HAM Tutorial :: Day 9 :: Maps

Some of you may wonder after looking at this tutorial what the purpose is or how it is different from Day 6. Honestly, there isn't a lot that's different except that today the map size will be 512x512 and you can see how to make a map that can be scrolled around.

We are going to do things a little differently for today's graphics. The reason is that I don't want an optimized palette. If you've been paying attention, you should have noticed that the *gfx2gba* optimizes the palette by removing colors not used in your images. This is usually fine. Today, however, I want to reference a specific color when I use *ham_SetTextCol()* and it's much easier to figure out the index number of the color in Photoshop ahead of time.

So, to convert the graphics, type:
gfx2gba -fsrc -m -t8 -x simple_world.bmp

If you look at the gfx2gba readme, you'll see that *-x* prevents optimizing your palette. You'll also notice that I left out *-p<palette_name>* because *gfx2gba* ignores it when you use -x.

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

Now let's get to the code:

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

// Graphics Includes
// gfx2gba -fsrc -m -t8 -x simple_world.bmp
#include "gfx/simple_world.pal.c"
#include "gfx/simple_world.raw.c"
#include "gfx/simple_world.map.c"

// Global Variables
u8 newframe = 1; // Signifies a new frame
u16 map_x = 0;   // X position of the map
u16 map_y = 0;   // Y position of the map

// Function Prototypes
void vbl_func(); // VBL function

// Function: main()
int main()
{
    // Variables
    map_fragment_info_ptr bg_simple_world;

    // Initialize HAMlib
    ham_Init();

    // Initialize the text display system
    ham_InitText(0);

    // Setup the background mode
    ham_SetBgMode(0);

    // Initialize the palettes
    ham_LoadBGPal((void*)simple_world_Palette,256);

    // Set the Text Color
    ham_SetTextCol(195, 40);

    // Setup the tileset for our image
    ham_bg[1].ti = ham_InitTileSet(
        (void*)simple_world_Tiles,
        SIZEOF_16BIT(simple_world_Tiles),1,1);

    // Setup the map for our image
    ham_bg[1].mi = ham_InitMapEmptySet(3,0);
    bg_simple_world = ham_InitMapFragment(
```

```
        (void*)simple_world_Map,64,64,0,0,64,64,0);

    // Copy (the whole) map to BG0 at x=0, y=0
    ham_InsertMapFragment(bg_simple_world,1,0,0);

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

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

    while(1)
    {
        if(newframe)
        {
            // Write some stuff to the screen
            ham_DrawText(1,1,"512x512 Map example");
            ham_DrawText(1,2,"Use pad to scroll around");

            ham_DrawText(1,17,"scroll-x: %5d",map_x);
            ham_DrawText(1,18,"scroll-y: %5d",map_y);

            // Let the user move the map around
            if(F_CTRLINPUT_DOWN_PRESSED)
            {
                if(map_y<(512-160))
                M_BG1SCRLY_SET(map_y++)
            }
            if(F_CTRLINPUT_UP_PRESSED)
            {
                if(map_y>0)
                M_BG1SCRLY_SET(map_y--)
            }
            if(F_CTRLINPUT_LEFT_PRESSED)
            {
                if(map_x>0)
                M_BG1SCRLX_SET(map_x--)
            }
            if(F_CTRLINPUT_RIGHT_PRESSED)
            {
                if(map_x<(512-240))
                M_BG1SCRLX_SET(map_x++)
            }
            newframe=0;
        } // End of if(newframe)
    } // End of while(1)

    return 0;
} // End of main()

//**********************
// Function: vbl_func()
// Purpose: VBL function
//**********************
void vbl_func()
{
    newframe=1; // Starting a new frame

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

Code Explanation

// Global Variables
newframe is used in *main()* in the *while* loop to determine if it's a new frame or not. 'When does a new frame occur?' you may be thinking. Well, every vertical blank (VBL), of course!
map_x and *map_y* should hopefully be clear to you. They keep track of the top-left corner of the part of the map that will be displayed on the screen.

Now on to *main()*

// Set the text color
ham_SetTextCol(195, 40);
This should be a pretty simple function to figure out. Basically you pass it the index number from the background palette of the color you want for the

foreground and background font color. 'How do I figure out the index number?' you ask. Well, take a look at the Graphics FAQ. If you look at the palette used with *simple_world.bmp*, you'll notice that 195 is red and 40 is a purplish color.

// Setup the map for our image
ham_bg[1].mi = ham_InitMapEmptySet(3,0);
mymap = ham_InitMapFragment(&simple_world_Map,64,64,0,0,64,64,0);
You hopefully noticed that I am passing 64,64 instead of 30,20. Why am I doing this? Well, I am setting up a map which is 512x512 pixels instead of a 240x160 background.

Now let's examine *while(1)*

if(newframe)...
This will be set to 1, or true, every VBL.

if(F_CTRLINPUT_DOWN_PRESSED)... if(map_y<(512-160))
This will allow the map to move down if it's not already at the bottom.
Not sure why we use 512 - 160? Well, there the screen height is 160 pixels. The image height is 512 pixels. The top line of the image will be at 512 - 160 pixels when the map is all the way at the bottom.
Still not sure? Click here.

M_BG1SCRLY_SET(map_y++)
This is just a macro which is used to scroll a background on the Y-axis (up and down axis). *map_y* is incremented 1 because down was pressed. I think the rest of these checks and the other background scrolling macros should be understandable.

There is one more thing you need to know about here - tiles. There is a limit to the number of tiles you can load at once - 651 (I believe). Because my image is 512x512 pixels and because I am loading the whole image at once, this becomes something I have to consider. When you run *gfx2gba* you'll notice that before optimization there are 4096 tiles and after optimization there are only 431. If my image were only a bit more complicated, then this would be a problem and I would have to load the image differently. There are other ways to do this kind of scrolling, but this is one of the easiest ways to understand for now.

That's pretty much it for Day 9. I hope this isn't too complicated to understand.

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: Day9_Maps.zip

View Demo Now

Discuss Day 9

<< HOME >>