Writing MFC-based dialogs for MicroStation V8

MSM ONLINE

By Stanislav Sumbera

MicroStation V8 provides strong support for writing applications in native form. This makes it possible to write and debug C/C++ applications inside an Integrated Development Environment (IDE), such as Visual C++. This ability will likely encourage developers to write graphical user interface modifications in native code. That raises a question: How does one effectively write native GUI items for the MicroStation environment? This article offers a short introduction on using native GUI in MicroStation.

The Windows Application Programming Interface (Win32 API) provides a rich set of user interface functions. These functions are very similar to traditional mdlDialog_ functions for use with windows, dialogs, items, etc. A large portion of the Win32 API was encapsulated into classes called MFC—Microsoft Foundation Classes. The MFCs make up an application framework based on C++, a framework on which you can build your native user interface.

Visual C++ provides several utilities for rapid application development: Application Wizard for starting projects; Class Wizard to help manage MFC classes; resource editors to visually create user interface and more. It is beyond the scope of this article to describe these utilities in detail. If you need to learn more, check out MSDN, the Microsoft Developer Network.

Instead, let's go step by step to create a simple non-modal dialog for MicroStation using Visual C++ v.6.0 and MFC. First, create a new project called **nativeDlg** using MFC AppWizard (dll). In Wizard Options, choose "Regular DLL using shared MFC DLL." This will give you the skeleton of your native DLL code using MFC libraries.

Next, create a simple window class called CNativeFrame using CWnd as the base class. This step can be done simply in Class Wizard by pressing the "Add Class" button. The class will provide a frame window for all other native dialogs or items.

To open the frame window, you will need an export function, which will be called by an MDL invoker:

```
extern "C"
{
// exported DLL function for opening dialog box
__declspec(dllexport) int OpenDialog(void){
  AFX_MANAGE_STATE(AfxGetStaticModuleState());
  HWND parent = frameWin.OpenFrame("Frame
Window",400,180)
  return true;
}}
```

The OpenFrame method is defined in class CNativeFrame and creates a window using methods of base CWnd class AfxRegisterWndClass and CreateEx. Your MDL invoker could be very simple, you need only to import the function OpenDialog via dynamic link specification, declare it as nativeCode in MDL source and call it in the main function. Now run your code, and you should see your non-modal frame window.

You may not be satisfied with the behavior of the window—it has no parent window, it doesn't behave as a child MicroStation window. To solve the problem, you must set the parent window of your frame window to the MicroStation main window. The function used in the example code to get the MicroStation main window enumerates all created windows, checks if the window is a MicroStation one and compare process ID of the founded window with the process ID of your DLL. If all criteria are fulfilled, the function returns the MicroStation main window.

Unfortunately, another problem may come concerning minimization and maximization of the frame window. When a window is minimized, it disappears from the visible area of the MicroStation window. If a window is maximized, it covers the whole area including the menu and status bars. This behavior can be easily changed in the window message event handler WM_SIZE.

```
void CNativeFrame::OnSize(UINT nType, int cx, int cy)
{
  if (nType == SIZE_MINIMIZED)
  MoveWindow(...);
  else if (nType == SIZE_MAXIMIZED)
  MoveWindow(...);
}
```

Now you have a nice MFC frame window but without any items. This would be the second task, to create a dialog and items using Resource Editor and Class Wizard to join dialog resource with the new created class CNativeBox. To display the dialog correctly, set its parent to frame window:

```
//..in OpenDialog function
HWND parent = frameWin.OpenFrame("Frame Window",400,180)
box.Create(IDD_NATIVE_DIALOG,CWnd::FromHandle(parent));
return box.ShowWindow(SW_SHOW);
```

In this version, the TAB key between dialog items doesn't work. The solution to this problem is well described on MSDN. For a modeless dialog box to process a TAB key, the message pump needs to call the IsDialogMessage API. However, if you are writing a DLL and do not have access to the application's source code, you cannot modify the message pump to do this. To work around this problem, use the WH_GETMESSAGE hook to capture the keystroke messages and call the IsDialogMessage API. If IsDialogMessage returns TRUE, then do not pass the message on to the message pump. Set the hook when handling WM_INITDIALOG and unset it when handling the WM_DESTROY message.

```
BOOL CNativeBox::OnInitDialog(){
  CDialog::OnInitDialog();
hHook = ::SetWindowsHookEx(WH_GETMESSAGE,
GetMsgProc, NULL, GetCurrentThreadId());
  return TRUE;
}

void CNativeBox::OnDestroy(){
  UnhookWindowsHookEx(hHook);
  CDialog::OnDestroy();
}

// hook function capturing dialog messages
LRESULT FAR PASCAL GetMsgProc(int nCode, WPARAM wParam,
LPARAM lParam)
{
    LPMSG lpMsgd = (LPMSG) lParam;
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    if( ( nCode >= 0 && PM_REMOVE == wParam ) &&
         (lpMsg->message >= WM_KEYFIRST &&
       lpMsg->message <= WM_KEYLAST) &&
     (IsDialogMessage((HWND)box.m_hWnd, lpMsg))) {
                lpMsg->message = WM_NULL;
                lpMsg->lParam  = 0;
                lpMsg->wParam  = 0;
       }
    return CallNextHookEx(hHook, nCode, wParam, lParam);
}
```

You may also add into the dialog any ActiveX component or call, via OLE automation, MicroStation methods available in Visual Basic for Applications. All you need to do is enable OLE in your DLL, add the new class from the MicroStation template library (ustation.exe) using ClassWizard and call your method:

```
_Application     MSapp;
AfxEnableControlContainer();
OleInitialize(NULL);
MSapp.CreateDispatch("MicroStationDGN.Application",NULL);
MSapp.SetCaption("MicroStation with MFC");
```

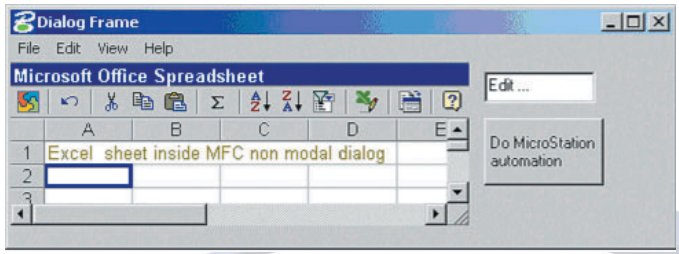

Figure 1. MFC-based non-modal dialog with Excel ActiveX.

Creating non-modal dialogs based on MFC need not be difficult. Utilizing all Windows features for GUI development gives us rich possibilities to extend the current user interface. There are also many ActiveX components we can easily add into MFC dialogs to create rich, user-friendly modern GUIs for MicroStation. The examples given in the article are for learning purposes, so I removed all error handling and exception handling for shorter code. The code was tested on MicroStation v8. 08.00.00.21. msm

Stanislav Sumbera is a MicroStation software developer for Bentley Integrator Berit Group (www.berit.com), specializing in raster and seamless map visualization. The programming code in this article is offered to the MicroStation community "as is," without responsibility for its use elsewhere. You may contact Stanislav Sumbera at sumbera@berit.cz.