Drag/drop effects: The little drop information box

devblogs.microsoft.com/oldnewthing/20210513-00

May 13, 2021

Raymond Chen

Explorer's drag/drop effects include a little message information box as part of the drop effect. For example, it will start with the message *Move to ‹folder›*, but if you hold the `Ctrl` key, the message changes to *Copy to ‹folder›*. How can you get a piece of that action?

As we saw last time, the visual drag/drop feedback effects used by Explorer are available for general use, via an object know as the shell drop target helper.

Let's take our scratch program and teach it how to add the fancy information box.

The information box is configured by the `DROPDESCRIPTION` structure which is associated with the `CFSTR_DROPDESCRIPTION` clipboard format.

```
#include <strsafe.h>

HRESULT SetDropDescription(
    IDataObject* dataObject, DROPIMAGETYPE type,
    PCWSTR message = nullptr, PCWSTR insert = nullptr)
{
    wil::unique_hglobal_ptr<DROPDESCRIPTION> data(
        reinterpret_cast<DROPDESCRIPTION*>(
            GlobalAlloc(GPTR, sizeof(DROPDESCRIPTION))));
    RETURN_IF_NULL_ALLOC(data);
    data->type = type;
    RETURN_IF_FAILED(StringCchCopyEx(data->szMessage, ARRAYSIZE(data->szMessage),
        message, nullptr, nullptr, STRSAFE_IGNORE_NULLS));
    RETURN_IF_FAILED(StringCchCopyEx(data->szInsert, ARRAYSIZE(data->szInsert),
        insert, nullptr, nullptr, STRSAFE_IGNORE_NULLS));
    FORMATETC fmte = { (CLIPFORMAT)RegisterClipboardFormat(CFSTR_DROPDESCRIPTION),
        nullptr, DVASPECT_CONTENT, -1, TYMED_HGLOBAL };
    STGMEDIUM med = { TYMED_HGLOBAL };
    med.hGlobal = data.get();
    RETURN_IF_FAILED(dataObject->SetData(&fmte, &med, true));
    data.release();
    return S_OK;
}
```

We allocate a `DROPDESCRIPTION` structure as an `HGLOBAL` and copy the drop image type, optional message, and optional message insertion into the structure before setting the data into the data object.

You can read the documentation for `DROPDESCRIPTION` for details on how the message and message insertion interact. For this example, we'll just use a simple message with no insertions.

```
struct SimpleDropTarget
{
    ...

    STDMETHOD DragLeave()
    {
        if (dto) SetDropDescription(dto.get(), DROPIMAGE_INVALID);
        dto = nullptr;
        return helper->DragLeave();
    }


    HRESULT CalculateFeedback(
        DWORD grfKeyState,
        DWORD* pdwEffect)
    {
        if (grfKeyState & MK_CONTROL) {
            *pdwEffect = DROPEFFECT_COPY;
            SetDropDescription(dto.get(), DROPIMAGE_COPY,
                L"Copy with awesomesauce.");
        }
        else {
            *pdwEffect = DROPEFFECT_MOVE;
            SetDropDescription(dto.get(), DROPIMAGE_MOVE,
                L"Move with awesomesauce.");
        }
        return S_OK;
    }

};
```

When we calculate the feedback, we also set the drop description to tell the user what will happen when the drop happens. We also clear the drop description (by setting it to "invalid" with an empty message)[1] when the user decided not to drop it onto our drop target after all. This ensures that our custom description doesn't hang around and accidentally get shown for other drop targets that don't set a description.

So there you have it, drag feedback done the same way that Explorer does it.

[1] There is no way to remove data from a data object, so the best you can do is replace the data with new data that says "Nevermind."

Raymond Chen

Follow