CS2110 Spring 2014
Homework 9

This assignment is due by:

Day: Wednesday, April 16nd
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 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 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. Your game should include everything in the requirements and be written neatly and efficiently! There are no specifics for how you choose to implement these requirements, though there are also some technical requirements that follow. This is your chance to wow and impress us, so be creative!

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 object (or 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 to the player. If you wish to use text within your game, read through Chapter 19 of tonc and use the files from lecture..
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 your game should be able to 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. ***Start screen and game over screen.*** You must have a start screen to introduce us to your game. A game over screen must also appear when the game ends. Feel free to add other images throughout your game, you are not limited to only two. These screens must be drawn with DMA.
8. ***Text.*** Your game must use text in some form. To write text, you should use the files that were provided in lecture.
9. ***Smoothness and no tearing.*** Your game must play smoothly and contain no significant tearing. You should use waitForVBlank and DMA to make your game run as smoothly as possible.

These are the only game requirements that need to be met for this assignment, though you must still implement DMA and image loading (more on this below). We **highly** encourage you to go out of your way and make something amazing! More sophisticated and awesome games will more likely receive higher scores. This will also be the last GBA homework of the semester. 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 your TA.

Additional Notes

If you are running a blank on game ideas, here are a few to help stimulate your brain muscles:

World's Hardest Game, Lunar Landing, Asteroid, Final Fantasy, Skyrim

You are REQUIRED to put images into your game, meaning that the use of an image package is required for this assignment. There are several different tools that you can use to achieve this. Past semesters used BrandonTools (https://github.com/TricksterGuy/brandontools), which is able to support a variety of formats and modes, but can sometimes be tricky to install. This semester we are encouraging the use of CS2110ImageTools.jar, which can be found on T-Square and is easy to use, but which does not support as many features. Yet another option is to use jGrit, which is both powerful and flexible. Whichever program you choose, you must draw your images with DMA. Information on BrandonTools (appliccable to other software packages as well) can be found later in this document.

C coding conventions

1. Do not jam all of your code into one function (i.e. the main function) – you will lose points if you do so.
2. Split your code into multiple files (have all of your game logic in your main file, library functions in mylib.c, and other game-specific code in their own files. For example, you could have breakout specific code in breakout.c)
3. Comment your code, comment what each function does. The better your code is the better your grade!

GBA Key Map

As a reminder, 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.

Warnings

1. 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 (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 thats bad (for larger values of n)! Contrary to this only worry about efficiency if your game is showing signs of tearing!

Images

If you decide to use BrandonTools (https://github.com/TricksterGuy/brandontools), here is an outline of how it works. Most of this information is still applicable to other applications.

Should be obvious who this is written by. BrandonTools (for the lack of a 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.

There is a readme file in the repository that outlines the installation process and how it can be used to convert images.

Basic usage for mode 3 is:

```
brandontools -mode3 filename imagefilename
```

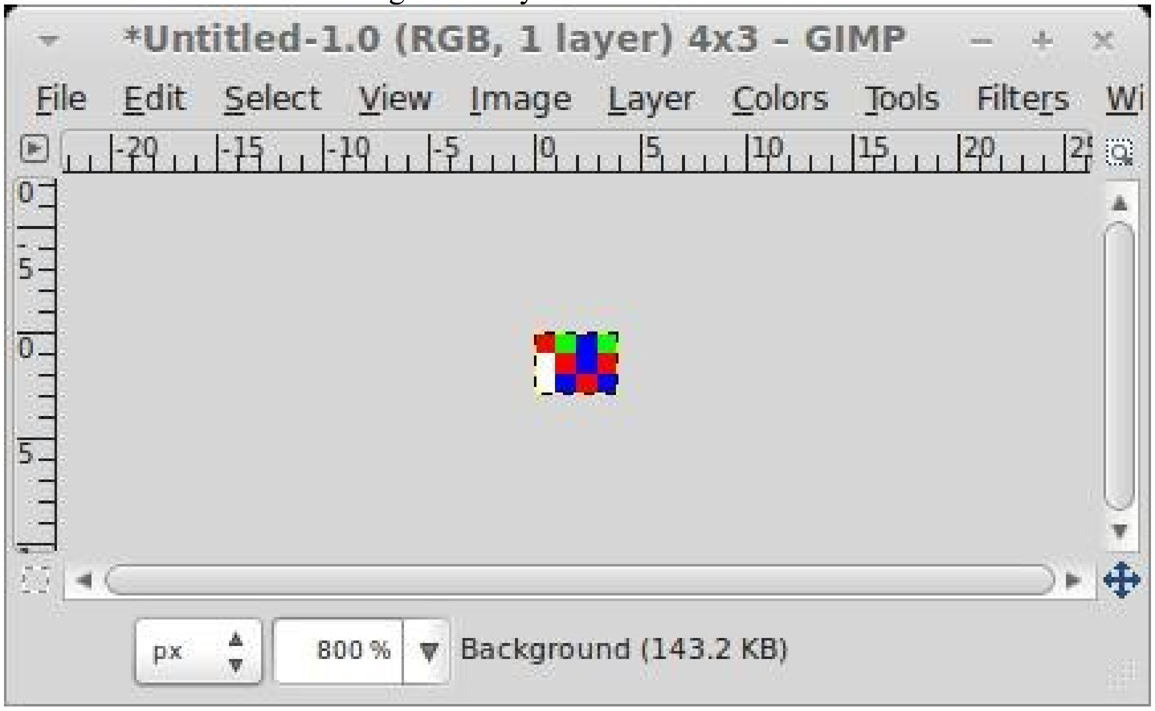
example:

```
 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


*Untitled-1.0 (RGB, 1 layer) 4x3 - GIMP
File Edit Select View Image Layer Colors Tools Filters Wi
px
800 %
Background (143.2 KB)

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, you can find sprites online or make them yourself. Here are a few links 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 the function drawImage3.

As you have learned in lecture, DMA stands for Direct Memory Access and may be used to make your rendering code run much faster.

You can read more about it in the online GBA resource book that we mentioned before - tonc.

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

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 receive 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

Readme.txt

You must include a readme.txt file with your submission. This file needs to outline the controls of your game. It also needs to explain what the goal of the game is, and how it can be reached. Let us know how a player can win or lose the game.

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 else's 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 will be due at the date specified. It will not be demoed.

Deliverables

Make sure to submit the following:

1) .zip ALL files required to compile and run your game with the command "make vba"

> Be sure to include the Makefile as well. Download your submitted code to check that it compiles. If your code does not compile, we will not be able to grade it, and you will receive a **zero**. Please double check your work.

2) Be sure to include the readme.txt file. It should outline what buttons do what, and what the goal of the game is. State clearly how one can win and lose the game.