CS 2110: Homework 10
GBA

Summer 2018

Contents

1 Overview

The goal of this assignment is to make a C program that will run on the GBA emulator. Your program should include everything in the requirements and be written neatly and efficiently. Your `main.c` should be something different from lecture code, since in this homework you will be creating your own program. However, you will keep the core setup with `videoBuffer`, MODE 3, `waitForVBlank`, etc.

Prototypes, `#define`'s, and `extern` declarations should be put into a `myLib.h`. You may use other `.c` and `.h` files to organize your logic if you wish. Make sure you include them in both your submission and `Makefile`.

Additionally, we want to make one point very clear: **Please do not rehash lecture code in your program.** This means you are not allowed to slightly modify lecture code and "call it a day." If we open your program and discover several boxes flying in random directions, that will be a very **bad** sign. Consequently, you will not receive a very pleasant grade.

1.1 Resources

To tackle this homework, we've provided:

- A `myLib.h` file containing all of the necessary GBA declarations such as DMA, `videoBuffer`, etc.
- A `Makefile` you can use to compile and run your program by typing `make vba`

Feel free to use code from these class resources, as you need to, but not from your friends or random, sketchy internet sites.

1.2 Warnings

- Do not use `float`'s or `double`'s 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 done in software, not hardware. Anywhere you use `float`'s, gcc has to insert assembly code to convert integers to that format. If you do need such things then you should look into fixed point math.
- Only call `waitForVBlank` once per iteration of your main loop
- 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 program is showing signs of tearing!
- If you use more advanced GBA features like sprites or sound, making them work is your responsibility; we (the TAs) don't necessarily know how they work, so we can't help you.

2 Requirements

- Should be in Mode 3:
  - You may use other modes (e.g. Mode 0, Mode 4), but **do so at your own risk**. It will be difficult to get TA help for modes other than Mode 3.
- You must also implement `drawImage3` with direct memory access (DMA). The prototype and explanation are given later in the assignment. If DMA has not been covered in lecture yet, you can first implement this function with `setPixel`, and then re-implement it with DMA once it's been covered in lecture.
- You must use 3 distinct images in your program, all drawn with DMA:
  - Two full screened images sized 240×160. One of these images should be the first screen displayed when first running your program.
  - A third image which will be used during the course of your program. The width of this image may not be 240 and the height of this image may not be 160.
  - Note: All images should be included in your submission.
- You must be able to reset the program to the title screen AT ANY TIME using the `SELECT` key.
- You must create a header (`myLib.h`), and move any `#define`'s, function prototypes, and `typedef`'s to this file from your code, along with your `extern videoBuffer` statement if you wish to use `videoBuffer` in other files. Remember that function and variable definitions should not go in header files, just prototypes and `extern` variable declarations.
- You must use at least one `struct`.
- Button input should visibly and clearly affect the flow of the program
- You must have two-dimensional movement of at least one entity. One entity moving up and down and another moving left and right alone does not count.
- You should implement some form of object collision. For programs where application of this rule is more of a gray area (like Minesweeper), core functionality will take the place of this criteria, such as the numbers for Minesweeper tiles calculated correctly, accurate control, etc.
- You must implement `waitForVBlank` and the `SCANLINECOUNTER` declaration.
- Use text to show progression in your program. Use the example files from lecture, and you can find more information in Tonc: `http://www.coranac.com/tonc/text/text.htm`
- There must be no tearing in your program. Make your code as efficient as possible!
- Include a `README.txt` file with your submission which briefly explains the program and controls.
- Do not include `.c` files into other files. Only `.h` files should be included and `.h` files should contain no functional code!

3 Deliverables

Please archive your source code files as a `.zip` or a `.tar` and upload to Canvas under the "Homework 10" assignment.

This includes all `.c` and `.h` files needed for your program to compile. Do not submit any compiled files. You can use `make clean` to remove any compiled files.

Download and test your submission to make sure you submitted the right files.

4 What to Make?

You may either: (1) create your own program, the way you wish it to be, as long as it covers the requirements, or (2) you can make programs that have been made before with your own code.

Your assignment must be entirely yours and not based on anyone else's code. As a reminder, this means you may not base your program on the code posted from lecture. Programs slightly modified from lecture code will be subject to heavy penalties.

Here are some previous programs you can either create or use as inspiration:

Interactive Storybook:

- Recreate a story from a movie or a book using the GBA
- Use text to narrate what is currently happening in the scene
- Use the controls to advance to the next scene or control a character within the scene
- Smooth movement (for any moving characters or objects)
- Start off with a full screen title image and end with a full screen credits image

Galaga:

- Use text to show lives
- Game ends when all lives are lost, and level ends when all aliens are gone
- Different types of aliens: there should be one type of alien that rushes towards the ship and attacks it
- Smooth movement (aliens and player)

The World's Hardest Game:

- Smooth motion for enemies and player (no jumping around)
- Constriction to the boundaries of the level
- Enemies moving at different speeds and in different directions
- Sensible, repeating patterns of enemy motion
- Enemies and the Player represented by Structs

Flyswatter:

- Images of yellow jackets or flies moving smoothly across the screen
- Player controlled flyswatter or net to catch the flies
- Score counter to keep track of how many flies have been swatted
- Fullscreen image for title screen and game background
- Enemies and the Player represented by Structs

Note: Do not make *Pong*! Everyone asks if they can make *Pong*, and it's simply a boring, low-effort product.

5 GBA Coding Guidelines

5.1 Installing Dependencies

To install dependencies, run:

```
$ sudo apt update
$ sudo apt install gcc-arm-none-eabi cs2110-vbam-sdl cs2110-gba-linker-script nin10kit
```

Note that this requires Brandon "The Machine" Whitehead's CS 2110 PPA, which you should've added earlier in the class for `complx`. If you need to re-add this, run the following and then run the two commands above again:

```
$ sudo add-apt-repository ppa:tricksterguy87/ppa-gt-cs2110
```

5.2 Building and Running your Code

To build your code and run the GBA emulator, run:

```
$ make vba
```

5.3 Images

As a requirement, you must use at least three images in your program and draw them all using `drawImage3`. To use images on the GBA, you first need to convert them into the suitable format. We recommend using a tool called `nin10kit`, which you installed with the command above.

You can read about `nin10kit` in the `nin10kit` documentation (there are pictures!):

`https://github.com/TricksterGuy/nin10kit/raw/master/readme.pdf`

`nin10kit` reads in, converts, and exports image files into C arrays in `.c` and `.h` files ready to be copied to the GBA video buffer by your implementation of `drawImage3`! It also supports resizing images before they are exported.

You want to use Mode 3, since this assignment requires it, so in order to convert a picture of smelly, festering garbage into GBA pixel format in `garbage.c` and `garbage.h`, resizing it to 50 horizontal by 37 vertical pixels, you would run `nin10kit` with the following command:

```
$ nin10kit --mode=3 --resize=50x37 garbage garbage.png
```

This creates a `garbage.h` file containing:

```
extern const unsigned short garbage[1850];
#define GARBAGE_SIZE 3700
#define GARBAGE_LENGTH 1850
#define GARBAGE_WIDTH 50
#define GARBAGE_HEIGHT 37
```

You can use this in your program by inserting `#include "garbage.h"`.

The `garbage.c` generated, which you should add to the `Makefile` under `OFILES` as `garbage.o` if you plan to use it, contains all of the pixel data in a huge array:

```
const unsigned short garbage[1850] =
{
        0x7fff,0x7fff,0x7fff,0x7fff,0x7fff, // ...
        0x7fff,0x7fff,0x7fff,0x7fff,0x7fff, // ...
        // ...
        0x7fff,0x7fff,0x7fff,0x7fff,0x7fff, // ...
        0x7fff,0x7fff,0x7fff,0x7fff,0x7fff, // ...
};
```

We've included `garbage.png`, `garbage.c`, and `garbage.h` in the homework `.zip` so you can check them out yourself. To draw the garbage in your own game, you can pass the array, width and height to your `drawImage3` accordingly: `drawImage3(10, 20, GARBAGE_WIDTH, GARBAGE_HEIGHT, garbage)` (to draw at row 10 and column 20). The next section will cover `drawImage3` in more detail.

5.4 DMA / drawImage3

In your program, you must use DMA to code `drawImage3`.

Drawing to the GBA screen follows the same guidelines as the graphics functions you implemented for Homework 09. The GBA screen is represented with a `short` pointer declared as `videoBuffer` in the `myLib.h` file. The pointer represents the first pixel in a 240 by 160 screen that has been flattened into a one-dimensional array. Each pixel is a `short` and has a red, green, and blue portion just like the pixels in Homework 09. There is a little modification (Hint: Include the graphics and geometry header files and use `videoBuffer` as the screen buffer), but you should be able to use the same code you implemented in Homework 09 to draw to the GBA screen.

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 it is covered in lecture, 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, you can choose to implement `drawImage3` without DMA and then, when you learn DMA, rewrite it. **Your final answer for `drawImage3` must use DMA!**

You must not use DMA for one pixel copies. Doing this defeats the purpose of DMA and is slower than using `setPixel`! Solutions which 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 :)
}
```

Note: If your implementation of this function does not use all parameters passed in then you are not implementing the function correctly. Understand that DMA acts as a for loop, but it is done in hardware. You should draw each row of the image, and allow DMA to handle drawing each row of the image.

5.5 GBA Controls

Here are 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

The directional arrows are mapped to the same directional arrows on the keyboard.

5.6 C Coding Conventions

- Do not jam all your code into one function (i.e. the `main` function).
- Split your code into multiple files. For example, you can have logic in your main file and library functions in `myLib.c` with declarations in `myLib.h`.
- Do not include `.c` files in other files. Only `.h` files should be included.
- `.h` files should contain no functional code.
- Comment your code, and comment what each function does. The quality of your comments will be factored into your grade!

6 Rules and Regulations

6.1 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.

6.2 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 Canvas 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.

6.3 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 Canvas. 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 Canvas.

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 11:54AM. You alone are responsible for submitting your homework before the grace period begins or ends; neither Canvas, 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.

6.4 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 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. This includes simply e-mailing it to them so they can look at it. 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 (Github), pastebin, etc. If you would like to use version control, use github.gatech.edu

6.5 Is collaboration allowed?

Collaboration is allowed on a high level, meaning that you may discuss design points and concepts relevant to the homework with your peers, as well as help each other debug code. What you shouldn't be doing, however, is paired programming where you collaborate with each other on a low level. Furthermore, sending an electronic copy of your homework to another student for them to look at and figure out what is wrong with their code is not an acceptable way to help them, and it is often the case that the recipient will simply modify the code and submit it as their own.

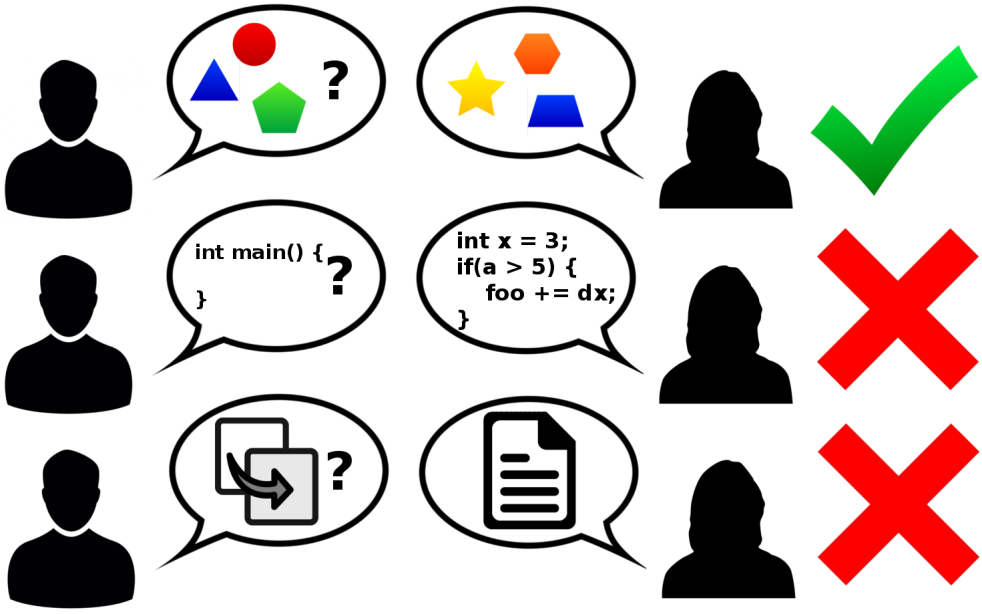

Figure 1: Collaboration rules, explained