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`
- `Bill.png`

Files to Edit/Add

- `main.c`
- `myLib.c`
- `pumpkin.c`
- `pumpkin.h`
- `Bill.c`
- `Bill.h`
- Your Makefile

Instructions

In this lab, you will be completing several different `TODO`s, 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 Lab05! 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 the `TODO`s 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.
  - TODO 1.0: In `myLib.c`, complete the `setPixel4` function.
  - 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 (we already did `drawChar4` and `drawString4` 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.
  - TODO 2.0: 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 (except for Start, because `drawFullscreenImage` is still blank) and see their titles printed on them. 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.
  - TODO 3.0: In `myLib.c`, complete `drawFullscreenImage`.
  - TODO 3.1: Since this is Mode 4, and all pixels hold indices of the palette, we need to be able to load an entire pre-made palette with a function. In `myLib.c`, complete `loadPalette`. It takes in an array of 256 shorts.
  - TODO 3.2: None of this is worth anything if we don't have an image to draw. Open `Bill.png` in Usenti. Resize it to 240x160 like we did before.
  - TODO 3.3: This time, 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 3.4: 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, which you can load up with `loadPalette`.
  - TODO 3.5: Add the new `Bill.c` to your Makefile `SOURCES`.
  - 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.
  - TODO 4.0: 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. This time, when we loaded in the palette in `initGame`, 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.
  - TODO 4.3: Add the new `pumpkin.c` to your Makefile `SOURCES`.
  - 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. 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 T-Square. Name your submission Lab06_FirstameLastname, for example: "Lab06_ZharLestin.zip".