Re: Button with Bitmap Disabled problem

Source:
http://www.tech−archive.net/Archive/Development/microsoft.public.win32.programmer.ui/2004−09/0341.html

---

From: Christian ASTOR (*castorix_at_club−internet.fr*)
Date: 09/25/04

Date: Sat, 25 Sep 2004 17:50:57 +0200

Remo wrote:

> You give me three workarounds:
> 1)Sublass
> 2)Toolbar C. C. (what means C.C?)
> 3)OwnerDrawn
> Do you have an example or HELP for 1 & 2
> OwnerDrawn I found in MSDN doku

I meant for Toolbar C. C. , the Toolbar Common Control.
For subclassing,
you can do something like that for disabled effect :

```
// To test :
// EnableWindow(hButton, !IsWindowEnabled(hButton));
// InvalidateRect(hButton, NULL, TRUE);

LRESULT CALLBACK ButtonProc (HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM
lParam);
LRESULT ButtonProcOld;

//... case WM_CREATE:
            { ...
ButtonProcOld = SetWindowLong(hButton,GWL_WNDPROC,(LONG) (WNDPROC)
ButtonProc);
//...

LRESULT CALLBACK ButtonProc (HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM
lParam)
{
      HBITMAP hBitmap2;
      switch (iMsg)
      {
            case WM_PAINT:
            {
                  if (IsWindowEnabled(hWnd))
                  {
```

```
                CallWindowProc((WNDPROC)ButtonProcOld, hWnd, iMsg, wParam, lParam);
            }
            else
            {
                PAINTSTRUCT ps;
                BeginPaint(hWnd, &ps );
                RECT rect;
                GetClientRect(hWnd, &rect);
               hBitmap2 = (HBITMAP)LoadImage(NULL, "YourBitmap.bmp",
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
                DrawEdge(ps.hdc, &rect, EDGE_RAISED, BF_ADJUST | BF_RECT);
                DitherBlt(ps.hdc, 2, 2, rect.right−2, rect.bottom−2, NULL, hBitmap2,
0, 0, GetSysColorBrush(COLOR_3DFACE),
GetSysColorBrush(COLOR_3DHILIGHT),GetSysColorBrush(COLOR_3DSHADOW), 1);
                DeleteObject(hBitmap2);
                EndPaint(hWnd, &ps );
                return 0;
            }
        }
        break;
    }
    return(CallWindowProc((WNDPROC)ButtonProcOld, hWnd, iMsg,
wParam, lParam));
}

// For DitherBlt(), quickly adapted and modified a bit from WTL =>

BOOL DitherBlt(HDC hDestDC, int x, int y, int nWidth, int nHeight, HDC
hSrcDC, HBITMAP hBitmap, int xSrc, int ySrc,
            HBRUSH hBrushBackground,HBRUSH hBrush3DEffect,HBRUSH
hBrushDisabledImage, bool bCenter )
    {

        HDC hDC = GetDC(NULL);

        // Create a generic DC for all BitBlts
        HDC hGenDC = (hSrcDC != NULL) ? hSrcDC : CreateCompatibleDC(hDC);

        if(hGenDC == NULL)
            return FALSE;

        // Create a DC for the monochrome DIB section
        HDC hMemDC = CreateCompatibleDC(hDC);
        if(hMemDC == NULL)
        {
            if(hSrcDC == NULL)
            {
                DeleteDC(hGenDC);
                ReleaseDC(NULL, hDC);
            }
            return FALSE;
```

```
        }

        // Create the monochrome DIB section with a black and white palette
        struct RGBBWBITMAPINFO
        {
            BITMAPINFOHEADER bmiHeader;
            RGBQUAD bmiColors[2];
        };

        RGBBWBITMAPINFO rgbBWBitmapInfo =
        {
            { sizeof(BITMAPINFOHEADER), nWidth, nHeight, 1, 1, BI_RGB, 0, 0, 0,
0, 0 },
            { { 0x00, 0x00, 0x00, 0x00 }, { 0xFF, 0xFF, 0xFF, 0x00 } }
        };

        VOID* pbitsBW;
        HBITMAP hMemBitmap = CreateDIBSection(hMemDC,
(LPBITMAPINFO)&rgbBWBitmapInfo, DIB_RGB_COLORS, &pbitsBW, NULL, 0);
        if(hMemBitmap == NULL)
        {
            if(hSrcDC == NULL)
            {
                DeleteDC(hGenDC);
                ReleaseDC(NULL, hDC);
            }
            return FALSE;
        }

        // Attach the monochrome DIB section and the bitmap to the DCs
        HBITMAP hMemBitmapOld = (HBITMAP)SelectObject(hMemDC, hMemBitmap);
        HBITMAP hBitmapOld = NULL;
        if(hBitmap != NULL)
            hBitmapOld = (HBITMAP)SelectObject(hGenDC, hBitmap);

        // Block: Dark gray removal: we want (128, 128, 128) pixels to become
black and not white
        {
            HDC hTempDC1 = CreateCompatibleDC(hDC);
            HDC hTempDC2 = CreateCompatibleDC(hDC);
            HBITMAP hTempBitmap1;
            hTempBitmap1 = CreateCompatibleBitmap(hGenDC, nWidth, nHeight);
            HBITMAP hTempBitmap2;
            hTempBitmap2 = CreateBitmap(nWidth, nHeight, 1, 1, NULL);
            HBITMAP hBitmapOld1 = (HBITMAP)SelectObject(hTempDC1, hTempBitmap1);
            HBITMAP hBitmapOld2 = (HBITMAP)SelectObject(hTempDC2, hTempBitmap2);
            // Let's copy our image, it will be altered
            BitBlt(hTempDC1, 0, 0, nWidth, nHeight, hGenDC, xSrc, ySrc, SRCCOPY);

            // All dark gray pixels will become white, the others black
            SetBkColor(hTempDC1, RGB(128, 128, 128));
```

```
            BitBlt(hTempDC2, 0, 0, nWidth, nHeight, hTempDC1, 0, 0, SRCCOPY);
            // Do an XOR to set to black these white pixels
            BitBlt(hTempDC1, 0, 0, nWidth, nHeight, hTempDC2, 0, 0, SRCINVERT);

            // BitBlt the bitmap into the monochrome DIB section
            // The DIB section will do a true monochrome conversion
            // The magenta background being closer to white will become white
            BitBlt(hMemDC, 0, 0, nWidth, nHeight, hTempDC1, 0, 0, SRCCOPY);

            // Cleanup
            SelectObject(hTempDC1, hBitmapOld1);
            SelectObject(hTempDC2, hBitmapOld2);
        }

        // Paint the destination rectangle using hBrushBackground
        if(hBrushBackground != NULL)
        {
            RECT rc = { x, y, x + nWidth, y + nHeight };
            FillRect(hDestDC, &rc, hBrushBackground);
        }

        // BitBlt the black bits in the monochrome bitmap into hBrush3DEffect
color in the destination DC
        // The magic ROP comes from the Charles Petzold's book
        BITMAP bm;
        GetObject(hBitmap, sizeof(BITMAP), &bm);
        if (bCenter)
        {
            x += (nWidth−bm.bmWidth)/2;
            y += (nHeight−bm.bmHeight)/2;
        }
        HBRUSH hOldBrush = (HBRUSH)SelectObject(hDestDC, hBrush3DEffect);
        //BitBlt(hDestDC, x + 1, y + 1, nWidth, nHeight, hMemDC, 0, 0, 0xB8074A);
        BitBlt(hDestDC, x + 1, y + 1, bm.bmWidth, bm.bmHeight, hMemDC, 0, 0,
0xB8074A);

        // BitBlt the black bits in the monochrome bitmap into
hBrushDisabledImage color in the destination DC
        SelectObject(hDestDC, hBrushDisabledImage);
        //BitBlt(hDestDC, x, y, nWidth, nHeight, hMemDC, 0, 0, 0xB8074A);
        BitBlt(hDestDC, x, y, bm.bmWidth, bm.bmHeight, hMemDC, 0, 0, 0xB8074A);

        SelectObject(hDestDC, hOldBrush);
        SelectObject(hMemDC, hMemBitmapOld);
        SelectObject(hGenDC, hBitmapOld);

        if(hSrcDC == NULL)
            DeleteDC(hGenDC);

        ReleaseDC(NULL, hDC);
```

```
    return TRUE;
}
```