HAM Tutorial :: Day 11 :: Backgrounds - Rotation

For Day 11 I'll explain how to rotate backgrounds. HAM makes this very simple with the RotBgEx() function. This function is very powerful because not only can you rotate backgrounds but also zoom in or out and scroll backgrounds.
NOTE: There is also a RotBg() function, but I prefer the newer RotBgEx() because you have more control over the rotation.

Oh, and don't forget. You can only rotate in Mode 1 and 2. In Mode 1, background 2 can rotate/scale. In Mode 2, backgrounds 2 & 3 can rotate/scale. In case you forgot, take a look at Day 1 again.

To convert the graphic, use the following:
gfx2gba -fsrc -m -protate.pal -rs -t8 rotate.bmp

Your **gfx** directory should have the following files:
rotate.map.c
rotate.pal.c
rotate.raw.c

On to the code:

```
// The Main HAM Library
#include "mygba.h"

// Graphics Includes
// gfx2gba -fsrc -m -protate.pal -rs -t8 rotate.bmp
#include "gfx/rotate.map.c"
#include "gfx/rotate.pal.c"
#include "gfx/rotate.raw.c"

// Global Variables
u32 frames = 0;
u32 zoomx = 256; // X co-ordinate of zoom center
u32 zoomy = 256; // Y co-ordinate of zoom center
u32 scrollx = 64; // X co-ordinate of scrolling center
u32 scrolly = 64; // Y co-ordinate of scrolling center
bool zoomin = 1;

// Function Prototypes
void vblFunc(); // VBL function
void query_keys(); // Query the keyboard
void redraw(); // Redraw the screen


// Main function
int main()
{
  // Initialize HAMlib
  ham_Init();

  // Setup the background mode
```

Discussions

Graphics FAQ

GFX2GBA Readme

Translations

Downloads

Games

Projects

Credits

Links

Support

```
  ham_SetBgMode(1);

  // Initialize the palettes
  ham_LoadBGPal(&rotate_Palette,256); // Background palette

  // Initialize the rotating background
  ham_bg[2].ti = ham_InitTileSet(&rotate_Tiles,
                                 SIZEOF_16BIT(rotate_Tiles),1,1);
  ham_bg[2].mi = ham_InitMapSet(&rotate_Map,256,0,1);

  // Display the background
  ham_InitBg(2,1,1,0);

  // Move the background to the center
  ham_RotBgEx(2,0,120,80,scrollx,scrolly,zoomx,zoomy);

  // Start the VBL interrupt handler
  ham_StartIntHandler(INT_TYPE_VBL,&vblFunc);

  while(1)
  {
    // Infinite loop to keep the program running
  }

  return EXIT_SUCCESS;
} // End of main()


// VBL function
void vblFunc()
{
  ham_CopyObjToOAM(); // Copy plane sprite to hardware
  query_keys(); // Check for movement
  redraw(); // Redraw the plane

  // Rotate the background
  ham_RotBgEx(2,frames%360,120,80,scrollx,scrolly,zoomx,zoomy);

  // Increment the frames
  ++frames;

  return;
} // End of vblFunc()


// Query the keyboard
void query_keys()
{
  // RIGHT only
  if (F_CTRLINPUT_RIGHT_PRESSED)
  {
    if (scrollx < 127) ++scrollx;
  }

  // LEFT only
  if (F_CTRLINPUT_LEFT_PRESSED)
  {
```

```
    if (scrollx > 0) --scrollx;
  }

  // START only
  if (F_CTRLINPUT_START_PRESSED) {
    scrollx = 64;
    scrolly = 64;
  }

  return;
} // End of query_keys()


// Redraw the screen
void redraw()
{
  if (zoomin) { // Zoom in
   if (zoomx != 396) {
    zoomx+=2;
    zoomy+=2;
   } else {
    zoomin = FALSE;
   }
  } else { // Zoom out
   if (zoomx != 116) {
    zoomx-=2;
    zoomy-=2;
   } else {
    zoomin = TRUE;
   }
  }

  return;
} // End of redraw()
```

Code Explanation

// Global Variables

u32 zoomx=256;
u32 zoomy=256;
These are used for the amount of zooming in the X and Y direction. In the HAM documentation, you will see that the author uses 0x100 for no zoom. This is a hexadecimal number which equals 256 in decimal. You can use either system when you call this function. Increasing the number increases the zoom, decreasing decreases.

u32 scrollx=64;
u32 scrolly=64;
These are used to scroll the point in the image that corresponds to the point of rotation in the X and Y directions. In this case I set them to the center of the image (it's a 128x128 pixel image). Be careful that you don't try to set the scroll point outside of the image's boundaries. This is explained in more detail later.

```
// Main()
```

You'll notice that the way I load the background here is different from how it was done in previous days. This is a shorter way to do it. Either method works fine, though.

```
// Initialize the rotating background
ham_bg[2].ti = ham_InitTileSet(&rotate_Tiles,
                               SIZEOF_16BIT(rotate_Tiles),1,1);
```

This call is exactly the same. The next part is different.

```
ham_bg[2].mi = ham_InitMapSet(&rotate_Map,256,0,1);
```

The first thing passed is the array that contains the map data. Next is "the size of the tiles to be copied into BG RAM, in number of 16bit chunks." The next number depends on the final argument (whether it is a rotation map or not). In this case it is, so the corresponding value for a 128x128 pixel image for a rotation background is 0. Finally, it is passed a 1, of course, because it is a rotation background.

NOTE: You could do the same thing with the following code:

```
// Variables
map_fragment_info_ptr bg_rotate; // Pointer to the background
// Setup the tileset for our image
ham_bg[2].ti = ham_InitTileSet(&rotate_Tiles,
                               SIZEOF_16BIT(rotate_Tiles),1,1);
// Setup the map for our image
ham_bg[2].mi = ham_InitMapEmptySet(0,1);
bg_rotate = ham_InitMapFragment(&rotate_Map,16,16,0,0,16,16,1);
ham_InsertMapFragment(bg_rotate,2,0,0);
// Display the background
ham_InitBg(2,1,0,0);
```

```
// Move the background to the center
ham_RotBgEx(2,0,120,80,scrollx,scrolly,zoomx,zoomy);
```

This is a sneak peak at RotBgEx(). As I said it is a powerful function. I am using it here not to rotate the background image, but to center it on the screen. I will rotate later in the VBL function and explain the arguments then.

```
// VBL function
```

```
// Rotate the background
ham_RotBgEx(2,frames%360,120,80,scrollx,scrolly,zoomx,zoomy);
```

Okay, now this is where the rotation (and zooming) is actually happening. The first argument passed is the background number. (It can only be 2 or 3 as you can't rotate BGs 0 and 1.) The next argument is the angle of rotation. Using *frames%360* the value will be 0 - 359. After that comes the center of rotation on the X and Y plane. I am using the center of the screen which is at 120, 80 (well, almost). Stop for a moment and make sure you understand that. This is the point around which the image will rotate. Next you pass the X and Y co-ordinates for the scroll offset. This is the point of the image that will correspond to the point of rotation. I chose 64,64 - the center of the image (again, almost). Take another moment to picture that

properly. The center of the image will rotate around the center of the screen. Finally you pass the amount you want to zoom in the X and Y direction. It starts out at 256 (or 0x100 in hex). This means no zoom. If you look at the redraw function, you will see how *zoomx* and *zoomy* are modified.

// query_keys()

if (scrollx < 127) ++scrollx;
if (scrollx > 0) --scrollx;
While the demo is running, the user can press left or right to move the point of the image that corresponds to the center of rotation. For example, if you move all the way to the left, the image will rotate with the left-hand side around the middle. Notice the 'R' in *Rotate* is near the center. If you press to the right, then the right-hand side will be near the middle and the 'e' in *Rotate* will be near the center. Oh, and if the user presses Start, it resets the scroll offset to the middle.

// redraw()

The code here is not difficult. I set the max for zooming in at 396 and the max for zooming out at 116. When a max is met, I swap *zoomin* so that it will swap between zooming in and then zooming out.

Ahh, that's it. Another Day, another wonderful lesson in the fine art of GBA programming with HAM. You should definitely tinker with the code to see what all you can do with RotBgEx() before moving on to Day 12.

Download Code

NOTE: You may need to Right-click and choose Save As.

HAM Version 2.40 And Higher
Necessary for all HAM programs: makefile
Graphics: rotate.map.c, rotate.pal.c, rotate.raw.c
Main source file: main.c
Compiled binary: hello.gba

Download All Files In One Zip: Day11_BG_Rotation.zip

View Demo Now

Discuss Day 11

<< HOME >>