CS 2110 Fall 2014
Homework 11

This assignment is due by:

Day: Tuesday, November 11th

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 who promises just to look at it. If he 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 the 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 will we 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 can not submit your assignment on T-Square due to the grace period ending then you will receive a zero, no exceptions.
4. Although you may ask TAs for clarification but you are ultimately responsible for what you submit.

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; however, this time you will have to do it in Mode 4! Your game should include everything in the requirements and be written neatly and efficiently. **Your game must be different from the games you made in previous homework assignments**. This is your chance to wow and impress us, so be creative!

Before you start

You should review Mode 4. Remember the following concepts and please read and reread these before asking questions.

1. You now have two displays, one of which you draw on while the other is displayed to the user. You will be using the technique of page flipping to swap between these pages. Remember what you draw is not shown to the user until you make a call to flipPage.
2. You don't have any additional memory allocated to you to have 2 videoBuffers. The memory is chopped in half which means pixels are represented by u8s instead of u16s. These u8s are NOT colors, but an index into the palette. You just write this index into your videoBuffer and the hardware takes care of the rest.
3. The palette is an array of colors located at memory address 0x5000000, with palette entry 0 being the background color. Again pixels are represented as an index into this array.
4. There is only one palette that must be shared among everything in your game. You may not draw an image and then change the palette and then draw the second image. The colors for the first image will change to whatever is in the palette for the second image. You will need a palette that works for all of the things you are trying to display.
5. As an example of the palette concept, consider this array: u16 palette[4] = {0x000, 0x001f, 0x3e0, 0x7c00}. If you want to refer to blue here instead of saying 0x7c00 you would say 3 because that color is at array index 3.

Here are some additional references. It is strongly suggested you read this once or twice before starting.
http://www.coranac.com/tonc/text/bitmaps.htm A dash of mode4
http://www.coranac.com/tonc/text/bitmaps.htm#sec-page 5.3 and 5.3.1

Requirements

Your game must satisfy these requirements:

1. **Must be in Mode 4**. Absolutely NO CREDIT will be given for Mode 3.
2. **Title screen and Game Over screen**
    - Both should be images sized 240x160
3. **Must have an animation of at least 4 frames**. For example, in Frogger, the froggy must have a jumping animation (at least 4 frames of animation) that plays when you move him. Or, in Minesweeper, an explosion animation could be displayed when the player hits a mine. You are NOT required to make either of those 2 games, they are just examples.
4. **You should have at least 2 images, other than the title and game over screens, whose width is not 240 and height is not 160 that will be used during gameplay**.
    - Your images should optimally mesh in with the gameplay
    - One of these images will be the image to satisfy the animation requirement. You must have another image other than the images you use for the animation.
    - You may go above and beyond and have more images than this, I don't mind.
    - You are allowed to combine your images into one spritesheet as long as they are still used as separate images in the game.
5. **You must code the function drawImage4 with DMA and use it to draw you images onto the screen**. A prototype is provided below:

```
/* drawImage4
 * 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 drawImage4(int r, int c, int width, int height, const
u16* image) {
    // TODO implement
}
```

6. **You must be able to reset the game (to the title screen) AT ANY TIME by pressing the SELECT button**.
7. Your game must be efficient, meaning **we should not see any tearing**.
8. **Button input**
9. **waitForVBlank**
10. **PageFlip**
    - Page flip is the function responsible for swapping the video buffers and flipping the page by setting the appropriate bit in the REG_DISPCNT
11. The use of **at least on structure** in C (DMA does not count).
12. **Winning and losing conditions**. A player must be able to in and must be able to lose. The object (or goal) of your game should be easily discernible.

13. **Lives [or some indication of progress (good or bad) to the player]**. This field does not need to be text, but it does need to be visible to the player.
14. **Collision detection**. Your game must determine if an object collides with another object. From here, some action may occur.

This is your chance to show us what you can do. So have fun! **If you want to write a game that doesn't fit the criteria for this assignment but is TOTALLY AWESOME, email a TA.**

Images

As a requirement you must use at least 4 images (start screen, game over screen, animated image that isn't full-screen, and at least one other), and you must draw them all using drawImage4.

You should not be using an image whose width is not divisible by 2. Remember that you have 2 pixels per address and if you have an odd width it won't work.

In addition you will need to get a global palette that works with all images you export, minus the title and game over screens. You may export those images separately since you will only be displaying them by themselves, so you can use their palettes, then swap them out when you are done displaying the image.

To export images in mode 4, you should use JGrit, which you can download at https://code.google.com/p/jgrit/. If all your images are in separate files you should modify the options for "First entry" and "Number of entries" for the palette (in the Advanced options), so that the global palette can contain all the colors from all the images without overlapping. Alternatively, you can combine all of your images into one spritesheet, which is exported with JGrit, then just draw the section of the spritesheet for you image.

Additional Notes

If you are running a blank on game ideas, here are a few to think about:

Frogger, Minesweeper

GBA Key Map

In case you are unaware, the GBA buttons correspond to the following keyboard keys:

```
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.

C Coding Conventions

1. Do not jam all of your code into one function (i.e. the main function)
2. Split your code into multiple files (have all of you game logic in your main file, library functions in mylib.c, etc.)
3. Comment your code, comment what each function does. The better your code is the better your grade!

Warnings

1. Do not use floats or doubles in your code. Doing so will SLOW you code down GREATLY. The ARM7 processor that 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 (google search).
2. Only call waitForVBlank once per iteration of your while/game loop
3. Keep your code efficient. If an O(1) solution exists to an algorithm and you are using an O(n^2) algorithm then that's bad.

Collaboration Reminder

Remember that you are more than welcome to work with other students are you have been on past assignments, however, you are not allowed to copy code amongst 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 else's code 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 will be demoed and you will be responsible for all code you write and you should know what it does. We will let you know when the 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. So make sure you run *make clean* before turning it in.