The mysterious stock bitmap: There's no way to summon it, but it shows up in various places

devblogs.microsoft.com/oldnewthing/20100416-00

April 16, 2010

Raymond Chen

A number of stock GDI objects are made available by the `GetStockObject` function, but one stock GDI object that is mysteriously missing is the stock bitmap. You can't summon the stock bitmap, but it manages to show up in various places, some of them perhaps unexpected.

The stock bitmap is a monochrome 1×1 bitmap which GDI uses in various places where it has to produce a `HBITMAP` even though there really isn't any bitmap worth speaking of. In other words, it's used when GDI has to return *something* but would rather return *nothing*.

- When you create a memory DC, the current bitmap selected into it is the stock bitmap.
- When you create a metafile, the current bitmap is the stock bitmap.

Every DC and metafile has a current bitmap (which you can retrieve with `GetCurrent-Object`), but when GDI creates a brand new DC or metafile, it doesn't know what bitmap the program is going to pass to `SelectObject`—after all, predicting the future has yet to be perfected. As a placeholder, it sticks in the dummy static bitmap.

There has to be a bitmap (as opposed to just leaving it `NULL`), because the `SelectObject` function returns the previous object or `NULL` on failure, so there needs to be a way to tell the difference between "I wasn't able to select the bitmap you requested" and "I was able to select the bitmap you requested, but there was no old bitmap." Returning `NULL` would also break the common coding pattern:

```
// select the new bitmap and save the old one
HBITMAP hbmPrev = SelectObject(hdc, hbmNew);
    ... do something with hdc ...
// all done - restore the original bitmap
SelectObject(hdc, hbmPrev);
```

If `SelectObject` had returned `NULL` when there was no bitmap previously selected into the DC, then the attempt to restore the original bitmap would fail. (Because GDI can't tell whether you passed it a `(HBITMAP)NULL` or a `(HBRUSH)NULL` or a `(HPEN)NULL` or...)

Normally, a single bitmap cannot be selected into more than one DC, but the stock bitmap has the magical power that it can be selected into multiple DCs at once. Without this magical power, GDI would have to create a different dummy bitmap to select into each newly-created DC and carry it around so that it can be selected back into the DC just before it is destroyed. Seems awful wasteful to allocate an extra bitmap per DC just for this, especially back in the days of 16-bit Windows when GDI heap space was extremely limited.

There is one more place (that comes to mind) where the stock bitmap appears, and it's somewhat unexpected:

> When you try to create a 0×*y* or a *x*×0 bitmap with the `CreateBitmap` or `CreateCompatibleBitmap` function you get the stock bitmap back.

In other words, if you ask for a nothing-bitmap, you get the dummy bitmap back. This is analogous to the case of calling `malloc(0)`, where the implementation is permitted to return a pointer to zero bytes. In other words, `malloc(0)` can return a non-`NULL` value which you can't dereference; the only things you can do with it is `free()` it or `realloc()` it to something bigger. In the same way that allowing zero-byte allocations simplifies boundary cases of certain algorithms, allowing impossibly thin bitmaps (and returning a dummy handle) may simplify certain graphical algorithms.

Note however that this behavior of returning the stock bitmap handle when asked to create an impossibly thin bitmap *does not apply to the* `CreateDIBSection` *function*! If you ask `CreateDIBSection` for an impossibly thin bitmap, it returns `NULL`. So much for consistency.

Raymond Chen

Follow