CS 2261 Lab 06: Mode 4

Provided Files

- main.c
- myLib.c
- myLib.h
- game.c
- game.h
- text.c
- text.h
- font.c
- font.h
- pumpkin.bmp
- Joshy.png

Files to Edit/Add

- main.c
- myLib.c
- pumpkin.c
- pumpkin.h
- Joshy.c
- Joshy.h
- Your Makefile

Instructions

In this lab, you will be completing several different TODOs, which will, piece by piece, add Mode 4 drawing to a simple Space Invaders-like game. Each TODO represents a component of this improvement, and is broken down into sub-TODOs. Your code may not compile until you complete an entire TODO block, at which point the game should compile with the new drawing function working as expected.

After you download and unzip the files, add your Makefile, add the SOURCES with it, and then take a look at the code. What you see should look familiar. It's a completed

Lab04! It has a few new features to highlight and test some Mode 4 functionality. If you compile and run it right now, however, it will be a blank black screen. Complete theTODOs on order, paying close attention to the instructions.

TODO 1 – setPixel4

- For us to be able to set pixels in Mode 4, we need to account for the additional cases that Mode 4 requires, so we're making a new function.
- TODO1.0: InmyLib.c, completethesetPixel4function.
- Compile and run. If you press START, you should be able to travel to the Game state and see that the text is drawing correctly (drawChar4 and drawString4 have already been written for you). If not, fix this before going further.

TODO 2 – fillScreen4

- We want to be able to see the rest of the states, so we need to fill the screen next.
- TODO2.0: In myLib.c, complete the fillScreen4 function.
- TODO 2.1: At this point, if you compile and run, you won't see anything. That's because our goto state functions are only filling the screen once (on the hidden page), then doing nothing. To fix this, we need to flip the page with flipPage() after do our drawing and after we wait for vertical blank. In main.c, in the goToStart function, wait for vertical blank, then flip the page.
- TODO 2.2: Do the same for goToPause.
- TODO 2.3: Do the same for goToWin.
- TODO 2.4: Do the same for goToLose.
- Compile and run. You should be able to travel through all the states you just edited and see their titles printed on them (except for Start, because drawFullscreenImage is still blank). If not, fix this before going further.

TODO 3 – drawFullscreenImage4

- We want to actually be able to see the start screen. Let's make it so we can draw fullscreen images.
- TODO3.0: In myLib.c,completedrawFullscreenImage4.
- TODO 3.1: This isn't worth anything if we don't have an image to draw. Open Joshy.png in Usenti. Resize it to 240x160 and be sure the image scales down rather than it getting cropped.
- TODO 3.2: We're in Mode 4, so the image has to be a max of 256 colors. Usenti has a tool to do this for us. Go to Palette > Requantize, type 256, and hit OK.
- TODO 3.3: Export your image, this time selecting "8bpp" in the export settings (8 bits per pixel means a char for each pixel, like Mode 4 wants). Also make sure

that Pal (in the top right) is checked. This will include the palette in the .c file as an array of 256 shorts.
- TODO3.4: Add the new Joshy.c to your Makefile SOURCES.
- TODO3.5: Include Joshy.h at the top of main.c.
- UNCOMMENT3.0: Uncomment the call to drawFullscreenImage4 in the goToStart() function.
- TODO 3.6: We need to be able to load the image's palette (located in Joshy.c) in the game, so that the hardware knows what colors to use. In main.c, in the goToStart() function, write a single call to DMANow() that will copy the entire JoshyPal into the game's PALETTE.
- Compile and run. You should be able to see all of the states now (except for the moving elements in Game). If not, fix this before going further.

TODO 4 – drawImage4

- We want to be able to draw smaller images as well.
- TODO4.0: In myLib.c, complete drawImage4.
- TODO 4.1: Open pumpkin.bmp in Usenti. This one was drawn using a small enough palette and size, so there is no need to resize or requantize.
- TODO 4.2: Export this image the same way you did the last one, making sure that Pal is checked. In initGame(), write the same DMANow() call you wrote to load in the palette last time, this time loading in pumpkinPal.
    - Note: This time, after we load in the palette, we also put some of our own colors in there (you can view this in game.c. It looks super complicated, and you don't have to do it this way when you are making your games). It has already been done for you.
- TODO4.3: Add the new pumpkin.c to your Makefile SOURCES.
- TODO4.4: Include pumpkin.h at the top of game.c.
- UNCOMMENT 4.0: Uncomment the rest of drawBall().
- Compile and run. When you reach the Game state, you should see two pumpkins bouncing around. If not, fix this before going further.

TODO 5 – drawRect4

- For our last touch, we need to be able to draw rectangles.
- TODO 5.0: In myLib.c, complete drawRect4. This will be by far the longest function you code in this lab. You need to account for all cases where you have to set pixels in front or behind each row. You also need to make sure it still works if you draw a rectangle of a small width (1 or 2).
    - Hint: draw lots of pictures. There are four col/width cases, so make sure you identify and account for all of them.

- Compile and run. When you reach the Game state, you should be able to see all of the rectangles you are familiar with, plus some mixed into the pumpkins. If you shoot, the bullets are now of different widths, and they don't fly straight up when you are moving; they lean in the direction you were moving when you shot. If any of your homeworks work like this, we will assume you copied code, and penalize heavily. These additions are to help you test drawRect. The rectangles should all be correctly sized, and they should not wiggle when they move diagonally (they should move smoothly in that direction). If this all works, submit your lab.

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 Lab06_FirstameLastname, for example: "Lab06_BatuRem.zip".