LAB 05:
Text and DMA

Provided Files

- main.c
- myLib.c
- myLib.h
- game.c
- game.h
- text.c
- text.h
- font.c
- font.h

Files to Edit/Add

- main.c
- mylib.c
- text.c
- Makefile
- .vscode
    - tasks.json

Instructions

In this lab, you will be completing several different TODOs, which will, piece by piece, add text and speed to a simple Space Invaders-like game. If you compile and run the code after unzipping, what you'll see should look familiar. It's a completed Lab04! This isn't as great as it could be, though. Each TODO represents a component of 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 improvement.

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

TODO 1.0 - drawChar

For us to be able to draw text, we need to be able to draw a single character first.

- TODO 1.0: In `text.c`, complete the `drawChar` function.
- TODO 1.1: In `main.c`, uncomment UNCOMMENT #1
- Build and run. You should be able to see this:

  

  If you do not see this, there is something wrong with your drawChar function. Fix this before going further.

TODO 2.0 - drawString

We want to be able to draw entire strings of text with a single function, as well.

- TODO 2.0
  - In `text.c`, complete the `drawString` function.
- TODO 2.1
  - In `main.c`, in the `goToStart()` function, write "CS 2261" at (99, 96) using `drawString()`
- TODO 2.2
  - In the `goToWin()` function, draw "WIN" at (12, 12)
- TODO 2.3
  - In the `goToLose()` function, draw "LOSE" at (12, 12)
- TODO 2.4
  - In the `goToPause()` function, draw "PAUSE" at (12, 12)
- Build and run. You should be able to travel through all the states you just edited and see their titles printed. If not, fix this before going further.

- **NOTE:** Why do we do this in the goTo functions, and not the every-frame state functions? That's because drawing text takes a long time. We don't want to draw text every frame unless we absolutely need to.

TODO 3.0 - Score

For our game to be user-friendly, we want to be able to see our current progress to victory during the game state.

- TODO 3.0
  - In the `goToGame()` function, draw "Ball Count: " in a free area (col 5 and row 145 should be a good spot)
- TODO 3.1
  - In the `game()` function, use `sprintf()` to save the current count of balls remaining (`ballsRemaining`) in the text form to a character array.
    - The char array has already been created for you, called "`buffer`". sprintf is a function in `stdio.h`, which we have #included for you. It returns null, and works like this:
      - `sprintf(arrayName, formatterString, variables, …)`
      - This is exactly like printf, but with an extra argument at the beginning (the arrayName to save to)
      - For more info on sprintf, check https://www.tutorialspoint.com/c_standard_library/c_function_sprintf.htm
- TODO 3.2
  - We want to erase the previous score before we draw the new one. So, draw a black rectangle at (76, 145) that is the size of a character.
    - **Hint:** look back at `font.c` - how wide and how tall is each character?
  - Then, draw the score at this location (using `drawString` with `buffer` as the character array parameter input).
- Build and run. You should be able to travel to the game state and see the current score. Shoot some balls, and the score should update. If not, fix this before going further.

- **NOTE:** Why did we draw "Ball Count: " ino `goTGame()`, but the actual number in `game()`? That's because the "Ball Count: " text does not change, so we only need to draw it once. The actual score, however, can change at any time, so we have to account for that possibility every frame.

TODO 4.0 - DMANow

For our game to be fast, we need to use DMA. The simplest way to do this is to write a function that sets up the registers of the specified channel.

- TODO 4.0
    - In `myLib.c`, complete the `DMANow` function.
    - This function sets up all the registers of the given DMA channel and turns it on for us. This allows us to use DMA wherever we need it with only a single line, without having to set all of the registers line-by-line in every location we want to use it. There are additional comments in the `DMANow` function that will help you write it.
- TODO 4.1
    - Rewrite the `fillScreen` function to use DMA, using your new `DMANow` function.
    - Make sure to use **DMA channel 3**. You **may not** use any loops.
    - **Hint:** If you are copying one thing (one source) to every pixel in the videoBuffer, what do you need to tell the DMA control to do (or not do) to the source? Look at myLib.h for helper macros!
    - **Hint:** Make sure the `src` and `dst` parameters are *addresses* to the locations you are copying from and to.
- TODO 4.2
    - Rewrite the `drawRect` function to use DMA, using your new `DMANow` function.
    - Make sure to use **DMA channel 3**. You may only use **one** loop.
    - **Hint:** You must use one loop here, because unlike fillScreen, the area you are copying to (destination) is not contiguous. Each row of the rectangle is, though. Use DMA to draw one row at a time.
    - **Hint:** Make sure the `src` and `dst` parameters are *addresses* to the locations you are copying from and to.
- Build and run. The game should look the same, but a lot snappier during transitions. If this all works, submit your lab. If not, fix it before submitting.

You will know if it runs correctly if:

- You can see the GBA text logo on the start screen and all other states are labeled
- The score updates in the game state
- There are no issues with fillScreen and drawRect after rewriting them to use DMA (the game should still play like it did in Lab04)

Tips

- Review lecture and recitation materials for how to implement `drawChar` and `drawString`.
- 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 Lab05_FirstameLastname, for example: “Lab05_GameBoy.zip”.