What are the dire consequences of not selecting objects out of my DC?

devblogs.microsoft.com/oldnewthing/20130306-00

March 6, 2013

Raymond Chen

The convention when working with device contexts is to restore them to the way you found them. If a drawing function selects a bitmap into a device context, then it should select the original bitmap into the device context before returning. Same for fonts, pens, all that stuff.

But what if you decide to violate that convention? For example, maybe you create a memory DC, select a bitmap into it, and just leave the bitmap selected there, selecting it out only when you get around to destroying the DC. Is that really so bad?

It sort of depends.

The danger of leaving objects selected into a DC for an extended period of time is that the owner of the object won't be able to destroy the object, because you can't destroy objects while they are selected into a DC. For example, if you select a font into a DC, and somebody tries to destroy the font, the `DeleteObject` call will fail, and you end up leaking a font.

Bitmaps can be selected into only one DC at a time. If you select the bitmap into your DC and just forget about it, then the owner of that bitmap won't be able to select it into any other DC.

Now, if the objects you are selecting into the DC are all under your control, then you can leave them selected into your private DC, because you will know how to get them out if you need to.

Remember that this "leave it lying around, I'll clean it up later" technique requires you to control both the vertical and the horizontal. We've been discussing what happens if you select an object that somebody else controls into your private DC and leave it there. Conversely, if you have a bitmap that you control and leave it selected into a DC that you don't control, then you've got the same sort of problem in reverse: You won't be able to select the bitmap back out of that DC when you need to, because you lost control of the DC.

Bonus chatter: "I've noticed that sometimes, `DeleteObject` claims to succeed even though it actually failed because the object is still selected in a DC." The GDI folks found that a lot of people mess up and try to destroy objects while they are still selected into DCs.

Failing the call caused two categories of problems: Some applications simply leaked resources (since they thought they were destroying the object, but weren't). Other applications checked the return value and freaked out if they saw that `DeleteObject` didn't actually delete the object.

To keep both of these types of applications happy, GDI will sometimes (not always) lie and say, "Sure, I deleted your object." It didn't actually delete it, because it's still selected into a DC, but it also ties a string around its finger, and when the object is finally deselected, GDI will say, "Oh, wait, I was supposed to delete this object," and perform the deletion. So the lie that GDI made wasn't so much a lie as it was an "optimistic prediction of the future."

Raymond Chen

Follow