Bitmap Graphics Tutorial

-
Alyce

Bitmap Graphics Tutorial | WHAT IS A BITMAP? | WHAT IS A PIXEL? | HOW DO WE USE BITMAPS IN LIBERTY BASIC? | LOADBMP | RELATIVE PATH | HARD-CODED PATH | FILEDIALOG | UNLOADBMP | GETBMP | DRAWBMP | BMPSAVE | DEMOS | HBMP

WHAT IS A BITMAP?

What is a bitmap? In simplest terms, a bitmap is a rectangular picture. Computers recognize things as collections of bits, so to a computer a bitmap is a rectangular grid of pixels.

WHAT IS A PIXEL?

"Pixel" is a shortened form of the term "Picture Element". Your display is made of pixels. If your screen resolution is set to 800x600, this means that it is divided into 800 distinct picture elements across the width, and 600 distinct picture elements along the height. Each of the picture elements is colored with a distinct, single color. Altogether, they comprise the image that you see when you look at your computer screen.

Here is a way to think of it. Imagine that you have a checker board. The board consists of 64 squares - 8 squares wide by 8 squares high. Think of each of these squares as ONE pixel. Now imagine that you have a palette that contains four colors of paint - red, green, blue and yellow. You are told to color each square in the checker board with one of these colors. You must fill each square completely with only one color, and you must fill all of the squares with one of the four colors. Using the first letters of the color names, you might create a picture that looks like this:

```
R R R R R R R R
R G R G B B B B
Y Y Y R R R Y Y
B B B B B Y Y R
R Y Y Y Y B B B
G G G G Y Y Y Y
B B B B B G G G
G G Y Y R R B B
```

This is a pretty fair representation of the way your computer sees images. A bitmap file on disk consists of information about the attributes of the bitmap such as width and height at the start of the file. After this file header you will find a map of bit values. These values are like the checker board representation above. The computer knows to paint the individual pixels on the screen according to the information in this map of bit values.

HOW DO WE USE BITMAPS IN LIBERTY BASIC?

Before a bitmap can be used, it must be loaded from the disk into the computer's memory (RAM). Once the bitmap is in the computer's RAM, it can be displayed on the screen. We load bitmaps into memory with the LOADBMP command.

LOADBMP

The LoadBmp command is easy to use. It looks like this:

```
loadbmp "PicName", "filename.bmp"
```

"PicName" can be any name that suits your fancy as long as it is a single word containing only alpha-numeric characters. After giving the loaded bitmap this name, you will use it to tell Liberty BASIC that you want to access this bitmap. It is best to give the bitmap a descriptive name, like "RedCar" or "EnemyShip", but Liberty BASIC doesn't care what you call it! You can call it "xxx" or "potato" or any other name. Just remember that this name is case sensitive, so "xxx" and "XXX" are two different bitmap names.

"filename.bmp" is the name of a bitmap file on disk. If this file is in the same directory as the Liberty BASIC program then the filename alone is enough -- no path information is needed.

RELATIVE PATH

You can place your bitmap files in a subfolder of your program if you'd like. If your subfolder containing bitmaps is called "images" then you would load a bitmap from that folder like this:

```
loadbmp "PicName", "images\filename.bmp"
```

The path is "relative" to that of the running program, hence a "relative path."

HARD-CODED PATH

You can include all path information if you'd like. This is called "hard-coding" a path.

```
loadbmp "PicName", "c:\mystuff\myprogs\filename.bmp"
```

The above hard-coded method works fine IF you are the only one who will use your program, and IF you will not be moving any of the files. Attempting to load a bitmap file that cannot be found by Liberty BASIC will cause a runtime error and the program will stop running. It is almost always best to avoid hard-coding paths to files used by a program. If you plan to share your program with someone else, the other person is not likely to have exactly the same directory setup as you have, so a command like the one above will crash the program.

FILEDIALOG

You can allow a program's user to select a bitmap using the FILEDIALOG command, like this:

```
filedialog "Open", "*.bmp", bmpfile$
loadbmp "MyPic", bmpfile$
```

UNLOADBMP

We've explained that you must load a bitmap into the computer's memory to use it. It stays in memory until it is unloaded. If you fail to unload bitmaps used by a program, the system will become sluggish and may even fail due to lack of resources. People who use your program will not appreciate this deterioration in their session at the PC, and they certainly won't appreciate the need to reboot from a program crash! For this reason, it is important to use the UNLOADBMP command for every loaded bitmap in a program. It looks like this:

```
unloadbmp "MyPic"
```

Liberty BASIC should automatically release all memory when the program closes, but it is best to specifically UNLOADBMP for all loaded bitmaps.

GETBMP

You can load a bitmap into memory from your own Liberty BASIC graphics that have been drawn in a graphicbox or graphics window. This method uses the GETBMP command:

```
print #graphics, "getbmp bmpname x y width height"
```

"bmpname" will be the name you give to the bitmap, just as you did when using the LOADBMP command to load a bitmap from disk. "x" will be the coordinate of the left side of the bitmap you "GET". "y" will be the coordinate of the top side of the bitmap. The width and height parameters will be the desired width and height of the bitmap. The code above will NOT run properly in Liberty BASIC! The actual values must be placed in the command. In a program, it might look like this:

```
print #graphics, "getbmp MyPic 10 14 200 100"
```

This command tells Liberty BASIC to load a bitmap into memory that is taken from the graphics window whose handle is #graphics. The upper left corner of the bitmap is at x=10, y=14. The width is 200 and the height is 100. The Liberty BASIC name for this loaded bitmap will be "MyPic". If you will be using variables in the command, then place them outside of the quote marks, making sure to preserve needed spaces in between the values like this:

```
picname$ = "MyPic"
x = 10
y = 14
w = 200
h = 100

print #graphics, "getbmp ";picname$;" ";x;" ";y;" ";w;" ";h
```

Be sure to UNLOADBMP for bitmaps loaded with GETBMP.

DRAWBMP

Once a bitmap has been loaded into memory with LOADBMP or GETBMP, it can be displayed in a graphics window or graphicbox with the DRAWBMP command.

```
print #graphics, "drawbmp bmpname x y"
```

Just as before, you must either include the actual values inside of the quote marks or place variables outside of the quote marks. To display a bmp that was loaded with the name "MyPic" at x=10 y=20:

```
print #graphics, "drawbmp MyPic 10 20"

'or

x = 10
y = 20
bmpname$ = "MyPic"
print #graphics "drawbmp ";bmpname$;" ";x;" ";y
```

BMPSAVE

You can save a bitmap to disk that is loaded into memory. Here is the syntax:

```
bmpsave "bmpname", "filename.bmp"
```

Just as when opening bitmaps from disk, if no path is specified, the bitmap will be saved in the same directory as the running program. As it might look in a program:

```
bmpsave "MyBmp", "test.bmp"
```

DEMOS

Here are two demos. The first one draws some graphics in a graphicbox, uses GETBMP to load the graphics into memory as a bitmap, then draws that memory bitmap onto a second graphicbox. It also saves the graphics as a bitmap on disk. The second demo allows the user to choose a bitmap on disk, loads it into memory with LOADBMP, draws it in a graphicbox, and saves it to disk with the filename specified by the user.

```
'first demo:
nomainwin
WindowWidth=300
WindowHeight=320
graphicbox #1.1, 10,10,102,102
graphicbox #1.2, 120,10,102,102
```

```
open "Bitmap Demo" for window_nf as #1
print #1, "trapclose [quit]"

'first draw some graphics:
print #1.1, "down; fill darkblue"
print #1.1, "color pink; backcolor yellow"
print #1.1, "size 4; place 50 50"
print #1.1, "circlefilled 30;flush"

'now use GETBMP to get the bitmap into memory:
print #1.1, "getbmp MyBmp 0 0 100 100"

'draw the bitmap in memory in the second graphicbox:
print #1.2, "down"
print #1.2, "drawbmp MyBmp 0 0;flush"

'save the bitmap to disk with BMPSAVE:
bmpsave "MyBmp", "test.bmp"

'unload the bitmap from memory:
unloadbmp "MyBmp"

wait

[quit]
close #1:end

'second demo:
nomainwin
WindowWidth=600
WindowHeight=420
graphicbox #1.1, 0, 0, 600, 400
open "Bitmap Demo" for window_nf as #1
print #1, "trapclose [quit]"

'allow the user to choose a bitmap file on disk:
filedialog "Open", "*.bmp", bmpfile$
if bmpfile$="" then [quit]

'load the bitmap into memory:
loadbmp "MyBmp", bmpfile$

'draw the bitmap in the graphicbox:
print #1.1, "down"
print #1.1, "drawbmp MyBmp 0 0;flush"
```

```
'let user choose a name for saving bitmap to disk:
filedialog "Save As", "*.bmp", savefile$
if savefile$="" then [quit]

'save the bitmap to disk with BMPSAVE:
bmpsave "MyBmp", savefile$

wait

[quit]
'unload the bitmap from memory:
unloadbmp "MyBmp"
close #1:end
```

HBMP

It is possible to load bitmaps into memory with Windows API calls and with third party DLLs. These APIs return a Windows handle to the bitmap in memory. Liberty BASIC can use this handle to load a bitmap so that it can be displayed with DRAWBMP and saved with BMPSAVE. The syntax, if the bitmap is loaded via Windows API and the bitmap handle is **hBmp**:

```
loadbmp "imageName", hBmp
```

Here is a demo program that loads a bmp with the LoadImageA API call, then loads and displays it with Liberty BASIC.

```
'Use LoadImageA to load bitmaps to be a specified size
nomainwin

width=200   'desired width
height=200  'desired height


menu #1, "&File", "&Open",[open],"E&xit",[quit]
graphicbox #1.g, 0,0,width,height
open "Image Size Test" for window as #1
#1 "trapclose [quit]"
wait

[open]
filedialog "Open Image","*.bmp", imagePath$
```

```
if imagePath$="" then wait

if hImage then unloadbmp "image"

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
width as long,_             'desired width
height as long,_            'desired height
_LR_LOADFROMFILE as long,_  'load flag
hImage as Ulong             'handle of loaded image

if hImage = 0 then
    notice "Cannot load image."
    wait
end if

loadbmp "image", hImage     'get the LB loadbmp name

print #1.g, "down"
print #1.g, "drawbmp image 0 0"

wait

[quit]
if hImage then unloadbmp "image"
close #1:end
```