Sprite Byte Tutorials Lesson Eight: Manipulating the Background

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

Table of Contents

Background from Graphics Commands

We can use Liberty BASIC graphics commands to create a background for the sprites. See the graphics commands topic in the helpfile for drawing and coloring graphic objects such as boxes, circles and lines. Liberty BASIC also allows us to change the fill color, foreground color, background color and the width of the pen.

Here are some simple graphics that fill the screen with cyan, set the foreground and background color, and draw some filled boxes.

```
'draw shapes
#w.g "fill cyan; color green; backcolor brown; size 3"
#w.g "place 10 10; boxfilled 40 60"
#w.g "place 3 100; boxfilled 120 160"
#w.g "place 150 10; boxfilled 240 80"
#w.g "place 320 110; boxfilled 380 260"
```

Once the desired graphics have been drawn on the graphicbox, we use the **GETBMP** command to capture the image and give it a name. **GETBMP** requires the XY location of the upper left corner, as well as the width and height of the captured bitmap.

print #handle, "getbmp bmpName x y width height"

GETBMP captures the image displayed on the screen, including parts that are outside the graphicbox. We want to avoid capturing the graphicbox border as part of our bitmap, so we use a width and height that are 2 pixels smaller than the width and height of the graphicbox. We'll give it the name "landscape".

```
 'capture image and give it a name
 #w.g "getbmp landscape 0 0 398 298"
```

Now we can cause that captured image to be the sprite background, using the **BACKGROUND** command.

```
 'make captured image the background for sprites
 #w.g "background landscape"
```

Here is a demo that uses drawn objects to create a sprite background.

```
'Sprite Byte 8 Demo
'background from graphics commands

nomainwin
loadbmp "crab1", "sprites\crab1.bmp"
WindowHeight = 350 : WindowWidth = 408
graphicbox #w.g, 0, 0, 400, 300
statictext #w.s, "",40,300,100,40
open "Drawn Background" for window_nf as #w
 #w "trapclose [quit]"
 #w.g "down"

 'draw shapes
 #w.g "fill cyan; color green; backcolor brown; size 3"
 #w.g "place 10 10; boxfilled 40 60"
 #w.g "place 3 100; boxfilled 120 160"
 #w.g "place 150 10; boxfilled 240 80"
 #w.g "place 320 110; boxfilled 380 260"
 'capture image and give it a name
 #w.g "getbmp landscape 0 0 398 298"

 'make captured image the background for sprites
 #w.g "background landscape"

 'add a sprite
 #w.g "addsprite crab crab1"
 'original location of crab
 crabX = 360 : crabY = 100
 'add variable for increment to move crab each time
 moveCrabX = -5 : moveCrabY = 2
 #w.g "spritexy crab ";crabX;" ";crabY
 #w.g "drawsprites" 'update screen
 'set up a timer to move crab sprite
 timer 100, [updateDisplay]
wait

[updateDisplay]
 'locate sprite at new positions and update display
 'add boundary detection and reverse direction at edges
 if (crabX > 370) then moveCrabX = -5
 if (crabX < 10) then moveCrabX = 5
 if (crabY > 270) then moveCrabY = -2
 if (crabY < 10) then moveCrabY = 2
 'move computer-controlled sprite at timer intervals
 crabX = crabX + moveCrabX : crabY = crabY + moveCrabY
 #w.g "spritexy crab ";crabX;" ";crabY
 #w.g "drawsprites"
 wait
[quit]
 timer 0
 unloadbmp "landscape"
 unloadbmp "crab1"
 close #w : end
```

Background from Loaded Bitmap

In previous lessons we used a loaded bitmap as the background. We do this by issuing the **LOADBMP** command before we issue the **BACKGROUND** sprite command.

```
'load a bitmap from file
loadbmp "landscape", "sprites\bg3.bmp"
```

```
'cause loaded bitmap to be the sprite background
 #w.g "background landscape"
```

Here us a demo that uses a loaded bitmap as the sprite background.

```
'Sprite Byte 8 Demo
'background from loaded bitmap

nomainwin
loadbmp "crab1", "sprites\crab1.bmp"
WindowHeight = 350 : WindowWidth = 408
graphicbox #w.g, 0, 0, 400, 300
statictext #w.s, "",40,300,100,40
open "Loaded Bitmap Background" for window_nf as #w
 #w "trapclose [quit]"
 #w.g "down"

'load a bitmap from file
loadbmp "landscape", "sprites\bg3.bmp"
'cause loaded bitmap to be the sprite background
 #w.g "background landscape"

 'add a sprite
 #w.g "addsprite crab crab1"
 'original location of crab
 crabX = 360 : crabY = 100
 'add variable for increment to move crab each time
 moveCrabX = -5 : moveCrabY = 2
 #w.g "spritexy crab ";crabX;" ";crabY
 #w.g "drawsprites" 'update screen
 'set up a timer to move crab sprite
 timer 100, [updateDisplay]
wait

[updateDisplay]
 'locate sprite at new positions and update display
 'add boundary detection and reverse direction at edges
 if (crabX > 370) then moveCrabX = -5
 if (crabX < 10) then moveCrabX = 5
 if (crabY > 270) then moveCrabY = -2
 if (crabY < 10) then moveCrabY = 2
 'move computer-controlled sprite at timer intervals
 crabX = crabX + moveCrabX : crabY = crabY + moveCrabY
 #w.g "spritexy crab ";crabX;" ";crabY
 #w.g "drawsprites"
 wait
[quit]
 timer 0
 unloadbmp "landscape"
 unloadbmp "crab1"
 close #w : end
```

Changing the Background Image

We can change the sprite background at any time with the **BACKGROUND** command. The following demo uses drawn graphics as the sprite background. When the user presses the button, the background is changed to be a loaded bitmap.

```
'Sprite Byte 8 Demo
'change background

nomainwin
loadbmp "crab1", "sprites\crab1.bmp"
WindowHeight = 380 : WindowWidth = 408
graphicbox #w.g, 0, 0, 400, 300
button #w.b, "Change Background",[change],UL,40,300,150,50
open "Drawn Background" for window_nf as #w
 #w "trapclose [quit]"
 #w.g "down"

 'draw shapes
 #w.g "fill cyan; color green; backcolor brown; size 3"
 #w.g "place 10 10; boxfilled 40 60"
 #w.g "place 3 100; boxfilled 120 160"
 #w.g "place 150 10; boxfilled 240 80"
 #w.g "place 320 110; boxfilled 380 260"

 'capture image and give it a name
 #w.g "getbmp landscape 0 0 398 298"

 'make captured image the background for sprites
 #w.g "background landscape"

 'add a sprite
 #w.g "addsprite crab crab1"
 'original location of crab
 crabX = 360 : crabY = 100
 'add variable for increment to move crab each time
 moveCrabX = -5 : moveCrabY = 2
 #w.g "spritexy crab ";crabX;" ";crabY
 #w.g "drawsprites" 'update screen
 'set up a timer to move crab sprite
 timer 100, [updateDisplay]
wait

[updateDisplay]
 'locate sprite at new positions and update display
 'add boundary detection and reverse direction at edges
 if (crabX > 370) then moveCrabX = -5
 if (crabX < 10) then moveCrabX = 5
 if (crabY > 270) then moveCrabY = -2
 if (crabY < 10) then moveCrabY = 2
 'move computer-controlled sprite at timer intervals
 crabX = crabX + moveCrabX : crabY = crabY + moveCrabY
 #w.g "spritexy crab ";crabX;" ";crabY
 #w.g "drawsprites"
 wait

[change]
 'load a bitmap from file
loadbmp "landscape2", "sprites\bg3.bmp"
'change background to loaded bitmap
#w.g "background landscape2"
wait

[quit]
 timer 0
 unloadbmp "landscape"
 unloadbmp "crab1"
 close #w : end
```

What is Background Scrolling?

Up until now, our sprite programs presented a static background. Scrolling the background causes the background image to move in the vertical direction, the horizontal direction, or in both directions at once.

BackgroundXY Statement

The **BACKGROUNDXY** statement causes the origin of the background bitmap to change to a new value.

print #w.g, "backgroundxy 25 20";
OR

x=25:y=20
print #w.g, "backgroundxy ";x;" ";y
This places the point x, y from the background bitmap at location 0, 0 of the sprite graphicbox or graphics window.

The X and Y values can be either positive or negative. Positive X values cause the background to move towards the left, while negative X values cause the background to move towards the right. Positive Y values cause the background to move towards the top, while negative Y values cause the background to move towards the bottom.

The background position can be changed in the X direction or the Y direction, or in both directions at the same time. Either positive or negative values can be used to locate the background bitmap.

If the background position is moved continuously, the background appears to be scrolling.

It is important to note that the sprite position is relative to the graphicbox. It is not relative to the position of the background.

We can scroll the position of the background bitmap by keeping the X value in a variable and incrementing that variable each time the display is updated.

```
 'increment background X variable
 backX = backX + 5
 'change background location on display
 #w.g "backgroundxy ";backX;" 0"
```

Here is a demo that causes the background to move (scroll) towards the left by 5 pixels each time the display is updated.

```
'Sprite Byte 8 Demo
'background from graphics commands

nomainwin
loadbmp "crab1", "sprites\crab1.bmp"
WindowHeight = 350 : WindowWidth = 408
graphicbox #w.g, 0, 0, 400, 300
open "Scrolling Background" for window_nf as #w
 #w "trapclose [quit]"
 #w.g "down"

 'draw shapes
 #w.g "fill cyan; color green; backcolor brown; size 3"
 #w.g "place 10 10; boxfilled 40 60"
 #w.g "place 3 100; boxfilled 120 160"
 #w.g "place 150 10; boxfilled 240 80"
 #w.g "place 320 110; boxfilled 380 260"

 'capture image and give it a name
 #w.g "getbmp landscape 0 0 398 298"
 'make captured image the background for sprites
 #w.g "background landscape"
 'create a background X variable
 backX=0
 'add a sprite
 #w.g "addsprite crab crab1"
 'original location of crab
 crabX = 360 : crabY = 100
 'add variable for increment to move crab each time
 moveCrabX = -5 : moveCrabY = 2
 #w.g "spritexy crab ";crabX;" ";crabY
 #w.g "drawsprites" 'update screen
 'set up a timer to move crab sprite
 timer 100, [updateDisplay]
wait

[updateDisplay]
 'increment background X variable
 backX = backX + 5
 'change background location on display
 #w.g "backgroundxy ";backX;" 0"

 'locate sprite at new positions and update display
 'add boundary detection and reverse direction at edges
 if (crabX > 370) then moveCrabX = -5
 if (crabX < 10) then moveCrabX = 5
 if (crabY > 270) then moveCrabY = -2
 if (crabY < 10) then moveCrabY = 2
 'move computer-controlled sprite at timer intervals
 crabX = crabX + moveCrabX : crabY = crabY + moveCrabY
 #w.g "spritexy crab ";crabX;" ";crabY
 #w.g "drawsprites"
 wait
[quit]
 timer 0
 unloadbmp "landscape"
 unloadbmp "crab1"
 close #w : end
```

Challenges

- challenge: add a scrolling background to the demo in the previous lesson
- challenge: add the ability to change the background to the demo in the previous lesson
- challenge: change the code above to cause the background to scroll in the Y direction
- challenge: change the code above to cause the background to scroll in both the X and Y directions

Table of Contents