Transferring Images with TransparentBlt

Janet Terra

A Transparent Background for Static Graphics

Do you find yourself creating sprites, not for animation, but to draw static images with a transparent background? If you do, then consider using the API TransparentBlt function in MSIMG32.DLL. The MSIMG32.DLL is included with all versions of Window 98 and later. Sorry, Windows 95 Users, this doesn't include you, although you may be able to download the DLL. The TransparentBlt coding is very similar to the BitBlt and StretchBlt functions of GDI32.DLL. If you are unfamiliar with transferring bits using BitBlt, please read Bitmaps the Fast Way (Newsletter 122) by Callum Lowcay. This is an excellent article demonstrating how BitBlt works.

Finding, Opening and Closing the DLL

There is no need to include a separate MSIMG32.DLL file with programs calling this DLL if your operating system is Windows 98 or later. You will need to OPEN the DLL. Common Windows operating system DLL's do not need to be OPENed before calling. These common DLL's include USER32.DLL, KERNEL32.DLL, GDI32.DLL, WINMM.DLL, SHELL32.DLL, COMDLG32.DLL, COMCTL32.DLL. MSIMG32.DLL DOES REQUIRE OPENing before CALLing.

```
open msimg32.dll for dll as #m
```

As with all OPENed DLL's, the DLL must be closed before exiting the program.

```
close #m
```

BitBlt, StretchBlt, TransparentBlt: A Quick Comparison

Capabilities	BitBlt	TransparentBlt	StretchBlt
Simple bitcopy	Yes	Yes	Yes
Bitcopy from memory device	Yes	Yes	Yes
Resize	No	Yes	Yes
Raster Operations	Yes	No	Yes
Flip	No	No	Yes

Mirror	No	No	Yes
Rotate 180 degrees	No	No	Yes
Assign Color Transparent	No	Yes	No

The Demo Bitmaps

These two demos generate a simple image. You may choose to load an image from file with the loadbmp command. The demos are based upon the white background of the drawn image. If you choose to substitute with an image having a different background color, change the long number value of white (16777215) to the long number value of the desired color.

Demo 1: Drawbmp vs TransparentBlt

This first demo illustrates the difference between drawing an image with the drawbmp command and drawing the same image with the TransparentBlt API function.

```
'Demo Drawing Bitmaps with Transparent Backgrounds
'Using msimg32.dll and TransparentBlt
'Copyright Janet Terra December 2004, 2011
'This information is free for anyone to use in your program
'but this demo itself may not be copied 'as is' and reposted

'Users of PRE WINDOWS 98, please note:
'The msimg32.dll may not be available for Windows 95
'If running this program gives you - Runtime Error: The
'specified module could not be found. (OS error 16r7E)
'then msimg32.dll is not present on your computer.

'In addition, memory leaks have been associated with
'repeated TransparentBlt calls and Windows 95/98
'Visit http://support.microsoft.com for more info

'Load a bitmap.  This demo creates a filled circled image
'named pic

    open "Quick Pic" for graphics as #1
    #1 "down; color red; backcolor yellow"
    #1 "place 50 30; circlefilled 25"
    #1 "getbmp pic 10 0 88 64"
```

```
    close #1

    nomainwin
    WindowWidth = 420
    WindowHeight = 420

'Define 6 Graphicboxes: 1,2,3 = Drawbmp, 4,5,6 = TransparentBlt
    graphicbox #w.g1, 10, 10, 100, 100
    statictext #w.stxt1a, "Drawbmp pic 0, 0", 8, 120, 110, 16
    statictext #w.stxt1b, "white background", 8, 136, 110, 16
    graphicbox #w.g2, 140, 10, 100, 100
    statictext #w.stxt2a, "Drawbmp pic 0, 0", 138, 120, 110, 16
    statictext #w.stxt2b, "blue background", 138, 136, 110, 16
    graphicbox #w.g3, 270, 10, 100, 100
    statictext #w.stxt3a, "Drawbmp pic 0, 0", 268, 120, 110, 16
    statictext #w.stxt3b, "patterned background", 268, 136, 110, 16
    graphicbox #w.g4, 10, 190, 100, 100
    statictext #w.stxt4a, "Using TransparentBlt", 8, 300, 110, 16
    statictext #w.stxt4b, "white background", 8, 316, 110, 16
    statictext #w.stxt4c, "same width and height", 8, 332, 110, 16
    graphicbox #w.g5, 140, 190, 100, 100
    statictext #w.stxt5a, "Using TransparentBlt", 138, 300, 110, 16
    statictext #w.stxt5b, "blue background", 138, 316, 110, 16
    statictext #w.stxt5c,
"smaller width and height (shrink)", 138, 332, 110, 32
    graphicbox #w.g6, 270, 190, 100, 100
    statictext #w.stxt6a, "Using TransparentBlt", 268, 300, 110, 16
    statictext #w.stxt6b, "patterned background", 268, 316, 110, 16
    statictext #w.stxt5c,
"larger width and height (stretch)", 268, 332, 110, 32


'Open the Window
    open "Demo #1" for window as #w
    #w "trapclose [endProgram]"

'Obtain the handles of the primary window, the source window or graphi
cbox,
'and the destination windows or graphicboxes.  In this demo, the first
 graphicbox
'(the bitmap drawn against a white background) also serves as the sour
ce bitmap)

    hW = hWnd(#w) 'Window handle
    hWG1 = hWnd(#w.g1) 'Source Handle
    hWG4 = hWnd(#w.g4) 'Destination Handle (#w.g4)
```

```
    hWG5 = hWnd(#w.g5) 'Destination Handle (#w.g5)
    hWG6 = hWnd(#w.g6) 'Destination Handle (#w.g6)
    hDC1 = GetDC(hWG1) 'Source Device Context
    hDC4 = GetDC(hWG4) 'Destination Device Context (#w.g4)
    hDC5 = GetDC(hWG5) 'Destination Device Context (#w.g5)
    hDC6 = GetDC(hWG6) 'Destination Device Context (#w.g6)

'Open the msimg32.dll
    Open "msimg32.dll" for DLL as #m

'Display White, Blue and Patterned Backgrounds
    call hueBackground "white", "#w.g1"
    call hueBackground "darkblue", "#w.g2"
    call patternedBackground "#w.g3"
    call hueBackground "white", "#w.g4"
    call hueBackground "darkblue", "#w.g5"
    call patternedBackground "#w.g6"

'Draw the bitmap in the first graphicbox.  Remember this also serves a
s the
'source bitmap.
    #w.g1 "drawbmp pic 0, 0"

'Draw the bitmap in the second graphicbox.  The white rectangular back
ground
'of the pic remains present
    #w.g2 "drawbmp pic 0, 0"

'Draw the bitmap in the third graphicbox.  Again, the white rectangula
r
'background of the bitmap obscures the patterned background
    #w.g3 "drawbmp pic 0, 0"

'Assign ONE color to be the transparent color.  Remember you must use
the
'long number.  In this case, the RGB of White is "255 255 255", so the
'long number is 255 + 255*256 + 255*256*256 or 16777215
    transpColor = 16777215

'Function variables are (DestinationDC, DestinationUpperLeftX, Destina
tionUpperLeftY,
'DestinationWidth, DestinationHeight, SourceDC, SourceUpperLeftX,
'SourceUpperLeftY, SourceWidth, SourceHeight, TransparentColor)

'Using the SAME Width and Height as the Source
    null = TransparentBlt(hDC4, 0, 0, 88, 64, hDC1, 0, 0, 88, 64,
```

```
16777215)

'Using a SMALLER Width and Height as the Source
    null = TransparentBlt(hDC5, 0, 0, 44, 32, hDC1, 0, 0, 88, 64,
16777215)

'Using a LARGER Width and Height as the Source (Destination x and y ar
e also changed)
    null = TransparentBlt(hDC6, 0, 0, 104, 80, hDC1, 0, 0, 88, 64,
16777215)

Wait

[endProgram]
'Release memory from any device contexts created during program
    null = ReleaseDC(hWG1, hDC1)
    null = ReleaseDC(hWG4, hDC4)
    null = ReleaseDC(hWG5, hDC5)
    null = ReleaseDC(hWG6, hDC6)
'Close msimg32.dll
    close #m
'Unload bitmap
    unloadbmp "pic"
'Close window
    close #w
'End program
end

sub hueBackground hue$, handle$
    #handle$ "down; fill ";hue$
    #handle$ "flush; discard"
end sub

sub patternedBackground handle$
    colorStrand$ = "yellow blue darkgreen darkblue darkpink"
    colorCode = 0
    #handle$, "down"
    for ii = 1 to 100
        colorCode = colorCode + 1
        if colorCode = 6 then
            colorCode = 1
        end if
        colorCode$ = word$(colorStrand$, colorCode)
        #handle$ "color ";colorCode$
        #handle$ "line ";ii;" 0 ";ii;" 100"
    next ii
```

```
    #handle$ "flush; discard"
end sub

'====================================================================
'These 3 Functions obtained from information found in API's for
'Liberty BASIC (c) Alyce Watson and used with permission of Alyce Wats
on
'====================================================================

function TransparentBlt(hdcDest, xDest, yDest, wDest, hDest,
 hdcSource, xSource, ySource, wSource, hSource, clrTransp)
    calldll #m, "TransparentBlt", _
        hdcDest as ulong, _        'handle to destination DC
        xDest as long, _   'x-coord of destination upper-left corner
        yDest as long, _   'y-coord of destination upper-left corner
        wDest as long, _     'width of destination rectangle
        hDest as long, _    'height of destination rectangle
        hdcSource as ulong, _         'handle to source DC
        xSource as long, _    'x-coord of source upper-left corner
        ySource as long, _    'y-coord of source upper-left corner
        wSource as long, _      'width of source rectangle
        hSource as long, _     'height of source rectangle
        clrTransp as ulong, _ 'color to make transparent
        TransparentBlt as long
end function

function GetDC(handle)
    calldll #user32, "GetDC", _
        handle as ulong, _
        GetDC as ulong
end Function

function ReleaseDC(handle, hdc)
    calldll #user32, "ReleaseDC", _
        handle as ulong, _
        hdc as ulong, _
        ReleaseDC as ulong
end Function
'====================================================================
```

You may choose to select any image. If you do, it is important to note that TransparentBLT will fail unless the width and height values of the bitmap in memory correspond with the file width and height values of that bitmap.

In this line

```
null = TransparentBlt(hDC2, 0, 0, 44, 32, hDCM, 0, 0, 88, 64,
16777215)
```

If your bitmap is 48 pixels wide by 60 pixels long and you want to shrink it to 24 pixels wide by 30 pixels long, then change the above line to

```
null = TransparentBlt(hDC2, 0, 0, 24, 30, hDCM, 0, 0, 48, 60,
16777215)
```

TransparentBlt is best used for transferring images using a transparent background. Although TransparentBlt is documented to resize images, at least one user with Windows 7 has found the resizing function to fail when transferring from one graphicbox to another. The function did succeed when the image was transferred from a memory dc to a graphicbox. It is best to transfer the image from a memory device context as described in the next demo.

Demo 2: BitBlt, TransparentBlt and Drawing in Memory

If you haven't done so already, please read Drawing IN MEMORY (Newsletter 101) by Alyce Watson . There you'll find all the theory you need to understand that you can transfer an image drawn in memory but you can't transfer an image drawn in a hidden graphic box. This second demo demonstrates how to load a bitmap and then transferring that image to a graphicbox using both BitBlt and TransparentBlt.

```
'Demo Transferring an Image from a Bitmap Loaded in Memory
'Using  gdi32.dll: BitBlt and msimg32.dll: TransparentBlt
'Copyright Janet Terra December 2004, 2011
'This information is free for anyone to use in your program
'but this demo itself may not be copied 'as is' and reposted

'Users of PRE WINDOWS 98, please note:
'The msimg32.dll may not be available for Windows 95
'If running this program gives you - Runtime Error: The
'specified module could not be found. (OS error 16r7E)
'then msimg32.dll is not present on your computer.


    nomainwin

'Load a bitmap.  This demo gets a bitmap 88 x 64 so no external
'files are needed

    WindowWidth = 280
```

```
    WindowHeight = 400

    open "Quick Pic" for graphics as #1
    #1 "down; color red; backcolor yellow"
    #1 "place 50 30; circlefilled 25"
    #1 "getbmp pic 10 0 88 64"
    close #1


'Define 2 Graphicboxes: 1 = BitBlt, 2 = TransparentBlt
    graphicbox #w.g1, 10, 10, 100, 100
    statictext #w.st1a,
"Image transferred from memory using", 8, 120, 110, 32
    statictext #w.st1b, "gdi32.dll", 8, 152, 110, 16
    statictext #w.st1c, "BitBlt", 8, 168, 110, 16
    graphicbox #w.g2, 140, 10, 100, 100
    statictext #w.st2a,
"Image transferred from memory using", 138, 120, 110, 32
    statictext #w.st2b, "msimg32.dll", 138, 152, 110, 16
    statictext #w.st2c, "TransparentBlt", 138, 168, 110, 16


'Open the Window
    open "Demo" for window as #w
    #w "trapclose [endProgram]"

'Obtain the handles of the primary window, the source window or graphi
cbox,
'and the destination windows or graphicboxes.  In this demo, #w.g1 rec
eives the
'image using BitBlt and #w.g2 receives the image using TransparentBlt

    hW = hWnd(#w) 'Window handle
    hWG1 = hWnd(#w.g1) 'Source Handle (#w.g1)
    hWG2 = hWnd(#w.g2) 'Destination Handle (#w.g2)
    hDC = GetDC(hW)
    hDC1 = GetDC(hWG1) 'Source Device Context (#w.g1)
    hDC2 = GetDC(hWG2) 'Destination Device Context (#w.g2)

'Open the msimg32.dll
    open "msimg32.dll" for dll as #m

'Create Device Context in Memory
    hDCM = CreateCompatibleDC(0)

'Obtain Handle of the Loaded BMP ("pic")
```

```
    hPic = hBmp("pic")

'Display Solid Backgrounds
    call hueBackground "Darkblue", "#w.g1"
    call hueBackground "Darkblue", "#w.g2"

'Select the Object in Memory
    memPic = SelectObject(hDCM, hPic)

'Draw the  Object in Memory to GraphicBox #w.g1 using BitBlt
    null =BitBlt(hDC1, 0, 0, 88, 64, hDCM, 0, 0, _SRCCOPY)

'Draw the Object in Memory to GraphicBox #w.g2 using TransparentBlt
    null = TransparentBlt(hDC2, 0, 0, 88, 64, hDCM, 0, 0, 88, 64,
16777215)

wait

[endProgram]
'Release memory from any device contexts created during program
    null = ReleaseDC(hWG1, hDC1)
    null = ReleaseDC(hWG2, hDC2)
    calldll #gdi32, "DeleteDC", _
        hDCM as ulong, _
        null as long

    calldll #gdi32, "DeleteObject", _
        memPic as ulong, _
        null as long

'Close msimg32.dll
    close #m
'Unload bitmap
    unloadbmp "pic"
'Close window
    close #w
'End program
end

sub hueBackground hue$, handle$
    #handle$ "down; fill ";hue$
    #handle$ "flush; discard"
end sub

'====================================================================
'The following functions are derived from information obtained from
```

```
'API's for Liberty BASIC (c) Alyce Watson and are used with permission
'of Alyce Watson
'=====================================================================

function CreateCompatibleDC(hDC)
    calldll #gdi32, "CreateCompatibleDC", _
        hDC as ulong, _ 'current screen
        CreateCompatibleDC as ulong 'handle of memory DC
end function

function GetDC(handle)
    calldll #user32, "GetDC", _
        handle as ulong, _
        GetDC as ulong
end function

function ReleaseDC(handle, hdc)
    calldll #user32, "ReleaseDC", _
        handle as ulong, _
        hdc as ulong, _
        ReleaseDC as ulong
end function

function SelectObject(hDC,hImage)
    calldll #gdi32, "SelectObject", _
        hDC as ulong, _ 'Memory Device Context
        hImage as ulong, _ 'Handle of Loaded Bitmap
        SelectObject as ulong 'Returns handle of previous object
end function

function BitBlt(hdcDest, xDest, yDest, wDest, hDest,
 hdcSource, xSource, ySource, ROP)
        CallDLL #gdi32, "BitBlt", _
        hdcDest as ulong, _        'handle to destination DC
        xDest as long, _   'x-coord of destination upper-left corner
        yDest as long, _   'y-coord of destination upper-left corner
        wDest as long, _     'width of destination rectangle
        hDest as long,_    'height of destination rectangle
        hdcSource as ulong, _         'handle to source DC
        xSource as long, _    'x-coord of source upper-left corner
        ySource as long, _    'y-coord of source upper-left corner
        ROP as ulong,_      'Raster Operation
        result as long
end function

function TransparentBlt(hdcDest, xDest, yDest, wDest, hDest,
```

```
 hdcSource, xSource, ySource, wSource, hSource, clrTransp)
    calldll #m, "TransparentBlt", _
        hdcDest as ulong, _        'handle to destination DC
        xDest as long, _   'x-coord of destination upper-left corner
        yDest as long, _   'y-coord of destination upper-left corner
        wDest as long, _     'width of destination rectangle
        hDest as long, _    'height of destination rectangle
        hdcSource as ulong, _         'handle to source DC
        xSource as long, _    'x-coord of source upper-left corner
        ySource as long, _    'y-coord of source upper-left corner
        wSource as long, _      'width of source rectangle
        hSource as long, _     'height of source rectangle
        clrTransp as ulong, _ 'color to make transparent
        TransparentBlt as long
end function
'====================================================================
```

Transferring Images with TransparentBlt | A Transparent Background for Static Graphics | The Demo Bitmaps | Demo 1: Drawbmp vs TransparentBlt | Demo 2: BitBlt, TransparentBlt and Drawing in Memory