LAB 07:
Animated Sprites

Provided Files

- main.c
- lab07Lib.c
- lab07Lib.h
- spritesheet.bmp
- gameBg.bmp
- gameBg.c
- gameBg.h

Files to Edit/Add

- main.c
- lab07Lib.c
- spritesheet.c
- spritesheet.h
- makefile
- .vscode
    - tasks.json

Instructions

In this lab, you will be completing several different TODOs, which will, piece by piece, complete an animated Link (from the Legend of Zelda) character that the player can control. This lab will provide significantly more detail in the comments of the code, so **READ VERY CAREFULLY**. Your code may not compile until you complete an entire TODO block, at which point the game should compile with a new component of the final outcome (unless otherwise specified).

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

TODO 1.0 - Loading and Enabling Sprites

Let's set up sprites!

TODO 1.0

- ➔ In main.c, #include spritesheet.h

TODO 1.1

- ➔ In initialize(), load the spritesheet palette and tiles into their desired spaces in memory
  - ◆ **HINT:** The sprite palette is different from the background palette (check lab07Lib.h)
  - ◆ **HINT:** Which charblock do sprite tiles go in?

TODO 1.2

- ➔ In lab07Lib.c, complete hideSprites()
- ➔ In main.c, call hideSprites()

TODO 1.3

- ➔ In initialize(), enable sprites

Build and run. You should not see anything new. If you see anything in the top-left (the dreaded CORNERFACE), fix this before going further.

TODO 2.0 - Animating Sprites

At the bottom of initialize(), READ CAREFULLY all of the animation variables for the Link player. Look back at the spritesheet to see how it is organized

Let's get Link showing and their animations working!

TODO 2.1

- ➔ If the sprite is not idle:
  - ◆ Set the previous state to the current state
  - ◆ Then reset Link's state to idle

TODO 2.2

- ➔ Increment Link's current frame
  - ◆ **HINT:** there are only numFrames number of frames, and the frames of the animation must end up LOOPING

TODO 2.3 - 2.6

- ➔ Set Link's aniState accordingly

TODO 2.7

- ➔ If Link is idle:

- We want the current frame to be of Link standing (frame 0) in whatever direction he was last facing (current state set to the previous state)

➔ Else:

- Increment the animation counter

TODO 2.8

➔ Set up all of the sprite attributes, place the sprite at index 0 of the shadowOAM

- Look at lab07Lib.h for sprite stuff!
- **Hint:** Open up spritesheet.bmp and look at how the sprites are organized. Animation states correspond to columns, and frames are the rows of the spritesheet
- **Hint:** this sprite is 32 x 32, you are going to need to do some multiplication when setting up ATTR2_TILEID
  - Each tile is 8x8
  - When Link is idle, where does that correspond to on the spritesheet? (0,0) because the sprite begins at tile 0,0 and is 4 tiles wide and 4 tiles tall

TODO 2.9

➔ Copy the shadowOAM into the OAM

- **HINT:** how many sprites can we have in the OAM? How many attributes does each sprite have?

Build and run. You should be able to walk the player Link around, see the background repeat and all while the player is at the bottom-center of the screen. If this is not the case, fix this before moving forward

TODO 3.0 - Animating Multiple Sprites

At the bottom of initialize(), READ CAREFULLY all of the animation variables for the Fairies. Look back at the spritesheet to see how it is organized.

Let's get multiple Fairies showing and animating! These sprites will all be doing the same animation (a single state) and it will be continuously looping, so no need to worry about checking for button inputs here.

TODO 3.1

➔ Increment each Fairy's current frame of animation every 12 frames of gameplay

➔ You will also need to increment each Fairies aniCounter

➔ Refer to the code associated with TODO 2.2 for how to check for frames of gameplay

- **HINT:** there are only numFrames number of frames, and the frames of the

animation must end up LOOPING

TODO 3.2

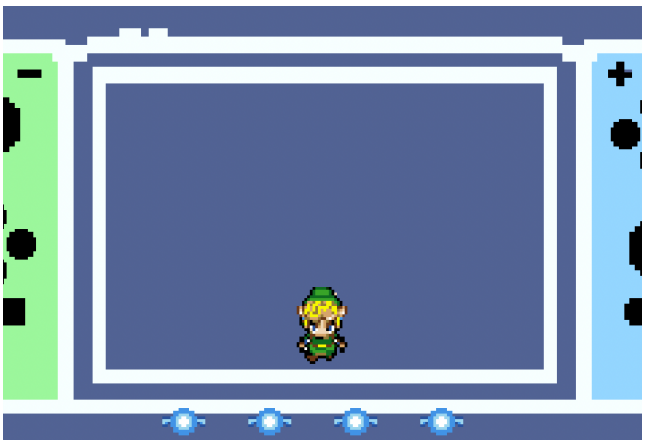
- ➔ Loop through each Fairy and set up all of the sprite attributes
- ➔ **Be careful** when placing the sprites in the shadowOAM as to not **overwrite where the player sprite is** (shadowOAM[0])
  - ◆ Since we only have a single state for our Fairies, it's col value of TILEID will not change. However, now we are starting from lower in the spritesheet, so we will need to add the starting tile row (12) to our curFrame value calculation
    - Look at lab07Lib.h for sprite stuff!
  - ◆ **Hint:** Open up spritesheet.bmp and look at how the sprites are organized. Animation states correspond to columns, and frames are the rows of the spritesheet
  - ◆ **Hint:** this sprite is 16 x 16, you are going to need to do some multiplication when setting up ATTR2_TILEID for the curFrame
    - Each tile is 8x8
    - Where does the Fairy sprite start in the spritesheet? (0,12) -- in tiles. This location is the first frame of our Fairy
    - The sprite is 2 tiles wide and 2 tiles tall

Build and run. You should now see 4 Fairies at the bottom of the screen, continuously moving up and down slightly and shining. They should follow the player, stuck at the bottom of the screen as the player Link moves.

You will know if it runs correctly if:

1. You can walk the player around infinitely.
2. See your background repeat while the walking Link stays in the middle/bottom of the screen.
3. See the four Fairies at the bottom of the screen move continuously.
4. You DO NOT see corner face.

You should get something similar to this, but animated:



Tips

- ➔ Follow each TODO in order, and only move forward if everything is correct.
- ➔ Review recitation and lecture notes on spritesheets and animating sprites.

Submission Instructions

Ensure that **cleaning** and building/running your project still gives the expected results. **Please reference the last page of Lab02.pdf for instructions on how to perform a "clean" command.**

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 Lab07_LastnameFirstname, for example:

> “Lab07_ZeldaPrincess.zip”

It is your responsibility to ensure that all the appropriate files have been submitted, and that your submitted zip can be opened and everything cleans, builds, and runs as expected.