What is the DC brush good for?

devblogs.microsoft.com/oldnewthing/20050420-28

April 20, 2005

Raymond Chen

The DC brush `GetStockObject(DC_BRUSH)` is a stock brush associated with the device context. Like the system color brushes, the color of the DC brush changes dynamically, but whereas the system color brushes change color based on the system colors, the color of the DC brush changes at your command.

The DC brush is handy when you need a solid color brush for a very short time, since it always exists and doesn't need to be created or destroyed. Normally, you have to create a solid color brush, draw with it, then destroy it. With the DC brush, you set its color and start drawing. But it works only for a short time, because the moment somebody else calls the `SetDCBrushColor` function on your DC, the DC brush color will be overwritten. In practice, this means that the DC brush color is not trustworthy once you relinquish control to other code. (Note, however, that each DC has its own DC brush color, so you need only worry about somebody on another thread messing with your DC simultaneously, which doesn't happen under any of the painting models I am familiar with.)

The DC brush is amazingly useful when handling the various WM_CTLCOLOR messages. These messages require you to return a brush that will be used to draw the control background. If you need a solid color brush, this usually means creating the solid color brush and caching it for the lifetime of the window, then destroying it when the window is destroyed. (Some people cache the brush in a static variable, which works great until somebody creates two copies of the dialog/window. Then you get a big mess.)

Let's use the DC brush to customize the colors of a static control. The program is not interesting as a program; it's just an illustration of one way you can use the DC brush.

Start, as always, with our scratch program, and making the following changes.

```
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
  g_hwndChild = CreateWindow(TEXT("static"), NULL,
        WS_VISIBLE | WS_CHILD, 0, 0, 0, 0,
        hwnd, NULL, g_hinst, 0);
 if (!g_hwndChild) return FALSE;
 return TRUE;
}
HBRUSH OnCtlColor(HWND hwnd, HDC hdc, HWND hwndChild, int type)
{
  FORWARD_WM_CTLCOLORSTATIC(hwnd, hdc, hwndChild, DefWindowProc);
  SetDCBrushColor(hdc, RGB(255,0,0));
  return GetStockBrush(DC_BRUSH);
}
    HANDLE_MSG(hwnd, WM_CTLCOLORSTATIC, OnCtlColor);
```

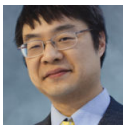
Run this program and observe that we changed the background color of the static window to red.

The work happens inside the `OnCtlColor` function. When asked to customize the colors, we first forward the message to the `DefWindowProc` function so that the default foreground and background text colors are set. (Not relevant here since we draw no text, but a good thing to do on principle.) Since we want to override the background brush color, we set the DC brush color to red and then return the DC brush as our desired background brush.

The static control then takes the brush we returned (the DC brush) and uses it to draw the background, which draws in red because that's the color we set it to.

Normally, when customizing the background brush, we have to create a brush, return it from the WM_CTLCOLORSTATIC message, then destroy it when the parent window is destroyed. But by using the DC brush, we avoided having to do all that bookkeeping.

There is also a DC pen `GetStockObject(DC_PEN)` which behaves in an entirely analogous manner.



Raymond Chen

Follow