Sprite Byte Tutorials Lesson One: The Absolute Minimum!

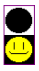
by Alyce Watson - http://alycesrestaurant.com/
-
Alyce

Table of Contents

Let's start at the very beginning!

Liberty BASIC has native sprite support. It is available in windows of type "graphics" and in graphicboxes. You may only have one sprite container (graphics window or graphicbox) in your program.

What is a sprite?

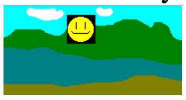
A sprite is a set of two pictures. The actual image that will appear in the window is called the sprite. The other picture is called the mask. Here is an example. The sprite is on the bottom. The mask is on the top.


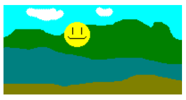
Why do you need a sprite and mask?

You can display loaded bitmaps with the DRAWBMP command. Why do you need a sprite? The answer can be found in the format of bitmaps. Bitmaps are rectangular pictures. The only way Liberty BASIC can put them on the screen is to put the whole rectangle onto the graphics control.

Here is a picture of a smiley face.

Here is a background picture.

Here's what you see if you use DRAWBMP to place the smiley face on the background.


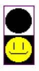
That doesn't look right, does it? We want it to look like this:


That's why we need both a sprite and a mask.

How do you create a sprite?

To create a sprite, make sure that the image fills as much as possible of the bitmap that contains it. That will be helpful for collision checking later on. Any part of the image that should not show on the background must be colored pure black. Here is an example:

How do you create a mask for a sprite?

In your Paint program, expand the sprite image so that it is twice as high as the original image. Locate the sprite image at the bottom of the bitmap. The top of the bitmap will contain the mask. The mask must be the same width and height as the sprite. Every pixel on the sprite that is meant to be visible when you put the sprite onto the background must be pure black in the mask. Every other pixel must be pure white in the mask. The sprite in the figure above, along with its mask, looks like this:


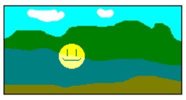
A future tutorial in this series will go into more detail about sprite and mask creation.

How does this method work?

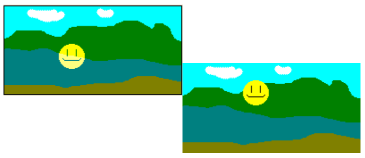
The method works because of Windows raster operations. Raster operation codes define how the color data for the bitmap source rectangle is to be combined with the color data for the destination rectangle to achieve the final color. The destination is the image already on your graphicbox, and the source is the sprite or mask.

The mask is put onto the background first. It is transferred using _SRCAND. This operation combines the colors of the source and destination rectangles by using the Boolean AND operator. This really means that the white pixels on the mask are ignored when it is placed on the background. Only the black pixels are transferred. If you put only the mask onto the screen without the sprite, it looks like this:

After the mask is transferred to the screen, the sprite image is placed onto the background, using _SRCPAINT. This raster operation combines the colors of the source and destination rectangles by using the Boolean OR operator. In plain terms, this means that the black pixels are not displayed on the background. Because of the way pixels are combined, the remainder of the sprite needs to be transferred onto a completely black background. If you only put the sprite on without a mask underneath, it looks like this:



You can see that it looks semi-transparent. If you don't want your sprite to look like a ghost, you must use a mask! Look at the difference side-by-side:



Adding a sprite.

Before you can add a sprite to a graphics window or graphicbox, you must load the bitmap with LOADBMP.

```
    loadbmp "smiley1", "sprites\smiley1.bmp"
```

The last string argument in that command is the disk filename of the bitmap to be loaded. The first string is the Liberty BASIC name you are giving to the bitmap. After it is loaded, you can refer to it by this name in commands such as DRAWBMP, BMPSAVE and ADDSPRITE. The bitmap in the example above is given the Liberty BASIC name "smiley1".

Here is the syntax for adding a sprite:

```
    #handle "addsprite spritename bitmapname"
```

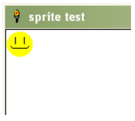
A spritename is not the same as the Liberty BASIC bitmap name. It *can* be the same, but it can be any name you want. In the following code, the bitmap name is "smiley1" and the spritename is "smiley."

```
    #wg "addsprite smiley smiley1"
```

Updating the display, or where is my sprite?
Added sprites don't show until you issue a DRAWSPRITES command! Each time you make a change in the location or other attributes of a sprite, you must issue this command to update the display.

```
    #wg "drawsprites"
```

Here's the result of adding a sprite and issuing a DRAWSPRITES command:



Locating a sprite.

By default, sprites are located at the point 0,0. You can put them any place you want, even at negative coordinates. To locate a sprite in your graphicbox or graphics window, use the SPRITEXY command, like this:

```
    #handle "spritexy spritename X Y"
```

The X, Y coordinates specify where to draw the upper left corner of the sprite. Here is the actual code from the demo, which locates the sprite at x equal to 30, y equal to 40:

```
    #wg "spritexy smiley 30 40"
```

The command must contain literal values inside of the quotation marks. To do the same thing using variables, place the variables outside of the quotation marks, preserving the blank spaces within the quotation marks, like this:

```
    xLoc = 30
    yLoc = 40
    #wg "spritexy smiley ";xLoc;" ";yLoc"
```

DEMO

Here is a tiny demonstration program. You can copy the code and paste it into the Liberty BASIC editor to run it. It loads a bitmap, adds a sprite, locates it in the graphics window and causes it to be displayed.

```
'Run this program from the root LB directory
'so that it can find the sprite bitmaps.
nomainwin

    'this is the sprite bitmap
    loadbmp "smiley1", "sprites\smiley1.bmp"

    WindowHeight = 300:WindowWidth = 400

    'sprites can only appear in a graphics window or graphicbox
    open "sprite test" for graphics_nf_nsb as #wg

    'trap the close event so the code can issue
    'the proper CLOSE and END commands
    #wg "trapclose [quit]"

    'add a sprite
    #wg "addsprite smiley smiley1"

    'give a location to sprite:
    #wg "spritexy smiley 30 40"

    'cause images to show
    #wg "drawsprites"


'wait for user input
wait

[quit]
    unloadbmp "smiley1"
    close #wg:end
```

Table of Contents