Dragging a shell object, part 1: Getting the IDataObject

devblogs.microsoft.com/oldnewthing/20041206-00

December 6, 2004

Raymond Chen

The shell gives you the `IDataObject`; all you have to do is drag it around. (This is the first of a five-part series.)

Start with the scratch program, and add the function `GetUIObjectOfFile` from an earlier article. Also, change the calls to `CoInitialize` and `CoUninitialize` to `OleInitialize` and `OleUninitialize`, respectively, since we're now going to be using full-on OLE and not just COM.

In order to initiate a drag/drop operation, we need a drop source:

```
class CDropSource : public IDropSource
{
public:
  // *** IUnknown ***
  STDMETHODIMP QueryInterface(REFIID riid, void **ppv);
  STDMETHODIMP_(ULONG) AddRef();
  STDMETHODIMP_(ULONG) Release();

  // *** IDropSource ***
  STDMETHODIMP QueryContinueDrag(BOOL fEscapePressed, DWORD grfKeyState);
  STDMETHODIMP GiveFeedback(DWORD dwEffect);

  CDropSource() : m_cRef(1) { }
private:
  ULONG m_cRef;
};

HRESULT CDropSource::QueryInterface(REFIID riid, void **ppv)
{
  IUnknown *punk = NULL;
  if (riid == IID_IUnknown) {
    punk = static_cast<IUnknown*>(this);
  } else if (riid == IID_IDropSource) {
    punk = static_cast<IDropSource*>(this);
  }

  *ppv = punk;
  if (punk) {
    punk->AddRef();
    return S_OK;
  } else {
    return E_NOINTERFACE;
  }
}

ULONG CDropSource::AddRef()
{
  return ++m_cRef;
}

ULONG CDropSource::Release()
{
  ULONG cRef = --m_cRef;
  if (cRef == 0) delete this;
  return cRef;
}

HRESULT CDropSource::QueryContinueDrag(
          BOOL fEscapePressed, DWORD grfKeyState)
{
  if (fEscapePressed) return DRAGDROP_S_CANCEL;
```

```
  // [Update: missing paren repaired, 7 Dec]
  if (!(grfKeyState & (MK_LBUTTON | MK_RBUTTON)))
    return DRAGDROP_S_DROP;

  return S_OK;
}

HRESULT CDropSource::GiveFeedback(DWORD dwEffect)
{
  return DRAGDROP_S_USEDEFAULTCURSORS;
}
```

As you can see, this drop source is extraordinarily boring. Even the interesting methods are uninteresting.

The `IDropSource::QueryContinueDrag` method is pretty much boilerplate. If the Escape key was pressed, then cancel the drag/drop operation. If the mouse buttons are released, then complete the operation. Otherwise, continue the operation.

The `IDropSource::GiveFeedback` method is even less interesting. It merely returns `DRAGDROP_S_USEDEFAULTCURSORS` to indicate that it wants default drag feedback.

Believe it or not, we now have everything we need to drag a file.

```
void OnLButtonDown(HWND hwnd, BOOL fDoubleClick,
                   int x, int y, UINT keyFlags)
{
  IDataObject *pdto;
  // In a real program of course
  // you wouldn't use a hard-coded path.
  // [comment added 11am because apparently some
  // people thought this wasn't self-evident.]
  if (SUCCEEDED(GetUIObjectOfFile(hwnd,
                   L"C:\\Windows\\clock.avi",
                   IID_IDataObject, (void**)&pdto))) {
    IDropSource *pds = new CDropSource();
    if (pds) {
      DWORD dwEffect;
      DoDragDrop(pdto, pds, DROPEFFECT_COPY | DROPEFFECT_LINK,
                 &dwEffect);
      pds->Release();
    }
    pdto->Release();
  }
}

    HANDLE_MSG(hwnd, WM_LBUTTONDOWN, OnLButtonDown);
```

To drag an object, you need two things, a data object and a drop source. We created our drop source above, and the data object comes from the shell. All that's left to do is start the drag/drop operation by calling the `DoDragDrop` function.

Notice that we specify that the permitted operations are `DROPEFFECT_COPY` and `DROPEFFECT_LINK`. We specifically disallow `DROPEFFECT_MOVE` because this program doesn't present a folder-like window; the user has no expectation that the drag/drop will result in a Move operation.

Next time, adding Move support, just to see how it works.

Raymond Chen

Follow