PlgBlt

-
Alyce

Parallelogram Blitting | PlgBlt | Demo One | Masking | Masking Demo *Some text below is copied from the Microsoft Developers Network Library.*

For an eBook or printed book on using the API with Liberty BASIC, see:
APIs for Liberty BASIC

Parallelogram Blitting

The **PlgBlt** function performs a bit-block transfer of the bits of color data from the specified rectangle in the source device context to the specified parallelogram in the destination device context.

Bitmaps are rectangular images. PlgBlt displays a copy of the bitmap skewed into the specified parallelogram shape.

PlgBlt

The parameters of the PlgBlt function are as follows.

```
calldll #gdi32, "PlgBlt",_
    hdcDest hdcDest  as ulong,_  'source device context
    lpPoint as struct,_   'array of points
    memdc as ulong,_      'destination device context
    nXSrc as long,_       'upper left x-coord of source
    nYSrc as long,_       'upper left y-coord of source
    nWidth as long,_      'width of source
    nHeight as long,_     'height of source
    hbmMask as ulong,_    'mask bmp handle, null if not used
    xMask as long,_       'ulx mask
    yMask as long,_       'uly mask
    re as long            'nonzero=success
```

hdcDest: A handle to the destination device context.

lpPoint: A pointer to an array of three points in logical space that identify three corners of the destination parallelogram. The upper-left corner of the source rectangle is mapped to the first point in this array, the upper-right corner to the second point in this array, and the lower-left corner to the third point. The lower-

right corner of the source rectangle is mapped to the implicit fourth point in the parallelogram.

hdcSrc: A handle to the source device context.

nXSrc: The x-coordinate, in logical units, of the upper-left corner of the source rectangle.

nYSrc: The y-coordinate, in logical units, of the upper-left corner of the source rectangle.

nWidth: The width, in logical units, of the source rectangle.

nHeight: The height, in logical units, of the source rectangle.

hbmMask: A handle to an optional monochrome bitmap that is used to mask the colors of the source rectangle.

xMask: The x-coordinate, in logical units, of the upper-left corner of the monochrome bitmap.

yMask: The y-coordinate, in logical units, of the upper-left corner of the monochrome bitmap.

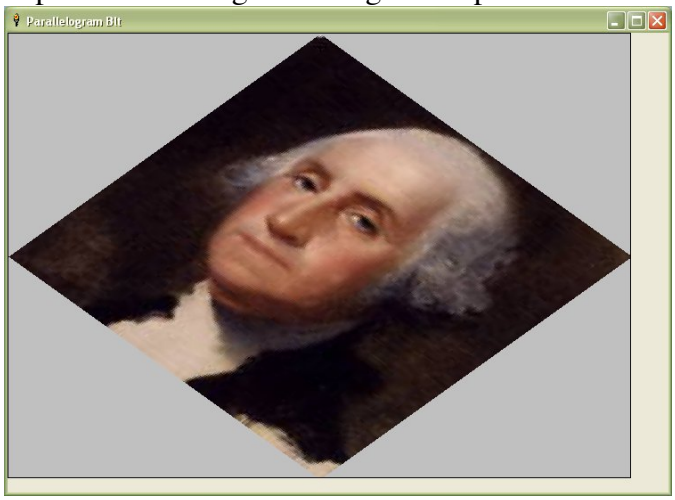
Return value: If the function succeeds, the return value is nonzero.

Not all devices support the PlgBlt function. If the source and destination device contexts represent incompatible devices, PlgBlt returns an error. When used in a multiple monitor system, both hdcSrc and hdcDest must refer to the same device or the function will fail.

Demo One

The example below allows the user to open a bitmap file. It is then displayed in a graphicbox in a specified parallelogram shape. The simple deom does not include a mask.

A picture of George Washington is opened in the demo program. The result looks like this:



```
    filedialog "Open","*.bmp",bmp$
    if bmp$="" then end
    loadbmp "image",bmp$
    hImage=hbmp("image")

    struct BITMAP,_
    bmType as long,_
    bmWidth As long,_
    bmHeight As long,_
    bmWidthBytes As long,_
    bmPlanes as word,_
    bmBitsPixel as word,_
    bmBits as Long
    length=len(BITMAP.struct)
    calldll #gdi32, "GetObjectA", hImage as ulong,_
       length as long,BITMAP as struct,_
       results as long
    bmpWidth=BITMAP.bmWidth.struct
    bmpHeight=BITMAP.bmHeight.struct

nomainwin
winWide=700:winHigh=500
WindowWidth=winWide+50:WindowHeight=winHigh+50
UpperLeftX=1:UpperLeftY=1

graphicbox #1.g, 0,0,winWide,winHigh
open "Parallelogram Blt" for window as #1
#1 "trapclose [quit]"
#1.g "down;fill lightgray"

h=hwnd(#1.g)  'graphicbox handle

'get device context for window:
calldll #user32, "GetDC",_
    h as ulong,_ 'graphicbox handle
    hdc as ulong 'returns handle to device context

CallDLL #gdi32,"CreateCompatibleDC",_
    hdc as uLong,_
    memdc as uLong

    CallDLL #gdi32,"SelectObject",_
    memdc as uLong,_
```

```
    hImage as uLong,_
    oldObject as uLong

STRUCT lpPoint,_
    x1 as long,_  'ulx
    y1 as long,_  'uly
    x2 as long,_  'urx
    y2 as long,_  'ury
    x3 as long,_  'llx
    y3 as long    'lly

'The STRUCT must be filled before it can be used in an api call:
    lpPoint.x1.struct = int(winWide/2)
    lpPoint.y1.struct = 0
    lpPoint.x2.struct = winWide
    lpPoint.y2.struct = int(winHigh/2)
    lpPoint.x3.struct = 0
    lpPoint.y3.struct = int(winHigh/2)

calldll #gdi32, "PlgBlt",_
    hdc as ulong,_        'device context of graphicbox
    lpPoint as struct,_   'array of points
    memdc as ulong,_      'memory DC
    0 as long,_           'ulx source
    0 as long,_           'uly source
    bmpWidth as long,_    'width source
    bmpHeight as long,_   'height source
    0 as ulong,_          'mask bmp handle, null = not used
    0 as long,_           'ulx mask
    0 as long,_           'uly mask
    re as long            'nonzero=success

'method to flush GDI graphics:
'#1.g "getbmp pix 0 0 ";winWide;" ";winHigh
'#1.g "drawbmp pix 0 0;flush"

wait

[quit]
calldll #user32, "ReleaseDC",_
    h as ulong,_    'window handle
    hdc as ulong,_  'device context
    ret as long

    CallDLL #gdi32, "DeleteDC",memdc as uLong, r As long
    close #1:end
```

Masking

If the given bitmask handle identifies a valid monochrome bitmap, the function uses this bitmap to mask the bits of color data from the source rectangle. If the mask rectangle is smaller than the source and destination rectangles, the function replicates the mask pattern.

If the bitmask exists, a value of one in the mask indicates that the source pixel color should be copied to the destination. A value of zero in the mask indicates that the destination pixel color is not to be changed.

Liberty BASIC's **loadbmp** statement does not load a monochrome bitmap. Instead, we use the API function **LoadImageA**. The flags for loading should include the flag for loading from a file and the flag for a monochrome bitmap. Be sure to use a mask bitmap that contains only white and black. The **LoadImageA** function looks like this:

```
imagePath$=DefaultDir$;"\plg_mask.bmp"
flags = _LR_MONOCHROME or _LR_LOADFROMFILE

calldll #user32, "LoadImageA",_
0 as ulong,_                'instance - use 0 for image from file
imagePath$ as ptr,_         'path and filename of image
_IMAGE_BITMAP as long,_     'type of image
bmpWidth as long,_          'desired width
bmpHeight as long,_         'desired height
flags as long,_             'load flags
hMask as Ulong              'handle of loaded image
```

The mask used looks like this:

Masking Demo

The code below looks like this when run:

```
filedialog "Open","*.bmp",bmp$
if bmp$="" then end
loadbmp "image",bmp$
hImage=hbmp("image")

struct BITMAP,_
bmType as long,_
bmWidth As long,_
bmHeight As long,_
bmWidthBytes As long,_
bmPlanes as word,_
bmBitsPixel as word,_
bmBits as Long
length=len(BITMAP.struct)
calldll #gdi32, "GetObjectA", hImage as ulong,_
   length as long,BITMAP as struct,_
   results as long
bmpWidth=BITMAP.bmWidth.struct
bmpHeight=BITMAP.bmHeight.struct

'need monochrome bmp
'create black and white bmp and save to disk
'use LoadImageA with monochrome flag set
'    to assure handle to monochrome bmp

imagePath$=DefaultDir$;"\plg_mask.bmp"
flags = _LR_MONOCHROME or _LR_LOADFROMFILE

calldll #user32, "LoadImageA",_
0 as ulong,_                'instance - use 0 for image from file
imagePath$ as ptr,_         'path and filename of image
_IMAGE_BITMAP as long,_     'type of image
```

```
    bmpWidth as long,_          'desired width
    bmpHeight as long,_         'desired height
    flags as long,_             'load flags
    hMask as Ulong              'handle of loaded image


nomainwin
winWide=700:winHigh=500
WindowWidth=winWide+50:WindowHeight=winHigh+50
UpperLeftX=1:UpperLeftY=1

graphicbox #1.g, 0,0,winWide,winHigh
open "Masked Parallelogram Blt" for window as #1
#1 "trapclose [quit]"
#1.g "down;fill lightgray"

h=hwnd(#1.g)  'graphicbox handle

'get device context for window:
calldll #user32, "GetDC",_
    h as ulong,_ 'graphicbox handle
    hdc as ulong 'returns handle to device context

CallDLL #gdi32,"CreateCompatibleDC",_
    hdc as uLong,_
    memdc as uLong

    CallDLL #gdi32,"SelectObject",_
    memdc as uLong,_
    hImage as uLong,_
    oldObject as uLong

STRUCT lpPoint,_
    x1 as long,_  'ulx
    y1 as long,_  'uly
    x2 as long,_  'urx
    y2 as long,_  'ury
    x3 as long,_  'llx
    y3 as long    'lly

'The STRUCT must be filled before it can be used in an api call:
    lpPoint.x1.struct = int(winWide/2)
    lpPoint.y1.struct = 0
    lpPoint.x2.struct = winWide
    lpPoint.y2.struct = int(winHigh/2)
    lpPoint.x3.struct = 0
```

```
    lpPoint.y3.struct = int(winHigh/2)

calldll #gdi32, "PlgBlt",_
    hdc as ulong,_          'device context of graphicbox
    lpPoint as struct,_     'array of points
    memdc as ulong,_        'memory DC
    0 as long,_             'ulx source
    0 as long,_             'uly source
    bmpWidth as long,_      'width source
    bmpHeight as long,_     'height source
    hMask as ulong,_        'mask monochrome bmp handle
    0 as long,_             'ulx mask
    0 as long,_             'uly mask
    re as long              'nonzero=success

'method to flush GDI graphics:
'#1.g "getbmp pix 0 0 ";winWide;" ";winHigh
'#1.g "drawbmp pix 0 0;flush"

wait

[quit]
calldll #user32, "ReleaseDC",_
    h as ulong,_     'window handle
    hdc as ulong,_   'device context
    ret as long

    CallDLL #gdi32, "DeleteDC",memdc as uLong, r As long
    close #1:end
```

GDI Tutorials Home