
344         {
345             CMFCToolBar::SetUserImages(&m_UserImages);
346         }
347     }
348
349     return 0;
350 }
351 int CMainFrame::init_commands()
```

```
352 {
353     CList<UINT, UINT> lstBasicCommands;
354 
355     lstBasicCommands.AddTail(ID_FILE_NEW);
356     lstBasicCommands.AddTail(ID_FILE_OPEN);
357     lstBasicCommands.AddTail(ID_FILE_SAVE);
358     lstBasicCommands.AddTail(ID_FILE_PRINT);
359     lstBasicCommands.AddTail(ID_APP_EXIT);
360     lstBasicCommands.AddTail(ID_EDIT_CUT);
361     lstBasicCommands.AddTail(ID_EDIT_PASTE);
362     lstBasicCommands.AddTail(ID_EDIT_UNDO);
363     lstBasicCommands.AddTail(ID_APP_ABOUT);
364     lstBasicCommands.AddTail(ID_VIEW_STATUS_BAR);
365     lstBasicCommands.AddTail(ID_VIEW_TOOLBAR);
366     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_OFF_2003);
367     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_VS_2005);
368     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_OFF_2007_BLUE);
369     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_OFF_2007_SILVER);
370     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_OFF_2007_BLACK);
371     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_OFF_2007_AQUA);
372     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_WINDOWS_7);
373     lstBasicCommands.AddTail(ID_SORTING_SORTALPHABETIC);
374     lstBasicCommands.AddTail(ID_SORTING_SORTBYTYPE);
375     lstBasicCommands.AddTail(ID_SORTING_SORTBYACCESS);
376     lstBasicCommands.AddTail(ID_SORTING_GROUPBYTYPE);
377 
378     CMFCToolBar::SetBasicCommands(lstBasicCommands);
379 
380     return 0;
381 }
382 
```