Blitting between color and monochrome DCs

devblogs.microsoft.com/oldnewthing/20061114-01

November 14, 2006

Raymond Chen

When blitting between color and monochrome DCs, The text foreground and background colors play a role. We saw earlier that when blitting from a monochrome DC to a color DC, the color black in the source turns into the destination's text color, and the color white in the source turns into the destination's background color. This came in handy when we wanted to colorize a monochrome bitmap.

This trick works in reverse, too. If you blit from a color DC to a monochrome DC, then all pixels in the source that are equal to the background color will turn white, and all other pixels will turn black. In other words, GDI considers a monochrome bitmap to be black pixels on a white background.

This trick comes in handy when you want to convert a bitmap with color-keyed transparency into a color bitmap and a mask. Select the color bitmap into the DC `hdcColor`, and create a monochrome bitmap with the same dimensions and select it into the DC `hdcMask`. Then the following operations will construct the mask:

```
// let's say that the upper left pixel is the transparent color
COLORREF clrTransparent = GetPixel(hdcColor, 0, 0);
COLORREF clrBkPrev = SetBkColor(hdcColor, clrTransparent);
BitBlt(hdcMono, 0, 0, cx, cy, hdcColor, 0, 0, SRCCOPY);
SetBkColor(hdcColor, clrBkPrev);
```

We can see this in action in our scratch program by making the following changes:

```
void
PaintContent(HWND hwnd, PAINTSTRUCT *pps)
{

  HBITMAP hbmMono = CreateBitmap(100, 100, 1, 1, NULL);
  HDC hdcMono = CreateCompatibleDC(pps->hdc);
  HBITMAP hbmPrev = SelectBitmap(hdcMono, hbmMono);
  HDC hdcScreen = GetDC(NULL);
  SetBkColor(hdcScreen, GetSysColor(COLOR_DESKTOP));
  BitBlt(hdcMono, 0, 0, 100, 100, hdcScreen, 0, 0, SRCCOPY);
  SetTextColor(pps->hdc, RGB(0xFF,0,0));
  SetBkColor(pps->hdc, RGB(0,0x80,0));
  BitBlt(pps->hdc, 0, 0, 100, 100, hdcMono, 0, 0, SRCCOPY);
  ReleaseDC(NULL, hdcScreen);
  SelectBitmap(hdcMono, hbmPrev);
  DeleteDC(hdcMono);
  DeleteObject(hbmMono);
}
```

We start by creating a 100 × 100 monochrome bitmap and selecting it into a memory DC. This will become our mask. Next, we take a screen DC and set its background color to the desktop color and blit from the screen to the monochrome bitmap. This creates a bitmap which is white where the screen has the desktop color and black where the screen has some other color. We show off we show off this new bitmap by painting it into our client area, but just for fun, I made the foreground pixels (black pixels in the monochrome bitmap) bright red and the background pixels (white pixels in the monochrome bitmap) dark green.

Minimize your windows so the upper left corner of the desktop is visible, and turn off your wallpaper (so the desktop color actually means something). Run this program and observe a copy of your desktop drawn in the window's client area, but with your desktop color turned to green and all the other pixels turned to red.

The rest of the job of drawing a color bitmap with transparency is now comparatively straightforward. I'll leave it as an exercise. Hint: Raster operation 0x00220326 (DSna) will probably be handy.

Next time, we'll look at DIB sections as a way of doing fast color manipulation, thereby avoiding the need to perform the DSna ROP entirely.

Raymond Chen

Follow