Fun With Transparent Windows

- **#User32 - "SetLayeredWindowAttributes"**

Janet Terra with inspiration from Alyce Watson and Stefan Pendl

Windows 2000 and Windows XP both support transparent and semi-transparent windows. If you are designing a program for others, you will want to be sure the host computer supports transparent windows. This function, available at Alyce's Restaurant, will retrieve the OS version. A returned MajorVer of 5, setting the OSVersionFlag value to 1, indicates the required Windows 2000 or XP.

```
    If OSVersionFlag() Then
        Print "This computer DOES support Transparent Windows"
    Else
        Print "This computer DOES NOT support Transparent Windows"
    End If
    End

Function OSVersionFlag()
    struct OSVERSIONINFO, _
        dwOSVersionInfoSize as Long, _
        dwMajorVersion as Long, _
        dwMinorVersion as Long, _
        dwBuildNumber as Long, _
        dwPlatformId as Long, _
        szCSDVersion as Char[128]
    L = Len(OSVERSIONINFO.struct)
    OSVERSIONINFO.dwOSVersionInfoSize.struct = L
    CallDLL #kernel32, "GetVersionExA", _
        OSVERSIONINFO as struct, _
        result as long
    MajorVer = OSVERSIONINFO.dwMajorVersion.struct
    If MajorVer < 5 Then
        OSVersionFlag = 0
    Else
        OSVersionFlag = 1
    End If
End Function
```

Once the programmer has established the host machine will support a transparent window, the window can be opened. Prior to Liberty BASIC v4.0 and Stylebits, the only way to code a transparent window was after opening the window and then issuing a SetWindowLongA to the User32.dll to give that window a layered quality. After the layered message was sent, a call to SetLayeredWindowAttributes was issued, again to the

User32.dll to define the amount of transparency, more correctly called opacity. This is the technique used in John Richardson's LB Spier and discussed in the Transparent Window article at the Liberty BASIC Programming Wiki Site.

A more recent, and perhaps more effective, method of coding a transparent window is by including `WS_EX_LAYERED` in the window's addExtendedBits, the third parameter of the Stylebits command. As in the previous method, a call is then made to SetLayeredWindowAttributes in the User32.dll to define the amount of opacity.

The WS_EX_LAYERED addExtendedBits Stylebit

`WS_EX_LAYERED` is one of the Windows constants that is not recognized by Liberty BASIC. This doesn't mean that the style can't be used, though. It just means the numerical value of that Windows constant will need to be determined and assigned as either a literal or a variable. *See* Stylebits and Windows Constants *for further discussion if needed.* Here, we'll assign the correct value of `524288` to the variable `WS.EX.LAYERED`.

```
    WS.EX.LAYERED = 524288
    Stylebits #w, 0, 0, WS.EX.LAYERED, 0
```

Having given the layered quality to the window, the opacity is then determined after the window is opened. This is done with a call to "SetLayeredWindowAttributes."

```
    CallDLL #user32, "SetLayeredWindowAttributes", _
        hMain as Ulong, _
        ColorRef as Ulong, _
        bAlpha as Short, _
        dwFLAGS as Long, _
        result as long
```

Looking at the four passed parameters more closely -

1. **hMain** is the handle of the main window. Typically, this is the first parameter to be passed to any DLL when that DLL modifies a window or control.
2. **ColorRef** is the color assigned to be transparent. ColorRef is only relevant when the fourth parameter, dwFLAGS, is assigned as LWA_COLORKEY.
3. **bAlpha** is the amount of transparency or opacity. bAlpha is sometimes referred to as BlendedAlpha. When bAlpha is set at 0, the window is completely transparent. When bAlpha is set at 255, the window is completely opaque, the normal or default state. bAlpha is only relevant when

the fourth parameter, dwFLAGS, is assigned as LWA_ALPHA.

4. **dwFLAGS** is the desired event to be passed to the DLL. There are actually two layered window actions that can be passed in the "SetLayeredWindowAttributes" call. The first is LWA_COLORKEY and the second is LWA_ALPHA.
   LWA_COLORKEY assigns one color of the window to be transparent. If the value of dwFLAGS is set to LWA_COLORKEY, then that one assigned color is the color passed in ColorRef and bAlpha is ignored. If the value of dwFLAGS is set to LWA_ALPHA, then the degree of opacity of the entire window is set to the value passed in bAlpha and ColorRef is ignored.

Demo of LWA_ALPHA

The Transparent Window demo works with the action `LWA_ALPHA`. The numerical value of the Windows constant `LWA_ALPHA` is `2`. Since `LWA_ALPHA` is not a Liberty BASIC recognized constant, then that value (`2`) will be assigned to the user defined variable `LWA.ALPHA`.

```
    LWA.ALPHA = 2
```

Here is a simple modification of Transparent Window using the previously described constants and variables. ***This demo requires Windows 2000, Windows XP, or better.***

```
' Set the Variables
    WS.EX.LAYERED = 524288
' WS_EX_LAYERED Unrecognized Windows constant
    ColorRef = 0
' ColorRef ignored in this demo so value is irrelevant
    bAlpha = 128 ' Degree of opacity (0 - 255)
    LWA.ALPHA = 2 ' LWA_ALPHA Unrecognized Windows constant
    dwFLAGS = LWA.ALPHA
' Action is transparency of entire window (LWA_ALPHA)
    Nomainwin

    WindowWidth = 600
    WindowHeight = 400

    UpperLeftX = int((DisplayWidth-WindowWidth)/2)
    UpperLeftY = int((DisplayHeight-WindowHeight)/2)

    text$ = "Translucence here is initially set at 128.  Try " + _
        "setting bAlpha to higher and lower numbers for desired " + _
        "transparency.  0 will make your window complete " + _
        "transparent.  255 is the maximum.  This demo will " + _
        "NOT work on Windows 98 or Windows 95."

    Statictext #main.msg, text$, 20, 20, 550, 200
    Stylebits #main, 0, 0, WS.EX.LAYERED, 0
    Open "Transparent Window" for Window as #main

    #main "Trapclose EndDemo"
    hMain = hWnd(#main)
    #main "Font Verdana 12 Bold"
    Call SetLayeredWindowAttributes, hMain, ColorRef, bAlpha, dwFLAGS

    Wait

Sub EndDemo handle$
    Close #main
    End
End Sub

Sub SetLayeredWindowAttributes hWindow, ColorRef, bAlpha, dwFLAGS
    CallDLL #user32, "SetLayeredWindowAttributes", _
        hWindow as Ulong, _
        ColorRef as Long, _
        bAlpha as Long, _
        dwFLAGS as Long, _
        result as Long
End Sub
```

Stefan Pendl's Demo Using LWA_COLORKEY

The inspiration for this article came from Stefan Pendl- StPendl, one of Liberty BASIC's true gurus. Stefan is a contributor at both the Official Liberty BASIC Support Group and the very popular Liberty BASIC Conforum.

The Request: *"I have read in the news letter 132 about how to make shaped windows, but it is a little bit confusing, is there a way faster then reading pixle by pixle, like setting one color as the part to be deleted, and if thats what the other examples do, can i make the drawing without haveing to use api calls for drawing. my whole goal in total is to make an icon sized animal walk around on the screen. I have a program that does it already, but i would like to make my own thing. Thanks in advance."* `Message #31512`

Stefan's Reply: *"How about making a fullscreen graphic window, make it transparent and use simple sprites ???"* `Message #31533`

Stefan's Demo `Message #31576` *Be sure to run this demo from your* Liberty BASIC *root directory so the* **/SPRITES** folder *and* **cave1.bmp** *can be found.*

```
'---code start
nomainwin
WindowWidth = DisplayWidth
WindowHeight = DisplayHeight

' You may want to change this path to your own path
LBFolder$ = "C:\Programme\Liberty BASIC v4.03"
' or simply
' LBFolder$ = DefaultDir$

graphicbox #m.g, 0, 0, WindowWidth, WindowHeight

'set the window to layered and topmost, to keep sprite visible
'and have access to the windows beneeth
WS.EX.LAYERED = hexdec("80000")
stylebits #m, 0, 0, WS.EX.LAYERED or _WS_EX_TOPMOST, 0

open "Sprite on Transparent Background" for window_popup as #m
#m "trapclose quit"
#m.g "down"

'make the white portion of the window transparent
hwndMain = hwnd(#m)
LWA.COLORKEY = 1
COLORREF = hexdec("FFFFFF")
calldll #user32, "SetLayeredWindowAttributes", _
hwndMain as ulong, _
COLORREF as ulong, _
bAlpha as short, _
LWA.COLORKEY as long, _
result as long

loadbmp "man", LBFolder$ + "\sprites\cave1.bmp"
loadbmp "man1", LBFolder$ + "\sprites\cave2.bmp"
#m.g "addsprite test man man1"
#m.g "home;posxy xPos yPos"
#m.g "spritexy test "; xPos; " "; yPos
#m.g "cyclesprite test 1"

call moveSprite "", "", xPos, yPos
timer 100, drawFrame
wait

sub quit handle$
unloadbmp "man"
close #handle$
```

```
end
end sub

sub moveSprite handle$, sprite$, xPosOld, yPosOld
xPos = int(rnd(0) * DisplayWidth) + 1
yPos = int(rnd(0) * DisplayHeight) + 1

if xPosOld < xPos then
#m.g "spriteorient test normal"
else
#m.g "spriteorient test mirror"
end if

#m.g "spritetravelxy test "; xPos; " "; yPos; " 15 moveSprite"
end sub

sub drawFrame
#m.g "drawsprites"
wait ' †Added to original code by Janet, see footnote for discussion
end sub
'---code end
```

† *A recently identified* bug in with a Timer Sub Event *using* Liberty BASIC *is easily fixed by adding a WAIT statement in that fired sub event.*

How cool is this!!

Closing Stefan's Demo

It may be difficult to regain focus of a transparent window. If you are finding difficulty closing Stefan's demo, press **Alt - Tab** until `Sprite on Transparent Background` is selected. Then press **Alt - F4**.

How Does Stefan's Demo Work?

In Stefan's Demo, a popup window is used. The background is white. White (&FFFFFF or 16777215) is set as the transparent color and no white pixels are drawn. Since the entire window is white, effectively, no pixel of the window is drawn. The window still provides a background for the sprite, though, and the sprite is content to bounce around that invisible window.

If you change

```
open "Sprite on Transparent Background" for window_popup as #m
```

to

```
open "Sprite on Transparent Background" for window as #m
```

you'll see more clearly how only the white pixels (and the graphicbox is comprised of all white pixels) are not drawn. All non-white pixels, in this case, the window caption and borders are drawn as usual. Notice the distortion of the title bar as the white pixels of the title have not been drawn.

In Stefan's demo, the Stylebits `_WS_EX_TOPMOST` is used to keep the invisible window on top of all other windows, even if that window isn't the window with focus. This is essential to keeping the sprite visible to the viewer. Since the invisible window spans the entire display screen and remains topmost, how then is the user able to interact with other programs? Simply stated, the ***"SetLayeredWindowsAttributes"*** allows all mouseclicks to filter down through the invisible pixels to the window underneath.

Reversing Transparency

There may be times when you will want to restore the translucent window to the default opacity of `255` or undo the single color transparency. To restore opacity, simply issue another call to ***"SetLayeredWindowAttributes"*** passing the parameters `bAlpha` with the value of `255` and `dwFLAGS` with the value of `LWA_ALPHA`, or `2`.

When the passed `dwFLAGS` parameter is `LWA_COLORKEY`, the transparent color can be made opaque by assigning `ColorKey` a value of a non present color or even by passing `bAlpha` with a value of `255` and `dwFLAGS` with the value of `LWA.COLORKEY` or `2`. Either might be effective in many cases, but the best method is to simply remove the Layered attribute. Get the current value of the `extendedAddBits` of the window with ***"GetWindowLongA"***. To this result, *subtract* the value of `WS.EX.LAYERED` using the Bitwise Operand `XOR`. Write the new value back to the window by calling ***"SetWindowLongA"***.

```
' Demo to demonstrating defining and reversing transparency
' Set the Variables
    WS.EX.LAYERED = 524288
' WS_EX_LAYERED Unrecognized Windows constant
    ColorRef = 255
' Long Pixel Color, ColorRef is only relevant when dwFLAGS = LWA.COLOR
KEY
    bAlpha = 128
' Degree of opacity (0 - 255), bAlpha is only relevant when dwFLAGS =
LWA.ALPHA
```

```
    LWA.COLORKEY = 1 ' LWA_COLORKEY Unrecognized Windows constant
    LWA.ALPHA = 2 ' LWA_ALPHA Unrecognized Windows constant
' Choose just ONE of these next two lines
    dwFLAGS = LWA.COLORKEY
' Action is transparency of just one color (LWA_COLORKEY)
'    dwFLAGS = LWA.ALPHA ' Action is transparency of entire window (LW
A_ALPHA)

' Specify variables holding Windows constants as Global
    Global WS.EX.LAYERED, LWA.COLORKEY, LWA.ALPHA

    Nomainwin

    WindowWidth = 600
    WindowHeight = 400

    BackgroundColor$ = "Red"
    ForegroundColor$ = "Blue"

    UpperLeftX = int((DisplayWidth-WindowWidth)/2)
    UpperLeftY = int((DisplayHeight-WindowHeight)/2)

    text1$ = "The red pixels have been rendered transparent " + _
        "in this demo.  This demo will NOT work on Windows " + _
        "98 or Windows 95."

    text2$ = "Translucence here is initially set at 128.  Try " + _
        "setting bAlpha to higher and lower numbers for desired " + _
        "transparency.  0 will make your window complete " + _
        "transparent.  255 is the maximum.  This demo will " + _
        "NOT work on Windows 98 or Windows 95."

    If dwFLAGS = 1 Then
        Button #main.btn, " Show All Pixels "
, ShowAllPixels, UL, 150, 250, 200, 30
    Else
        Button #main.btn, " FullOpacity "
, FullOpacity, UL, 150, 250, 200, 30
    End If

    Statictext #main.msg, "", 20, 20, 550, 200
    Stylebits #main, 0, 0, WS.EX.LAYERED, 0
    Open "Transparent Window" for Window as #main

    #main "Trapclose EndDemo"
    hMain = hWnd(#main)
    #main, "Font Verdana 12 Bold"

    If dwFLAGS = 1 Then
        #main.msg text1$
    Else
        #main.msg text2$
    End If
    Call SetLayeredWindowAttributes, hMain, ColorRef, bAlpha, dwFLAGS

    Wait

Sub EndDemo handle$
    Close #main
    End
End Sub

Sub SetLayeredWindowAttributes hWindow, ColorRef, bAlpha, dwFLAGS
    CallDLL #user32, "SetLayeredWindowAttributes", _
        hWindow as Ulong, _
        ColorRef as Long, _
        bAlpha as Long, _
        dwFLAGS as Long, _
        result as Long
End Sub

Sub FullOpacity handle$
    Call SetLayeredWindowAttributes hWnd(#main), 0, 255, 2
    #main.msg, "The window is now 100% opaque."
    #main.btn, "!Hide"
End Sub

Sub ShowAllPixels handle$
    hMain = hWnd(#main)
    CallDLL #user32, "GetWindowLongA", _
        hMain as Ulong, _
        _GWL_EXSTYLE as Long, _
        style as Long
' Remove the extendedAddBits WS.EX.LAYERED
        newStyle = style xor WS.EX.LAYERED
    CallDLL #user32, "SetWindowLongA", _
        hMain as Ulong, _
        _GWL_EXSTYLE as Long, _
        newStyle as Long, _
        result as Long
    #main.msg, "The red pixels have been repainted."
    #main.btn, "!Hide"
```

```
End Sub
```

A special ***Thank You*** to Alyce Watson - alyce and Stefan Pendl - StPendl for their continued and unfailing willingness to mentor and share expertise. - JanetTerra

---