The disembodiment of DIBs from the DIB section

devblogs.microsoft.com/oldnewthing/20090717-00

July 17, 2009

Raymond Chen

So far this week, we've separated the DIB metadata (`BITMAPINFO`) from the pixels of a DIB section. But there's really no need for the DIB section at all! As long as you have the pixels and the metadata, you can draw bits.

We demonstrate this by drawing a rather stupid-looking bitmap onto the screen, but doing so without the use of `HBITMAP` s at all! Start with a brand new scratch program and make these changes:

```
const BYTE g_rgbPixels[16][8] = {
 { 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD },
 { 0xDD, 0xDD, 0xD0, 0x00, 0x00, 0xDD, 0xDD, 0xDD },
 { 0xDD, 0xD0, 0x01, 0x11, 0x11, 0x00, 0xDD, 0xDD },
 { 0xDD, 0x01, 0x11, 0x11, 0x11, 0x11, 0x0D, 0xDD },
 { 0xD0, 0x11, 0x11, 0x11, 0x11, 0x11, 0x10, 0xDD },
 { 0xD0, 0x11, 0x00, 0x00, 0x00, 0x01, 0x10, 0xDD },
 { 0x01, 0x11, 0x09, 0x99, 0x99, 0x01, 0x11, 0x0D },
 { 0x01, 0x11, 0x09, 0x99, 0x99, 0x01, 0x11, 0x0D },
 { 0x01, 0x11, 0x09, 0x99, 0x99, 0x01, 0x11, 0x0D },
 { 0x01, 0x11, 0x09, 0x99, 0x99, 0x01, 0x11, 0x0D },
 { 0x01, 0x11, 0x09, 0x99, 0x99, 0x01, 0x11, 0x0D },
 { 0xD0, 0x11, 0x00, 0x00, 0x00, 0x01, 0x10, 0xDD },
 { 0xD0, 0x11, 0x11, 0x11, 0x11, 0x11, 0x10, 0xDD },
 { 0xDD, 0x01, 0x11, 0x11, 0x11, 0x11, 0x0D, 0xDD },
 { 0xDD, 0xD0, 0x01, 0x11, 0x11, 0x00, 0xDD, 0xDD },
 { 0xDD, 0xDD, 0xD0, 0x00, 0x00, 0xDD, 0xDD, 0xDD },
};
struct BITMAPINFO16 {
 BITMAPINFOHEADER bmiHeader;
 RGBQUAD bmiColors[16];
} g_bmi = {
 { sizeof(g_bmi.bmiHeader), // biSize
                        16, // biWidth
                        16, // biHeight
                         1, // biPlanes
                         4, // biBitCount
                    BI_RGB, // biCompression
                         0, // biSizeImage
                         0, // biXPelsPerMeter
                         0, // biYPelsPerMeter
                        16, // biClrUsed
                        16, // biClrImportant
 },
 {
  { 0x00, 0x00, 0x00, 0x00 },// bmiColors[0]
  { 0x80, 0x00, 0x00, 0x00 },// bmiColors[1]
  { 0x00, 0x80, 0x00, 0x00 },// bmiColors[2]
  { 0x80, 0x80, 0x00, 0x00 },// bmiColors[3]
  { 0x00, 0x00, 0x80, 0x00 },// bmiColors[4]
  { 0x80, 0x00, 0x80, 0x00 },// bmiColors[5]
  { 0x00, 0x80, 0x80, 0x00 },// bmiColors[6]
  { 0xC0, 0xC0, 0xC0, 0x00 },// bmiColors[7]
  { 0x80, 0x80, 0x80, 0x00 },// bmiColors[8]
  { 0xFF, 0x00, 0x00, 0x00 },// bmiColors[9]
  { 0x00, 0xFF, 0x00, 0x00 },// bmiColors[10]
  { 0xFF, 0xFF, 0x00, 0x00 },// bmiColors[11]
  { 0x00, 0x00, 0xFF, 0x00 },// bmiColors[12]
  { 0xFF, 0x00, 0xFF, 0x00 },// bmiColors[13]
  { 0x00, 0xFF, 0xFF, 0x00 },// bmiColors[14]
  { 0xFF, 0xFF, 0xFF, 0x00 },// bmiColors[15]
 }
```

```
};
void
PaintContent(HWND hwnd, PAINTSTRUCT *pps)
{
 StretchDIBits(pps->hdc, 0, 0,
               g_bmi.bmiHeader.biWidth*4,
               g_bmi.bmiHeader.biHeight*4,
               0, 0,
               g_bmi.bmiHeader.biWidth,
               g_bmi.bmiHeader.biHeight,
               g_rgbPixels, (BITMAPINFO*)&g_bmi,
               DIB_RGB_COLORS, SRCCOPY);
}
```

We are drawing a bitmap without using an `HBITMAP`! The technique is the same one we've been using all week: Building a `BITMAPINFO` and using it to guide the interpretation of a chunk of memory containing pixels. Just to make things easier to see, I magnified the image, but the point is the same.

Okay, now let's look at the limitations and caveats of this technique. First, of course, is that this works only when the memory contains the source bitmap for a `BitBlt` or `StretchBlt` operation. It's nice when you have a bunch of bitmaps that you only intend to draw *from*, but you can't use this technique to draw *into* a chunk of memory. For that, you'll want to create a DIB section and draw into that.

Another problem with this technique is that it doesn't play friendly with remote desktops, one of those Windows programming taxes. After all, pumping a chunk of pixels to the screen is the logical equivalent of a `BitBlt` of a bitmap nobody has ever seen before. You've got pixels just coming out of nowhere. There's nothing to cache since there is no `HBITMAP` to associate the pixels with. On remote desktops, it's probably okay to use this technique for small bitmaps (like ours) where the cost of sending the pixels is insigificant, or if your render target is not the screen (for example, if you are rendering to an off-screen bitmap because you're doing some image manipulation behind the scenes).

Raymond Chen

Follow