The system manages the system image lists; don't go changing the art on the walls

devblogs.microsoft.com/oldnewthing/20170224-00

February 24, 2017

Raymond Chen

A customer reported that their program ran on Windows XP just fine, but on Windows 7, its icons are all black boxes. They included a code fragment to demonstrate:

```
CBitmap *pBmp = new CBitmap;
pBmp->LoadBitmap(IDB_BITMAP);

CImageList m_ImageList;

SHFILEINFO sfi;
HIMAGELIST hHandle = (HIMAGELIST)SHGetFileInfo(_T(".txt"),
    FILE_ATTRIBUTE_NORMAL, &sfi, sizeof(sfi),
    SHGFI_USEFILEATTRIBUTES | SHGFI_SYSICONINDEX | SHGFI_LARGEICON);
m_ImageList.Attach(hHandle);
int n = m_ImageList.Add(pBmp, RGB(0,255,0));

m_listctrl.SetImageList(&m_ImageList, LVSIL_NORMAL);

m_listctrl.InsertItem(0, _T("Sample"), n);
```

Okay, there are multiple things wrong here, but they all boil down to the same root cause: Modifying the system image list.

The system image lists are managed by the system. It adds images to the system image list when you do things like call `SHGetFileInfo` with the `SHGFI_SYSICONINDEX` flag, or if you host an Explorer Browser control or open a common file dialog. The system needs to keep all of the system image lists in sync, so that an index refers to the same image across all of the system image lists. It also needs to know how to regenerate each of those images, in case it needs to resize the images.

If you go in and make changes to the system image list (say, by adding new images), this bypasses all of the internal shell bookkeeping and gets everything out of sync. Once that happens, the results are unpredictable.

It so happens that Windows XP was far more forgiving of misuse of the system image lists, but optimizations introduced in Windows Vista made it less tolerant of abuse.

(The other bug here is that when the `m_ImageList` destructs, it will destroy the attached image list, which means destroying the system image list. This also tends not to end well, but my guess is that the customer never noticed because the `m_ImageList` was not destructed until the program was exiting anyway.)

The correct thing to do in the customer's scenario above is to create your own image list and put your custom images there.

Of course, if the image list had consisted entirely of app-provided bitmaps, then the customer probably would have done this in the first place. I suspect they are adding custom images to the system image list because they want their list view to be able to display both custom images and system images. In that case, it should create a custom image list, and put its custom images there. If it also wants to show a system image in the list view control, it should copy that image into their custom image list.

Raymond Chen

Follow