HAM Tutorial :: Day 7 :: Project 1 - Tetris

Well, we've finally gotten to the end of the first 'week.' I think it's about time to put together the various things we've learned into a project. What we'll do now is work on a Tetris-style game.

NOTE: We are not going to actually finish the game now (although you can feel free to try that if you really want). I may come back to this for a later project.

I hope you all know about function prototypes.

For this tutorial I've created one sprite image and two background images. Copy these files to the **gfx** directory. This is going to be a two step process. First, go to that directory and type:
gfx2gba -t8 -m -pmaster.pal -fsrc *.bmp

This will create the following files:
master.pal.c
block_L.map.c
block_L.raw.c
bubbles.map.c
bubbles.raw.c
outline_plain.map.c
outline_plain.raw.c

Now we have the necessary background image files and one palette for all of our graphics. The problem is, however, that we don't want the sprite to be in a tile/map format, so we'll type the following command:
gfx2gba -D -fsrc -pblock_L.pal -t8 block_L.bmp

This will create the proper **blocks_L.raw.c**

NOTE: We will not need **block_L.map.c** or **block_L.pal** so you should delete those now.

There is one thing you'll need to know before we start that I didn't explain fully earlier - transparency (or alpha blending). The first color in the palette array is the alpha color. What this means is that any part of your image which has that color is transparent. The system palette in Windows has black in the first spot. Therefore, any part of my image which is black will not be drawn to the screen. You can tell this if you look at outline_plain.bmp and compare it to how it looks in the 'game.'

Well, here we go. Pay attention now, this may seem a little tricky!

Poll

Discussions

Graphics FAQ

GFX2GBA Readme

Translations

Downloads

Games

Projects

Credits

Links

Support

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

// Graphics Includes
// gfx2gba -t8 -m -pmaster.pal -fsrc *.bmp
#include "gfx/master.pal.c"
#include "gfx/bubbles.raw.c"
#include "gfx/bubbles.map.c"
#include "gfx/outline_plain.raw.c"
#include "gfx/outline_plain.map.c"
// gfx2gba -D -fsrc -pblock_L.pal -t8 block_L.bmp
#include "gfx/block_L.raw.c"

// Global Variables
u8 block;         // Sprite object number of the block
u8 block_x = 61;  // X position of block (column)
u8 block_y = 0;   // Y position of block (row)
u8 vbl_count = 0; // Keeps track of the number of VBLs

// Function Prototypes
void vbl_func();      // VBL function
void query_buttons(); // Query for input
void move_block();    // Drop the block
void update_block();  // Apply block's new position

// Function: main()
int main()
{
    // Variables
    map_fragment_info_ptr bg_bubbles;
    map_fragment_info_ptr bg_outline_plain;

    // Initialize HAMlib
    ham_Init();

    // Initialize the text display system
    ham_InitText(2);

    // Setup the background mode
    ham_SetBgMode(1);

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

    // Setup the tileset for our image
    ham_bg[0].ti = ham_InitTileSet(
            (void*)bubbles_Tiles,
            SIZEOF_16BIT(bubbles_Tiles),1,1);
    ham_bg[1].ti = ham_InitTileSet(
            (void*)outline_plain_Tiles,
            SIZEOF_16BIT(outline_plain_Tiles),1,1);

    // Setup the map for our image
```

```
    ham_bg[0].mi = ham_InitMapEmptySet(3,0);
    ham_bg[1].mi = ham_InitMapEmptySet(3,0);

    bg_bubbles = ham_InitMapFragment(
        (void*)bubbles_Map,30,20,0,0,30,20,0);
    bg_outline_plain = ham_InitMapFragment(
        (void*)outline_plain_Map,30,20,0,0,30,20,0);

    ham_InsertMapFragment(bg_bubbles,0,0,0);
    ham_InsertMapFragment(bg_outline_plain,1,0,0);

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

    // Draw some text
    ham_DrawText(21,8,"Next Up");

    // Setup the block
    block = ham_CreateObj((void*)block_L_Bitmap,
        0,2,OBJ_MODE_NORMAL,1,0,0,0,0,0,0,
        block_x,block_y);

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

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

    return 0;
} // End of main()

////********************
// Function: vbl_func()
// Purpose: VBL function
////********************
void vbl_func()
{
    // Increment VBL counter
    ++vbl_count;

    // Copy block sprite to hardware
    ham_CopyObjToOAM();

    // Process the following every VBL
    query_buttons(); // Query for input
    move_block(); // Drop the block
    update_block(); // Apply block's new position

    return;
} // End of vbl_func()

//*************************
// Function: query_buttons()
```

```
// Purpose: Query for input
//*************************
void query_buttons()
{
    if(F_CTRLINPUT_UP_PRESSED)
    {
        if (block_y > 0) block_y = 0;
        ham_DrawText(1,18,"Up   ");
    }

    if(F_CTRLINPUT_DOWN_PRESSED)
    {
        if (block_y < 117) block_y++;
        ham_DrawText(1,18,"Down ");
    }

    if(F_CTRLINPUT_LEFT_PRESSED)
    {
        if (block_x > 61) block_x--;
        ham_DrawText(1,18,"Left ");
    }

    if(F_CTRLINPUT_RIGHT_PRESSED)
    {
        if (block_x < 120) block_x++;
        ham_DrawText(1,18,"Right");
    }

    return;
} // End of query_buttons()

//*******************************************
// Function: move_block()
// Purpose: Drop the block 1 pixel every 5 VBLs
//*******************************************
void move_block()
{
    // Check if 5 VBLs have passed
    if (vbl_count == 4) {
        if (block_y < 117) block_y++; // Drop
        vbl_count = 0; // Reset the VBL counter
    }

    return;
} // End of move_block()

//*************************************
// Function: update_block()
// Purpose: Apply the block's new position
//*************************************
void update_block()
{
    ham_SetObjX(block,block_x);
    ham_SetObjY(block,block_y);
```

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

Code Explanation

// Global Variables
I think these should all be self explanatory.

// Function Protoypes
I will explain each function after I explain what's in main()

map_fragment_info_ptr bg_bubbles;
See Day 6 if you've forgotten what this is.

ham_InitText(2)
For this project we will do our text output on background 2. Backgrounds 0 and 1 will be used for the playing field.

block = ham_CreateObj(...)
If you don't remember what this is, take a look at Day 5.

ham_StartIntHandler(INT_TYPE_VBL,(void)&vbl_func);*
This is a very important function. Basically what it does is tells the CPU in the GBA that every vertical blank it should run the function we specifiy. In this case, we will create a function called vbl_func(). You may ask, what is a vertical blank? A vertical blank occurs every time the GBA finishes writing everything to the screen which is usually 60 times a second.

NOTE: For more info on GBA interrupts, etc, click here.
(*8. Hardware Interrupts*)

So, let's talk about the the other functions.

void vbl_func() {...}
ham_CopyObjToOAM();
Again, you should remember this from Day 5.

Every VBL we need to check for user input, move the block 1 pixel downward, and then update the X & Y position in memory (which gets updated to the screen the next time ham_CopyObjToOAM() is called.

void query_buttons() {...}
All this function does is check if the D-Pad is pressed, outputs the direction that was pressed, and if it's legal, it moves the piece. I don't think there's anything difficult about this, but feel free to tinker with it to see how/why it works. Notice that pressing UP moves the block back to the top.

void move_block() {...}
Notice that *vbl_count* gets incremented at the beginning of vbl_func() or once every VBL. When 5 VBLs have passed (0,1,2,3,4) we moved the piece one pixel downward and reset the VBL counter to zero.

void update_block() {...}
ham_SetObjX(block,block_x);
ham_SetObjY(block,block_y);
These two calls set the new location of the block based on what happened in the query_buttons() function.

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: Day7_Project1_Tetris.zip

View Demo Now

Discuss Day 7

<< HOME >>