CS2110 Spring 2015
Homework 09

This assignment is due by:

Day: Wednesday April 1, 2015
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 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. Make sure that you submit and demo your assignment by the last demo date of the class. We will not demo anyone past that date, and all unfinished homework assignments will receive a score of zero.

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 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 in 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.

Objective:

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! Make sure you keep the same set up with your files as you did before in Homework 9. That includes your VideoBuffer, macros, REG_DISPCNT, typedefs, and use of a main.c and myLib.c files as you did before. Your main.c file should be something different, since in this homework you will be creating your own game, but keep the core setup. You can still use the same myLib.c file as you did before, but with declarations split into a **myLib.h**, and we encourage you to add even more functions to the file. It is also optional for you to use other .c/.h files to organize your game logic if you wish, **just make sure you include them in submission and makefile**. Additionally, we want to make one point very clear: **please do not rehash lecture code in your game**. This means that you are not allowed to just slightly modify lecture code and to call it a day. If we open your game and we see several boxes flying in random directions, that will be a very bad sign, and you will not receive a very pleasant grade. Also, please do not make Pong.

Requirements:

1. Must be in Mode 3
2. Images – You must use 3 distinct images in your game, all drawn with DMA
   a. A title screen sized 240x160
   b. A game over screen sized 240x160
   c. A third image which will be used during game play. The width of this image may not be 240 and the height of this image may not be 160.
   d. Note: all images should be included in your submission
3. You must also implement **drawImage3** with **DMA**. The prototype and explanation are later in the assignment
4. You must be able to reset the game to the title screen AT ANY TIME using the select key
5. You must create a header (**myLib.h**), and move any #defines, function prototypes, and typedefs to this file from myLib.c, along with your extern videoBuffer statement. Remember that function and variable definitions should not go in header files, just prototypes and extern variable declarations.
6. You must use at least one **struct**.
7. **Button input** should visibly and clearly affect the flow of the game
8. You must have **2-dimensional movement** of at least one entity
9. You should implement some form of object collision.
10. You must implement **waitforVBlank** and the scanlinecounter declaration.
11. Use text to show progression in your game. Use the example files from lecture, and you can look into it more in Tonc: http://www.coranac.com/tonc/text/text.htm
12. **There must be no tearing in your game. Make your code as efficient as possible!**

Include a **readme.txt** file with your submission that briefly explains the game and the controls.

What Game to Make?

You may either create your own game the way you wish it to be as long as it covers the requirements, or you can make games that have been made before with you own code. **However, your assignment must be yours entirely and not based on anyone else's code. This also means that you are not allowed to base your game off the code posted from lecture. Games that are Bill's code that have been slightly modified are subject to heavy penalties.** Here are some previous games that you can either create or use as inspiration:

Galaga: http://en.wikipedia.org/wiki/Galaga
Requirements:
1. Accurate, Efficient O(1) Rectangular Collision Detection implemented as a function.
2. Lives, you can use text to show them.
3. Game ends when all lives are lost. Level ends when all aliens are gone.
4. Different types of aliens – there should be one type of alien that rushes towards the ship and attacks it
5. Smooth movement (aliens and player)

The World's Hardest Game (Challenge our skill with your game):
http://www.addictinggames.com/action-games/theworldshardestgame.jsp
Requirements:
1. Accurate, Efficient O(1) Rectangle Collision detection as a function.
2. Smooth motion for enemies and player (no jumping around)
3. Constriction to the boundaries of the level.
4. Enemies moving at different speeds and in different directions.
6. Sensible, repeating patterns of enemy motion
7. Enemies and the Player represented by Structs

Frogger: http://en.wikipedia.org/wiki/Frogger
Requirements:
1. The Frog and the logs/lily pads must be represented by Structs internally.
2. O(1) collision detection with any object. If the frog collides with traffic, it dies. If it does not land on a lilypad/log, then it also dies. The materials in the river "lily pads/logs" must move the froggy along with them.
3. There is a time limit in which the frog must get to his home if time expires then the frog also dies (hint there are about 60 vblanks per second.)
4. Once a Froggy occupies a home, another frog cannot occupy that home.
5. The player must have a set amount of lives they can lose before losing. The number of lives must be displayed to the user. The game is over when all of the lives are lost. The game is won if all frogs get to their homes.

These are just some suggestions to get you on the right track with making your game. If you are having any trouble deciding or an idea you would like to check, please consult with the TAs.

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

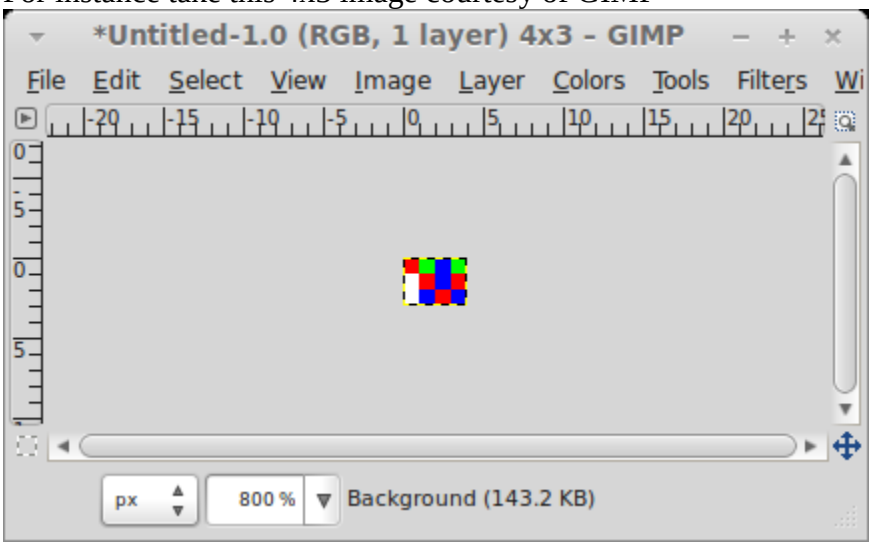
CS2110ImageTools.jar – available on T-Square in Resources/GBA

CS2110ImageTools.jar reads in, converts, and exports an image file into a format the gba can read – .c/.h files! It also supports resizing the images before they are exported.

You may run the program from terminal in the directory where you downloaded it like this:
`java -jar CS2110ImageTools.jar`

CS2110ImageTools.jar will give you a graphical interface to load image files and save their converted files. The output of this program will be a .c file and a .h file. In your game you will #include 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 CS2110ImageTools.jar exported. Inside the exported .c file is a 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.

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.

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.

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 your game logic in your main file, library functions in mylib.c with declarations in mylib.h, game specific functions in game.c)
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 your code down GREATLY. The ARM7 processor the GBA uses does not have a Floating Point Unit which means floating point operations are SLOW and are done in software, not hardware. 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 (for larger values of n)! Contrary to this only worry about efficiency if your game is showing signs of tearing!

I **STRONGLY ADVISE** that you go **ABOVE AND BEYOND** on this homework. **PLEASE do not just clutter your screen with squares that run around with no meaning in life** – such submissions will **lose** points. Make some catchy animations, or cut scenes! These games are always fun to show off after the class, you can even download emulators on your phone to play them. Also, **remember that your homework will be partially graded on its creative properties and work ethic**. You are much more likely to make your TAs very happy and to have a better demo/grade if you go above and beyond what is required.

You may research the GBA Hardware on your own. You can read Tonc, however you may not copy large swaths of code, only use it as a reference. The author assumes you are using his gba libraries, which you are not and may not.

Here are the inputs from the GameBoy based on the keyboard for the default emulator vbam:

```
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
```

Holding the space bar will make the emulator run faster. This might be useful in testing, but the player should never have to hold down space bar for the game to run properly and furthermore there is no space bar on the actual GBA.

You can learn more about button inputs on this site: http://www.coranac.com/tonc/text/keys.htm

If you want to add randomness to your game then look up the function rand in the man pages. Type man 3 rand in a terminal.

Deliverables

main.c
myLib.c
myLib.h
Makefile (your modified version)
and any other files that you chose to implement

Or an archive containing ONLY these files and not a folder that contains these files. DO NOT submit .o files as these are the compiled versions of your .c files, you can type ‘make clean’ to remove those files.

Note: make sure your code compiles with the command:

```
make vba
```

Make sure to double check that your program compiles, as **you will receive a zero (0)** if your homework does not compile.

Good luck, and have fun!