Using the MNS_DRAGDROP style: Dragging out

devblogs.microsoft.com/oldnewthing/20111228-00

December 28, 2011

Raymond Chen

Windows 2000 introduced the `MNS_DRAGDROP` menu style, which permits drag/drop operations in a menu. Nobody uses this style, probably because it's totally undiscoverable by the end-user. But I'll write a sample program anyway.

Mind you, I knew nothing about the `MNS_DRAGDROP` menu style until I started writing this entry. But I simply read the documentation, which says that if you set this style, you will receive `WM_MENUDRAG` and `WM_MENUGETOBJECT` messages. The `WM_MENUDRAG` message is sent when the user drags a menu item, so let's go with that first. The documentation says that you get information about the item that was dragged, and then you return a code that specifies whether you want the menu to remain up or whether you want it torn down.

Simple enough. Let's do it.

Start with the scratch program, add the function `GetUIObjectOfFile` and the class `CDrop-Source`, and change the calls to `CoInitialize` and `CoUninitialize` into `Ole-Initialize` and `OleUninitialize`, respectively. Next, define the menu we're going to play with:

```
// resource header file
#define IDM_MAIN 1
#define IDC_CLOCK 100
// resource file
IDM_MAIN MENU PRELOAD
BEGIN
    POPUP "&Test"
    BEGIN
        MENUITEM "&Clock", IDC_CLOCK
    END
END
```

Now we can add some new code to our scratch program. First, we add a menu to our window and enable drag/drop on it:

```
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
 MENUINFO mi = { sizeof(mi), MIM_STYLE, MNS_DRAGDROP };
 return SetMenuInfo(GetMenu(hwnd), &mi);
}
// InitApp
 // wc.lpszMenuName = NULL;
 wc.lpszMenuName = MAKEINTRESOURCE(IDM_MAIN);
```

For both dragging and dropping, we need a way to obtain the COM object associated with a menu item, so I'll put them in this common helper function:

```
HRESULT GetMenuObject(HWND hwnd, HMENU hmenu, UINT uPos,
                      REFIID riid, void **ppvOut)
{
 HRESULT hr = E_NOTIMPL;
 *ppvOut = NULL;
 if (hmenu == GetSubMenu(GetMenu(hwnd), 0)) {
  switch (GetMenuItemID(hmenu, uPos)) {
  case IDC_CLOCK:
   hr = GetUIObjectOfFile(hwnd, L"C:\\Windows\\clock.avi",
                                                riid, ppvOut);
   break;
  }
 }
 return hr;
}
```

If the menu is our "Test" popup menu, then we know how to map the menu items to COM objects. For now, we have only one item, namely *Clock*, which corresponds to the `C:\Windows\clock.avi` [1] file.

Now we can hook up a handler to the `WM_MENUDRAG` message:

```
#define HANDLE_WM_MENUDRAG(hwnd, wParam, lParam, fn) \
 (fn)((hwnd), (UINT)(wParam), (HMENU)(lParam))
LRESULT OnMenuDrag(HWND hwnd, UINT uPos, HMENU hmenu)
{
 LRESULT lres = MND_CONTINUE;
 IDataObject *pdto;
 if (SUCCEEDED(GetMenuObject(hwnd, hmenu, uPos,
                                  IID_PPV_ARGS(&pdto)))) {
  IDropSource *pds = new(std::nothrow) CDropSource();
  if (pds) {
   DWORD dwEffect;
   if (DoDragDrop(pdto, pds, DROPEFFECT_COPY | DROPEFFECT_LINK,
                  &dwEffect) == DRAGDROP_S_DROP) {
    lres = MND_ENDMENU;
   }
   pds->Release();
  }
  pdto->Release();
 }
 return lres;
}
```

This function is where the magic happens, but it's really not all that magical. We get the data object for the menu item being dragged and tell OLE to do a drag/drop operation with it. Just to make things interesting, I'll say that the menu should be dismissed if the user dropped the object somewhere; otherwise, the menu remains on the screen.

Finally, we hook up the message handler to our window procedure:

```
HANDLE_MSG(hwnd, WM_MENUDRAG, OnMenuDrag);
```

And there you have it. A program that calls up a menu with drag enabled. If you drag the item labeled *Clock*, then the drag/drop operation proceeds as if you were dragging the `clock.avi` file.

Next time, we'll look at the drop half of drag and drop.

Footnote

[1] I hard-coded the `clock.avi` file for old time's sake. Yes, I know the file is no longer included with Windows. That'll teach people to use hard-coded paths!

Raymond Chen

Follow