LAB 02:

Input and Collision

Provided Files

- main.c
- myLib.c
- myLib.h

Files to Edit/Add

- main.c
- myLib.c
- myLib.h
- Makefile
- .vscode
    - tasks.json

Instructions

Note: Make sure to copy over your Makefile and .vscode/tasks.json from one of your previous assignments.

TODO 1.0

- void drawRect(int col, int row)
    - Complete this function in myLib.c (right under the setPixel function). You should already know how to do this from your last lab.
    - In main.c, find and uncomment UNCOMMENT#1. You should see a blue rectangle in the center of the screen.

TODO 2.0

- void fillScreen(unsigned short color)
    - Head back to myLib.c, find fillScreen (right beneath drawRect) and complete it.
    - You may only use a single loop (one for-loop) to complete this one.
        - **Hint:** the GBA screen has a total of 240 x 160 pixels which is equal to 38400.

- Enter main.c and uncomment UNCOMMENT#2. The background of your game should now be a lovely shade of cyan, and you should see three new rectangles. There should be 2 eye rectangles that are white, and one black mouth rectangle at the bottom of the screen that moves horizontally. There should also still be the blue rectangle in the center of the screen.

TODO 3.0-3.2

This one requires a lot of moving parts to work, so it is broken into three parts.

- TODO 3.0
  - Open myLib.h, find the button macros, and complete them.
  - For BUTTON_PRESSED, assume that buttons and oldButtons have been set up appropriately.
- TODO 3.1
  - Since buttons and oldButtons *haven't* been set up correctly, head on back to main.c and initialize them in the initialize function.
  - They have already been declared in main.c (and in myLib.h as extern), so you don't have to worry about that part this time, but you will when you code things yourself for your homework.
- TODO 3.2
  - The buttons still won't do anything unless you update them each frame, so do that in the main while-loop.
  - After you have completed these tasks, find UNCOMMENT#3 in the update function and do the thing.
  - Run it, and now you can press button A to toggle the eyes between white and black, as if they are closing and opening..
  - Holding down the A button for a long time should have the same effect as just tapping it a single time. If not, you did one of these three TODOs incorrectly.

TODO 4.0

- Now that you can take button input, find TODO 4.0 in the update function and complete it so that the blue rectangle can move up, down, left, and right if you press the corresponding arrow key.
  - This time, it should move for as long as the key is held, meaning that if you tap it once quickly, it will barely move, but if you press it for several seconds, it will move a lot. If not, fix that.
- Compile and run, and you should be able to move the blue rectangle anywhere on the screen.

TODO 5.0-5.2

This is the most exhaustive part of the lab, but also the most important.

- TODO 5.0
    - At the bottom of myLib.c, implement the collision function. It takes in the positions and dimensions of two rectangles, and returns a 1 if they are colliding, or a 0 if not.
        - **Hint:** using graph paper, or drawing a grid yourself, is extremely useful. This function should work regardless of the size or velocity of the rectangle (if one is moving).
- TODO 5.1
    - Now that you have the collision function implemented, use it in the update function. If the blue rectangle collides with any of the two stationary eye rectangles, reset the blue rectangle's position to the center of the screen (this is where the blue rectangle was first initialized).
- TODO 5.2
    - Now use the collision function to account for if the blue rectangle collides with the moving mouth rectangle. Additionally, if mColor (the mouth's current color) is black, change it to red. If it is already red, change it to black.
        - **Hint:** you can refer to the logic that was used to make the eyes change colors when you press A.
    - Compile and run. Verify that it runs correctly, and then you are done.
- **Hint:** since there are three separate rectangles for which you need to detect a collision, you'll need to use the collision function three separate times.

You will know if it runs correctly if you can:

- Move the blue rectangle using the arrow keys
- Press A to change the eye color from black to white, and vice versa
- Make the blue rectangle collide with any of the other rectangles and be transported to the middle of the screen
- Collide with the mouth rectangle at the bottom, changing the color from black to red, or vice versa.

Tips

- Start early, and attend recitation
- Use graph paper and draw pictures to conceptualize what is happening
- Follow each TODO in order, and only move forward if everything is correct

---

Submission Instructions

Zip up your entire project folder, including all source files, the Makefile, and everything produced during compilation **(including the .gba file)**. Submit this zip on Canvas. Name your submission Lab02_FirstameLastname, for example: “Lab02_MickeyMouse.zip”.