HAM Tutorial :: Day 4 :: Backgrounds - Bitmap Modes

Today we'll start learning about backgrounds (also known as layers or planes) and background modes. Remember there are six background modes and up to four backgrounds (layered on top of each other) per mode.

I am going to start with the bitmap background modes, Mode 3 - 5, because they are the easiest to understand.

The first thing you have to know about a background is that it requires a certain format. HAM comes with an easy to use program called *gfx2gba* that is used to convert BMPs or PCXs to the proper format.

NOTE: Please take the time to read the Readme that comes with gfx2gba, it will save you from many headaches! The latest version of the readme can be found here.

The image should be 240x160 pixels (or 160x128 if you are using Mode 5). BMPs can be 4, 8, or 15 bits while PCXs should be 8 bits. Feel free to take a peek at the Graphics FAQ if you have questions about formats and palettes, etc.

For all of my tutorials I work exclusively with bitmaps. I've created a picture (which is quite ugly) called bg.bmp. It is a 240x160 pixel, 8 bit bitmap. This is important because ham_LoadBitmap() requires this when using Mode 4. Copy bg.bmp to your directory. For these tutorials we'll have a subdirectory **gfx** for all of the graphics used in our games. Change to that directory and type: **gfx2gba -D -fsrc -pbg.pal -t1 bg.bmp**

This will create two files: **bg.pal.c** and **bg.raw.c**

Go ahead and take a quick peek at the files. The important thing about *bg.pal.c* is the name of the array that is created. It should be **bg_Palette**. The array created in *bg.raw.c* is **bg_Bitmap**. You will see these names later in the tutorial program.

Speaking of which, let's go ahead and display that background.

Poll

Discussions

Graphics FAQ

GFX2GBA Readme

Translations

Downloads

Games

Projects

Credits

Links

Support

```
// The Main HAM Library
#include <mygba.h>

// Graphics Includes
// Created using: gfx2gba -D -fsrc -pbg.pal -t1 bg.bmp
#include "gfx/bg.raw.c"
#include "gfx/bg.pal.c"

// Function: main()
int main()
{
    // Initialize HAMlib
    ham_Init();

    // Setup the background mode
    ham_SetBgMode(4);

    // Initialize the background image palette
    ham_LoadBGPal((void*)bg_Palette,256);

    // Load the background image
    ham_LoadBitmap((void*)bg_Bitmap);

    // Flip the buffer to show the image
    ham_FlipBGBuffer();

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

    return 0;
} // End of main()
```

Code Explanation

#include "gfx/bg.raw.c"
#include "gfx/bg.pal.c"
Again these are the palette and raw file created from our original image.

ham_SetBgMode(4);
For this example we need to use Mode 4. This mode is great for dealing with the screen as if it were a big bitmap. In this mode the pixels are 8 bit indexes in a 16 bit palette. What this means is that you can store 256 16 bit colors for the background into palette memory and then access the screen as an array. The index of the color is put into the array giving you the advantage of drawing the screen with half the memory, twice as fast and allows you to have two screens in memory. These screens are also known as buffers. You can setup one screen (the back buffer) while the other is being displayed (the front buffer). When finished, you just swap the buffers. This is known as double-buffering.

ham_LoadBGPal(...)

Load the palette for our background. You pass it a void pointer to the bitmap array so that's why I 'typecast' it with *(void *)*. We want to load the entire 256 colors, so we pass it 256.

ham_LoadBitmap(...)
This function will load a bitmap graphic into the back buffer. Of course, you have to convert the graphic using a tool such as **gfx2gba**. Again, you pass it a void pointer to the bitmap array with *(void *)*.

ham_FlipBGBuffer()
When you use other bitmap mode functions, such as *ham_LoadBitmap(),* they will always draw to the back buffer. This function will flip the back buffer to the front buffer which means you can now see it on the GBA screen. All subsequent drawing will then be done on a new back buffer until you call *ham_LoadBitmap()* again.

Well, that was a bit much for today. Try creating your own image, convert it with **gfx2gba** and see if you can display it.

Download Code

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

HAM Version 2.80 And Higher
Download All Files In One Zip: Day4_BGs_Bitmap_Modes.zip

View Demo Now

Discuss Day 4

<< HOME >>