CS2110 Fall 2013
Homework 9

This assignment is due by:

Day: Wednesday October 30th

Time: 11:54:59pm

Rules and Regulations

Academic Misconduct

Academic misconduct is taken very seriously in this class. Homework assignments are collaborative. However, each of these assignments should be coded by you and only you. This means you may not copy code from your peers, someone who has already taken this course, or from the Internet. You may work with others **who are enrolled in the course**, but each student should be turning in their own version of the assignment. Be very careful when supplying your work to a classmate that promises just to look at it. If he/she turns it in as his own you will both be charged.

We will be using automated code analysis and comparison tools to enforce these rules. **If you are caught you will receive a zero and will be reported to Dean of Students.**

Submission Guidelines

1. You are responsible for turning in assignments on time. This includes allowing for unforeseen circumstances. If you have an emergency let us know ***IN ADVANCE*** of the due time supplying relevant documentation (i.e. note from the dean, doctor's note, etc). Extensions will only be granted to those who contact us in advance of the deadline and no extensions will be made after the due date.

2. You are also responsible for ensuring that what you turned in is what you meant to turn in. No excuses, what you turn in is what we grade. In addition your assignment must be turned in on T-Square. When you submit the assignment you should get an email from T-Square telling you that you submitted the assignment. If you do not get this email that means that you did not complete the submission process correctly. Under no circumstances whatsoever we will accept any email submission of an assignment. Note: if you were granted an extension you will still turn in the assignment over T-Square.

3. There is a random grace period added to all assignments and the TA who posts the assignment determines it. The grace period will last at least one hour and may be up to 6 hours and can end on a 5 minute interval; therefore, you are guaranteed to be able to submit your assignment before 12:55AM and you may have up to 5:55AM. As stated it can end on a 5 minute interval so valid ending times are 1AM, 1:05AM, 1:10AM, etc. **Do not ask us what the grace period is we will not tell you**. So what you should take from this is not to start assignments on the last day and depend on this grace period past 12:55AM. There is also no late penalty for submitting within the grace period. If you cannot submit your assignment on T-Square due to the grace period ending then you will receive a zero, no exceptions.

General Rules

1. In addition any code you write (if any) must be clearly commented and the comments must be meaningful. You should comment your code in terms of the algorithm you are implementing we all know what the line of code does.
2. Although you may ask TAs for clarification, you are ultimately responsible for what you submit.
3. Please read the assignment in its entirety before asking questions.
4. Please start assignments early, and ask for help early. Do not email us the night the assignment is due with questions.
5. If you find any problems with the assignment it would be greatly appreciated if you reported them to the author (which can be found at the top of the assignment). Announcements will be posted if the assignment changes.

Submission Conventions

1. All files you submit for assignments in this course should have your name at the top of the file as a comment for any source code file, and somewhere in the file, near the top, for other files.
2. In addition any code you write must be clearly commented and the comments must be meaningful. You should comment your code in terms of the algorithm you are implementing we all know what the line of code does.
3. When preparing your submission you may either submit the files individually to T-Square (preferred) or you may submit an archive (zip or tar.gz only please) of the files.
4. If you choose to submit an archive please don't zip up a folder with the files, only submit an archive of the files we want.
5. Do not submit compiled files, that is .class files for Java code and .o files for C code.

Overview

The goal of this assignment is to make a C program, a game. This is to get your hands wet with the C programming language. Your game should include everything in the requirements and be written neatly and efficiently. This is your chance to wow and impress us, so be creative!

Functionality Requirements

Each of these requirements must be implemented within your game. Therefore, your game must have:

1. *Winning and losing conditions*. A player must be able to win and must be able to lose. The goal of your game should be easily discernible.
2. *Lives*. Your game must have an indication of progress (good or bad) to the player. This field does not need to be text, but it does need to be visible.
3. *Button input*. When buttons are pressed, the flow of the game should change. You can use buttons to move characters, create events, speed up time, etc.
4. *2-Dimensional movement*. At least one element of the game should move in a two-dimensional way.
5. *Collision detection*. Your game must determine if an object collides with another object. From here, some action may occur.
6. *Reset functionality*. You must be able to reset your game at any time using the SELECT button.
7. *Images.* You must have a start screen to introduce us to your game. A game over screen must also appear when the game ends. You must also have at least one other image which is used during play that is not 240x160. All images should be included with your assignment submission.

Program Requirements

All games must also satisfy these requirements

1. *Mode 3.* Your game must use Mode 3. If you know other modes and want to use them or want to use a different mode, then you must email a TA in advance of the due date.
2. *DMA*. You must implement drawImage3 with DMA and be prepared to explain DMA during your demo. The function prototype is explained later in the assignment.
3. *C coding conventions*
    a. Do not jam all of your code into one function (i.e. the main function)
    b. Split your code into multiple files (have all of your game logic in your main file, library functions in mylib.c, breakout specific code in breakout.c)
    c. Use a header file (mylib.h) for any macros, function prototypes, and typedefs you have created
    d. Comment your code, comment what each function does. The better your code is the better your grade!
4. Your game must be efficient, meaning we should not see any tearing. Tearing occurs when the amount of time it takes to draw and update your objects is longer than the vertical blank period
5. Do not use floats or doubles in your code. Doing so will SLOW your code down GREATLY. The ARM7 processor the GBA uses does not have a Floating Point Unit which means floating point operations are SLOW. If you do need such things then you should look into fixed point math.
6. Only call waitForVBlank once per iteration of your while/game loop

If you want to write a game that doesn't fit the criteria for this assignment but is TOTALLY AWESOME, email a TA.

Game ideas: *World's Hardest Game, Asteroid, Galaga*

Images

As a requirement you must use at least 3 images in your game and you must draw them all using drawImage3. We have provided a tool that will make this task easier for you

BrandonTools – Source code available at https://github.com/TricksterGuy/brandontools

Should be obvious who this is written by. BrandonTools (for lack of better name) handles pretty much any multimedia format (including pdf, avi, ppt) and exports the file into a format the gba can read (in the case of such files you will get an array for each page/frame/slide). Also supports manipulating the images before they are exported.

Before using the program make sure you read the readme which explains program usage in detail.

Basic usage for mode 3 is `brandontools -mode3 filename imagefilename`

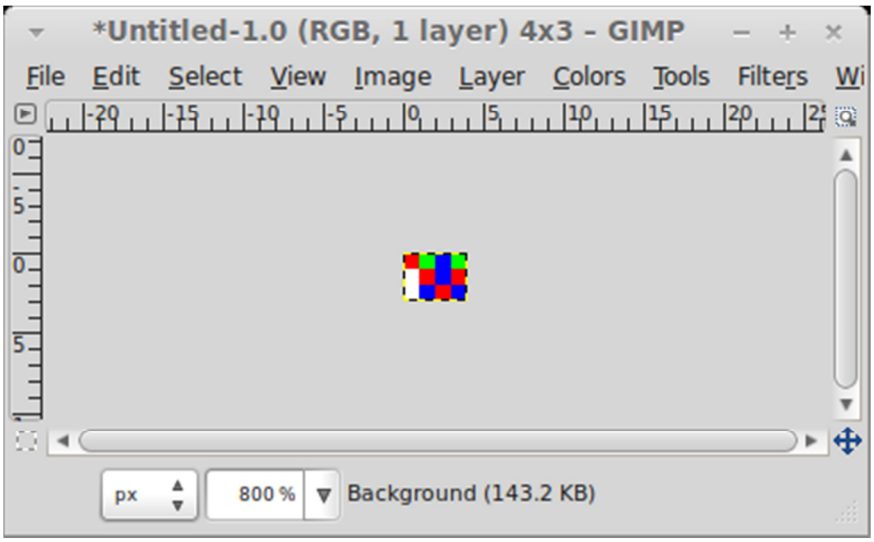
Example: `brandontools -mode3 myimages kirby.png`

The output of this program will be a .c file and a .h file. In your game you will #include the header file. In the header file it contains an extern statement so any file that includes it will be able to see the declarations given in the .c file brandontools exported.

Inside the exported .c file is an 1D array of colors which you can use to draw to the screen.

Example

For instance take this 4x3 image courtesy of GIMP



When this file is exported here is what the array will look like

```
const unsigned short example[12] =
{
	// first row red, green, blue, green
	0x001f, 0x03e0, 0x7c00, 0x03e0,
	// white, red, blue, red
	0x7fff,0x001f, 0x7c00, 0x001f,
	//white, blue, red, blue
	0x7fff, 0x7c00, 0x001f, 0x7c00,
}
```

The number of entries in this 1D array is 12 which is 4 times 3. Each row from the image is stored right after the other. So if you wanted to access coordinate (row = 1, col = 3) then you should get the value at index 7 from this array which is red.

For image files I don't care where you get them or you can make them yourself. Here is a link with some image files

http://www.gamedev.net/community/forums/topic.asp?topic_id=272386

http://gamemaking.indiangames.net/index_files/FreeSpritesforGames.htm

DMA / drawimage3

In your game you must use DMA to code drawImage3.

DMA stands for Direct Memory Access and may be used to make your rendering code run much faster. If you want to read up on DMA before Bill goes over this in class you may read these pages from tonc.

http://www.coranac.com/tonc/text/dma.htm (Up until 14.3.2).

If you want to wait then you can choose to implement drawImage3 without DMA and then when you learn DMA rewrite it using DMA. However your final answer for drawImage3 must use DMA.

As for restrictions on DMA. You must not use DMA to do one pixel copies (Doing this defeats the purpose of DMA and is slower than just using setPixel!).

Solutions that do this will recieve no credit for that function

The prototype and parameters for drawImage3 are as follows.

```
/* drawimage3
 * A function that will draw an arbitrary sized image
 * onto the screen (with DMA).
 * @param r row to draw the image
 * @param c column to draw the image
 * @param width width of the image
 * @param height height of the image
 * @param image Pointer to the first element of the image.
 */
void drawImage3(int r, int c, int width, int height, const u16* image)
{
    // @todo implement :)
}
```

Protip if your implementation of this function does not use all of the parameters that are passed in then **YOU'RE DOING IT WRONG.**

Here is a hint for this function. You should know that DMA acts as a for loop, but it is done in hardware. You should draw each row of the image and let DMA handle drawing a row of the image.

Lastly note that we don't care if the background of your image is being drawn. If you want a transparent background for your image then you will need to resort to using sprites. Of course, this is way beyond the Call of Duty for this assignment so if you get this working then more power to you

http://www.coranac.com/tonc/text/regobj.htm

Debugging

The cs2110-tools-debug package allows you to print things to the terminal from within a gba game. (Remember that the GBA does not have a terminal to print things to).

We have enabled this functionality, by modifying the emulators source code. To use this functionality here are the steps.

1. #include <debugging.h> at the top of any file you need to print something.
2. Use the Macro DEBUG_PRINTF(format_string, varags) to print using a format string
3. Use the Macro DEBUG_PRINT(string) to just print a string out.
4. Examples.
   - `DEBUG_PRINTF("My age is %d\n", 25);`
   - `DEBUG_PRINT("HELLO WORLD\n");`
5. To compile your code with debugging support you must type the following in the terminal
   - `make clean`
   - `make debugvba (or make debugwxvba)`
6. [IMPORTANT] When you are done debugging type the following command. into a terminal
   - `make clean && make gvba (or make wxvba)`
7. Don't forget to do the previous step if you do then you will still be running the debug version. Likewise if you fail to type the command in step 5 you will be still running the non-debug version.

DO NOT CALL PRINTF DIRECTLY FROM GBA CODE. USE THE MACROS DEBUG_PRINTF AND DEBUG_PRINT.

REMEMBER THAT WHENEVER YOU USE THESE MACROS TO INCLUDE A \n WITHIN THE STRING YOU PRINT OUT. IF YOU DO NOT DO THIS THEN THE OS WILL BUFFER IT.

GBA Key Map

In case you are unaware, the GameBoy Advance buttons correspond to the following keys on the keyboard.

```
GameBoy | Keyboard
--------|----------
Start   | Enter
Select  | Backspace
A       | Z
B       | X
L       | A
R       | S
Left    | Left Arrow
Right   | Right Arrow
Up      | Up Arrow
Down    | Down Arrow
```

Additionally, holding the space bar will make the game run faster. This might be useful in testing, however the player should never have to hold down spacebar for the game to run properly and furthermore there is no space bar on the actual gba.

Collaboration Reminder

Remember that you are more than welcome to work with other students as you have been on past assignments, however, you are not allowed to copy code among each other. Please keep discussion to high level details. All code you write must be coded by you. You are free to help other students with problems with their code; however you aren't allowed to take someone elses code (via the internet, previous semester, current semester) and submit it as your own.

We will be employing use of code analysis tools to catch people who violate these rules and if you are caught you will receive a zero.

See Academic Misconduct in the rules section at the top of this assignment.

This assignment and will be demoed you will be responsible for all code you write and you should know what it does. We will let you know when these demo times are up via an announcement.

Deliverables

All files needed to compile and link your program. Plus any image files you have used to export to gba.

Or an archive containing ONLY these files and not a folder that contains these files. In addition DO NOT submit .o files these are the compiled versions of your .c files