CS2110 Fall 2014

Homework 9

This assignment is due by:

Day: Tuesday October 28, 2014
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

Starting the homework:

We do not want you to build on your last homework, but make sure you keep the same set up with your files as you did before in Homework 8. That means keeping the same set up for 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, and we encourage you to add even more functions to the file. It is also optional for you to use other '.c' 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. **Animation**! Make stuff move.
2. Maintain the same structure of you myLib.c and main.c files as before.
3. ~~You should use **Structs** to display at least three different objects on the screen at once.~~ Structs are now optional for you to use. It is recommended that you use them but Bill will not cover them until Tuesday.
4. **Button input** should visibly and clearly affect the flow of the game.
5. You should implement some form of **object collision**, usually to help progress your game. There are many ways you can make some sort of collision detection, feel free to ask the TAs for help with it.
6. **2-Dimensional movement** of at least one entity, be it a player's avatar or an enemy.
7. Try to avoid tearing of the image – implement a function that waits for **vBlank**, as was demonstrated in lecture, to make the game run smoother.
8. Use **text** to show progression in your game. Use the example files that were provided for you during lecture, and you can look into it more in Tonc:
    a. Check Tonc for more help on this: http://www.coranac.com/tonc/text/text.htm
    b. You may also look at the lecture slides
9. Include a **readme.txt** file with your submission that briefly explains the game and the controls.
10. **Creativity and sophistication** – the more time you put into this, the better the grade.

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. If you

choose the second option, you will have to create it with primitive shapes since we have not covered drawing images onto the screen. You can have fun with it and build some familiar shapes from the games using pixel art, but we do not expect you to draw a masterpiece with it. It is best to focus on the core requirements of the games. Here are some previous games that you can either create or use as inspiration:

There are additional requirements to create the game, but most will overlap the given requirements.

Galaga (Draw the ship with pixel art!): 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 it.
4. Game ends when all lives are lost. Level ends when all aliens are gone.
5. Different types of aliens there should be one type of alien that rushes towards the ship and attacks it
6. 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.
5. Sensible, repeating patterns of enemy motion, you should have enemies that move in a pattern.
7. Enemies and the Player represented by Structs

Frogger (You can make a frog with pixel art!): 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.

The Elder Scrolls V: Skyrim (DO IT!):
http://en.wikipedia.org/wiki/The_Elder_Scrolls_V:_Skyrim
Requirements:
1. Build the Creation Engine
2. Build Skyrim

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.

Warning

Two things to note for now

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. Do not do anything very intensive, you can only do so much before the GBA can't update the screen fast enough. You will be fixing this problem in the next assignment.

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 (available at http://www.coranac.com/tonc/text/), however you may not copy code wholesale from this site. The author assumes you are using his gba libraries, which you are not.
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.

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, game specific functions in game.c)
3. Comment your code, comment what each function does. The better your code is the better

your grade!

Here the inputs from the Gameboy based 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

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

Deliverables

main.c
mylib.c
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!