You can drag multiple virtual objects, you know

devblogs.microsoft.com/oldnewthing/20080331-00

March 31, 2008

Raymond Chen

A customer wanted to know how they could find out the directory that the user dropped a file onto. As we already noted, users can drop files onto things other than directories, so the question itself comes with incorrect hidden assumptions. This is another one of those cases where you have to ask the customer, "What are you really trying to do?" They have a problem and solved half of it and are asking you for help with the second half, the part that makes little sense.

In this case, what the customer really wanted to do was create additional supporting files into the directory that the user dropped the file onto. To solve the real problem, all you have to do is add virtual objects to the data object the file being dragged.

Let's illustrate this by adding a second file to our minimal example of dragging a virtual file. Actually, let's make it more interesting. We're going to drag one real file plus one virtual file. Start by adding another file's contents to our list of clipboard formats:

```
  enum {
    DATA_FILEGROUPDESCRIPTOR,
    DATA_FILECONTENTS0,
    DATA_FILECONTENTS1,
    DATA_NUM,
    DATA_INVALID = -1,
  };
```

Of course, we need to initialize the `FORMATETC` for the contents of our new virtual file.

```
CTinyDataObject::CTinyDataObject() : _cRef(1)
{
  SetFORMATETC(&_rgfe[DATA_FILEGROUPDESCRIPTOR],
               RegisterClipboardFormat(CFSTR_FILEDESCRIPTOR));
  SetFORMATETC(&_rgfe[DATA_FILECONTENTS0],
               RegisterClipboardFormat(CFSTR_FILECONTENTS),
               TYMED_ISTREAM, /* lindex */ 0);
  SetFORMATETC(&_rgfe[DATA_FILECONTENTS1],
               RegisterClipboardFormat(CFSTR_FILECONTENTS),
               TYMED_HGLOBAL, /* lindex */ 1);
}
```

We need to add this second file to our `FILEGROUPDESCRIPTOR`. Doing this is trickier because the `FILEGROUPDESCRIPTOR` is a variable-size structure, so we have to declare our own version that has room for two files.

```
// Hard-coded for expository purposes
// (I can't believe I had to write that.)
#define FILETODRAG TEXT("C:\\windows\\clock.avi")
HRESULT CreateFileGroupDescriptor(HGLOBAL *phglob)
{
  union {
     FILEGROUPDESCRIPTOR fgd;
     BYTE buffer[FIELD_OFFSET(FILEGROUPDESCRIPTOR, fgd[2])];
  } u;
  ZeroMemory(&u, sizeof(u));
  u.fgd.cItems = 2;
  // item 0: the file itself
  WIN32_FILE_ATTRIBUTE_DATA wfad;
  if (!GetFileAttributesEx(FILETODRAG, GetFileExInfoStandard,
                           &wfad)) {
   return E_FAIL;
  }
  u.fgd.fgd[0].dwFlags = FD_ATTRIBUTES | FD_CREATETIME |
                 FD_ACCESSTIME | FD_WRITESTIME | FD_FILESIZE;
  u.fgd.fgd[0].dwFileAttributes = wfad.dwFileAttributes;
  u.fgd.fgd[0].ftCreationTime   = wfad.ftCreationTime;
  u.fgd.fgd[0].ftLastAccessTime = wfad.ftLastAccessTime;
  u.fgd.fgd[0].ftLastWriteTime  = wfad.ftLastWriteTime;
  u.fgd.fgd[0].nFileSizeHigh    = wfad.nFileSizeHigh;
  u.fgd.fgd[0].nFileSizeLow     = wfad.nFileSizeLow;
  StringCchCopy(u.fgd.fgd[0].cFileName,
                ARRAYSIZE(u.fgd.fgd[0].cFileName),
                PathFindFileName(FILETODRAG));
  // item 1: The virtual "tag-along" file
  StringCchCopy(u.fgd.fgd[1].cFileName,
                ARRAYSIZE(u.fgd.fgd[0].cFileName),
                TEXT("TagAlong"));
  return CreateHGlobalFromBlob(&u, sizeof(u),
                               GMEM_MOVEABLE, phglob);
}
```

The ad-hoc union declares a block of memory large enough for a `FILEGROUPDESCRIPTOR` that holds two files. File zero is the file we are dragging, and as a courtesy (and in violation of the "doing the absolute least amount of work necessary" that has guided the series), we fill in the file attributes so that when the file is dropped onto Explorer, the resulting file has the right metadata. On the other hand, our virtual file tries to sneak by with as little as possible, providing only the mandatory file name.

The last thing to do is hand out the `FILEGROUPDESCRIPTOR` and the two files when we are asked for them.

```
HRESULT CTinyDataObject::GetData(FORMATETC *pfe, STGMEDIUM *pmed)
{
  ZeroMemory(pmed, sizeof(*pmed));
  switch (_GetDataIndex(pfe)) {
  case DATA_FILEGROUPDESCRIPTOR:
    pmed->tymed = TYMED_HGLOBAL;
    return CreateFileGroupDescriptor(&pmed->hGlobal);
  case DATA_FILECONTENTS0:
    pmed->tymed = TYMED_ISTREAM;
    return SHCreateStreamOnFile(FILETODRAG, STGM_READ,
                                &pmed->pstm);
  case DATA_FILECONTENTS1:
    pmed->tymed = TYMED_HGLOBAL;
    return CreateHGlobalFromBlob("Dummy", 5, GMEM_MOVEABLE,
                                 &pmed->hGlobal);
  }
  return DV_E_FORMATETC;
}
```

There you have it, a data object that consists of a file (`FILETODRAG`) and a virtual file. When the user drops the data object into a folder, both files are dropped into the destination directory.

Raymond Chen

Follow