CS 2110 Fall 2015
Homework 10

This assignment is due by:

Day: Friday, November 6th

Time: 11:54:59pm

Rules and Regulations

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 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. After submitting you should be sure to download your submission into a brand new folder and test if it works. No excuses if you submit the wrong files, what you turn in is what we grade. In addition, your assignment must be turned in via 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 6-hour grace period added to all assignments. You may submit your assignment without penalty up until 11:55PM, or with 25% penalty up until 5:55AM. *So what you should take from this is not to start assignments on the last day and plan to submit right at 11:54PM.* You alone are responsible for submitting your homework before the grace period begins or ends; neither T-Square, nor your flaky internet are to blame if you are unable to submit because you banked on your computer working up until 11:54PM. The penalty for submitting during the grace period (25%) or after (no credit) is non-negotiable

General Rules

1. Starting with the assembly homeworks, 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. This means that (in the case of demos) you should come prepared to explain to the TA how any piece of code you submitted works, even if you copied it from the book or read about it on the internet.
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 unless otherwise noted.
2. When preparing your submission you may either submit the files individually to T-Square or you may submit an archive (zip or tar.gz only please) of the files (preferred). You can create an archive by right clicking on files and selecting the appropriate compress option on your system.
3. 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. (See **Deliverables**).
4. Do not submit compiled files that is .class files for Java code and .o files for C code. Only submit the files we ask for in the assignment.
5. Do not submit links to files. We will not grade assignments submitted this way as it is easy to change the files after the submission period ends.

Syllabus Excerpt on Academic Misconduct

Academic misconduct is taken very seriously in this class.

Quizzes, timed labs and the final examination are individual work.

Homework assignments are collaborative, In addition many if not all homework assignments will be evaluated via demo or code review. During this evaluation, you will be expected to be able to explain every aspect of your submission. Homework assignments will also be examined using electronic computer programs to find evidence of unauthorized collaboration.

What is unauthorized collaboration? Each individual programming assignment should be coded by you. You may work with others, but each student should be turning in their own version of the assignment. Submissions that are essentially identical will receive a zero and will be sent to the Dean of Students' Office of Academic Integrity. Submissions that are copies that have been superficially modified to conceal that they are copies are also considered unauthorized collaboration.

You are expressly forbidden to supply a copy of your homework to another student via electronic means. If you supply an electronic copy of your homework to another student and they are charged with copying you will also be charged. This includes storing your code on any site which would allow other parties to obtain your code such as but not limited to public repositories, etc.

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 game you made in your previous homework assignment, but you may use your old library files**. 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 **`u8`**s instead of **`u16`**s. These **`u8`**s 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* 1 more image, other than the title/game over screens and the animation, whose width is not 240 and height is not 160 that will be used during gameplay**.
    - Your image should optimally mesh in with the gameplay.
        i. It could be used as scenery, a path, an enemy, etc.
    - This image may be a part of another animation, if you wish, but it must be separate from the first 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 use **waitForVBlank** and be efficient; **we should not see any tearing**.
8. **Button input**
9. **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
10. The use of **at least one `struct`** in C (DMA does not count).
11. **Winning and losing conditions**. A player must be able to win and must be able to lose. The object (or goal) of your game should be easily discernible.

12. **Use of text** to write words to the screen, as described in the class.
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.
15. You will maintain mostly the same project structure as for your previous homeworks, but will create a new **myLib.h** file that will contain the declarations for things defined in myLib.c. You will then include this file where needed instead of copying the declarations themselves into each file. Feel free, and it is good practice, to put related code into their own .c/.h files. Remember that function and variable definitions should not go in header files, just prototypes and extern variable declarations. There are examples of separation between header and c files in the resources. Make sure you add your header files in your Makefile.

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 (or any other tool, if you wish), 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.

Game Choices

You may implement any game you wish, but it should be similar in scope to the games from the previous homework. You do not have to implement multiple levels. If you are unsure about game ideas, consider the list from the previous homework (**this game has to be different from the one you made in the previous homework!)** Here are a few more game ideas to think about:

Frogger, Minesweeper, Breakout, Galaga, old-school RPGs like Final Fantasy, adventure, side scrollers, Skyrim, Witcher 3.

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.

Helpful Tips

There are alternatives for processing images for mode 4. You could try the cs2110imagetools and nin10kit courtesy of Brandon at: https://github.com/TricksterGuy/nin10kit. Incase performance is bad for your vba try running it on mednafen by using the command: "make med". If you do not have this software then: sudo apt-get install mednafen. Make sure that place all of your header and c files in the Makefile.

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 (all your .c and .h files, and your Makefile), 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.