# Dragging a shell object, part 5: Making somebody else do the heavy lifting

devblogs.microsoft.com/oldnewthing/20041210-00

December 10, 2004

Raymond Chen

Creating that drag image was a bit of work. Fortunately, the listview control is willing to do some of the work for you.

Throw away the `OnLButtonDown` function (and the `HANDLE_MESSAGE` that goes with it). Instead, we'll make the listview do all our presentation for us.

```
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
  g_hwndChild = CreateWindow(WC_LISTVIEW, NULL,
                             WS_CHILD | WS_VISIBLE | LVS_ICON |
                             LVS_SHAREIMAGELISTS, // flag added 13 Dec
                             0, 0, 0, 0,
                             hwnd, (HMENU)1, g_hinst, 0);
  if (!g_hwndChild) return FALSE;


  SHFILEINFOW sfi;
  HIMAGELIST himl = (HIMAGELIST)
    SHGetFileInfoW(g_pszTarget, 0, &sfi, sizeof(sfi),
                   SHGFI_SYSICONINDEX |
                   SHGFI_DISPLAYNAME | SHGFI_LARGEICON);
  if (!himl) return FALSE;


  ListView_SetImageList(g_hwndChild, himl, LVSIL_NORMAL);


  LVITEM item;
  item.iSubItem = 0;
  item.mask = LVIF_TEXT | LVIF_IMAGE;
  item.pszText = sfi.szDisplayName;
  item.iImage = sfi.iIcon;
  if (ListView_InsertItem(g_hwndChild, &item) < 0)
    return FALSE;


  return TRUE;
}
```

We now let the listview control worry about the icon and its text and all the other UI that goes along with it. And we can make the listview worry about the drag image, too.

```
void OnBeginDrag(HWND hwnd, NMLISTVIEW *plv)
{
  IDataObject *pdto;
  if (SUCCEEDED(GetUIObjectOfFile(hwnd, g_pszTarget,
                  IID_IDataObject, (void**)&pdto))) {
    IDragSourceHelper *pdsh;
    if (SUCCEEDED(CoCreateInstance(CLSID_DragDropHelper, NULL,
                   CLSCTX_ALL, IID_IDragSourceHelper, (void**)&pdsh))) {
      pdsh->InitializeFromWindow(g_hwndChild, &plv->ptAction, pdto);
      pdsh->Release();
    }


    IDropSource *pds = new CDropSource();
    if (pds) {
      DWORD dwEffect;
      if (DoDragDrop(pdto, pds, DROPEFFECT_MOVE |
                     DROPEFFECT_COPY | DROPEFFECT_LINK,
                     &dwEffect) == DRAGDROP_S_DROP &&
          (dwEffect & DROPEFFECT_MOVE)) {
        DeleteFileW(g_pszTarget);
      }
      pds->Release();
    }
    pdto->Release();
  }
}


LRESULT OnNotify(HWND hwnd, int idCtrl, NMHDR *pnm)
{
  if (idCtrl == 1) {
    NMLISTVIEW *plv;
    switch (pnm->code) {
    case LVN_BEGINDRAG:
      plv = (NMLISTVIEW*)pnm;
      OnBeginDrag(hwnd, plv);
      break;
    }
  }
  return 0;
}


    HANDLE_MSG(hwnd, WM_NOTIFY, OnNotify);
```

Instead of detecting the drag operation, we let the listview do it and just wait for the `LVN_BEGINDRAG` notification, at which point we get the data object for the file we want to drag and ask the listview to create the drag image by passing its window handle to the `IDragSourceHelper::InitializeFromWindow` method.

The listview control does the work of generating the drag image and setting it into the data object. In our specific case, it may have been a toss-up which way is easier, but if you enable multiple-selection capability in the listview, using the `IDragSourceHelper::InitializeFromWindow` method is a major savings because the listview will do the work of generating the radial gradient alpha channel that you see when dragging multiple files in Explorer.

You may notice some color fringes around the icons generated by the listview. That's because we're using version 5 of the common controls, which doesn't support alpha channels very well. If you switch to version 6, you'll find that the fringes are gone and the icon looks a lot prettier.

That's all for now on the subject of initiating a drag/drop operation. Back to one-day topics for a while.

Raymond Chen

**Follow**