How do I override the default icon selection algorithm?

devblogs.microsoft.com/oldnewthing/20121005-00

October 5, 2012

Raymond Chen

The usual way of loading an icon from a resource is to use the `LoadIcon` or `LoadImage` function and letting the window manager's icon selection algorithm decide which image to use based on the images available in the icon, the desired icon size, and the current color depth. A customer wanted to override that algorithm, because the window manager uses the current display color depth to select an image, but they were obtaining the icon for printing purposes, so they wanted to get the highest-color-quality icon rather than the one that matched the screen's color depth. How do you override the default algorithm?

You basically do the same thing the window manager does. As we saw earlier, icon resources are actually stored in multiple pieces. The thing you use to talk about icons is actually the *icon directory*, which in turn points to a set of images. The first step, then, is to obtain the icon directory.

```
HRSRC hrsrcIcon = FindResource(hResources,
                    MAKEINTRESOURCE(IDI_MY_ICON), RT_GROUP_ICON);
HGLOBAL hIcon = LoadResource(hResources, hrsrcIcon);
auto lpIcon = static_cast<GRPICONDIR *>(LockResource(hIcon));
```

You then take the images listed in the `GRPICONDIR` and apply your custom algorithm to decide which one you like best. (If you want to use the default algorithm, you can call `LookupIconIdFromDirectory` or `LookupIconIdFromDirectoryEx`. But if you want to use the default algorithm, then just use `LoadImage` already!)

When you've found the image you like, take the `nId`, and that's the resource ID for the `RT_ICON`.

```
HRSRC hrsrcImage = FindResource(hResources,
                    MAKEINTRESOURCE(nId), RT_ICON);
HGLOBAL hImage = LoadResource(hResources, hrsrcImage);
auto lpImage = static_cast<PBYTE>(LockResource(hImage));
```

You can then convert the icon image data into an icon by using the `CreateIconFromResource` or `CreateIconFromResourceEx` function.

Raymond Chen

Follow