Bitmaps in the Device Context

-
Alyce

SelectObject | hBmp | CreateCompatibleBmp | DeleteObject | Demo | Transferring Bits *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

SelectObject

In earlier lessons we discussed the loading of bitmaps. We also discussed device contexts, both for a window and in memory.

Memory device contexts contain a default, monochrome bitmaps that is 1 pixel wide and 1 pixel high. We can select another bitmap into a memory device context with SelectObject. It looks like this:

```
    CallDLL #gdi32,"SelectObject",_
    memoryDC as uLong,_    'handle to memory device context
    hObject as uLong,_     'handle of bitmap
    oldObject as uLong
'returns handle to previously selected object
```

The bitmap object can be a bitmap that was loaded into memory with Liberty BASIC's LOADBMP command, or with the API call to LoadImageA.

hBmp

We see that the SelectObject function requires a handle to the object to be selected. If we used the LoadImageA API call to load the image, we already have the handle. If we used Liberty BASIC's ability to load a bitmap and give it a "name" we must use the HBMP() function to retrieve the Windows handle to the bitmap. It looks like this:

```
loadbmp "example", "filename.bmp"
hBitmap = hbmp("example")
```

CreateCompatibleBmp

It is possible to create a bitmap in memory. This is done with CreateCompatibleBitmap. It is important to remember that the memory bitmap must be created to be compatible with a display or window device context, not with a memory device context.

```
calldll #gdi32, "CreateCompatibleBitmap",_
    hDC AS ulong,_ 'window DC, NOT memory DC
    nWidth AS long,_ 'width of created bitmap
    nHeight AS long,_ 'height of created bitmap
    handleBMP AS ulong 'returns handle if successful
```

The memory bitmap must be selected into the device context that was created in memory with CreateCompatibleDC. All graphics commands to the memory device context will affect the memory bitmap that is currently selected into it.

```
CallDll #gdi32, "SelectObject",_
    hdcMem as ulong,_    'memory device context
    handleBMP as ulong,_ 'handle of memory bitmap
    oldBmp as ulong      'returns handle of previous object
```

A program can have many memory bitmaps. Each device context can hold only one memory bitmap at a time. A bitmap that is selected into a memory device context replaces the previous bitmap in that context. The handle to the previous bitmap is returned by the SelectObject call.

DeleteObject

When a memory bitmap is no longer needed, delete it with DeleteObject:

```
calldll #gdi32,"DeleteObject",_
    handleBMP as ulong,_ 'handle of memory bitmap
    r as long         'nonzero if successfull
```

Do not attempt to delete an object that is currently selected into a device context. Retain the handle of the original, default bmp from the DC and use SelectObject to select it back into the DC. When the memory bmp is not longer selected into a device context it can be deleted.

```
    'select default bmp back into DC
    CallDLL #gdi32,"SelectObject",_
    hMemDC as uLong,_     'memory DC
    oldBmp as uLong,_     'handle of original, default bmp
    handle as uLong       'returns previously selected bitmap
```

Demo

The following demonstration program does not display any graphics. It is presented as a framework for working with graphics in memory. Further lessons will build on this framework.

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

graphicbox #1.g, 0,0,winWide,winHigh
open "GDI Demo" for window as #1
    #1 "trapclose [quit]"
    #1.g "down"

    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,_  'graphicbox DC
    hMemDC as ulong 'memory DC

    nWidth=100 : nHeight=200
    calldll #gdi32, "CreateCompatibleBitmap",_
    hdc AS ulong,_     'window DC, NOT memory DC
    nWidth AS long,_   'width of created bitmap
    nHeight AS long,_  'height of created bitmap
    handleBmp AS ulong 'returns handle if successful

    CallDLL #gdi32,"SelectObject",_
    hMemDC as uLong,_     'memory DC
    handleBmp as uLong,_  'handle of bmp
    oldBmp as uLong       'returns previously selected bitmap
```

```
wait

[quit]
    'select default bmp back into DC
    CallDLL #gdi32,"SelectObject",_
    hMemDC as uLong,_     'memory DC
    oldBmp as uLong,_     'handle of original, default bmp
    handle as uLong       'returns previously selected bitmap

    CallDLL #gdi32,"DeleteObject",_
    handleBmp as uLong,_    'handle of bmp
    r As long

    calldll #gdi32, "DeleteDC",_
    hMemDC as ulong,_   'DC to delete
    re as long          'nonzero=success

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

    close #1:end
```

Transferring Bits

We now know how to create a device context in memory. It functions as a canvas for graphics. It contains a memory bitmap. This bitmap can be displayed on the program window with GDI API calls. We'll discuss that subject in the next lesson.

Transferring Bits

GDI Tutorials Home