**Lab 4 Observations**

**Chapter 6, Problem 1:**

This question asked me to change Snail Bait’s sprite metadata to move sprites to different initial locations. All this involved was going through the data and changing the “left” and “top” attributes for everything that supported it. For those that didn’t, such as the button data, I was able to change the platform index to get them to appear on different platforms. I only made minor adjustments, usually moving sprites back or forward and up or down by 100-200 pixels, but the changes are there. For the buttons and snail, I moved them one platform backwards. None of the new positions make any sense, which should help to show that I moved them all.

**Chapter 6, Problem 2:**

This problem asked me to add another sprite to the game. For this problem, and in preparation for the runner replacement, I grabbed the flour sack sprite sheet from the provided sprites. I then added it to the main sprite sheet. Once done, I added all the relevant sprite information to the main script, including the cell locations, sizes, and so on. That was the most annoying part of the entire thing because doing it manually took a lot of time. Using a generated JSON file would be a much better solution if it was an option. After that, it was a matter of copying the required sprite variables and functions over to make everything work. I added a flour bag data array, I added the flour bag to addSpritesToSpriteArray(), and initializeSprites(), then I added a createFlourBagSprites() function that I copied from createSnailSprites() and added that function to createSprites(). With that, the flour bag showed up on the second platform.

**Chapter 6, Problem 3:**

This problem asked me to add a zIndex property to the Sprite object and modify the Sprite.draw() method to draw sprites with higher z-indexes on top of sprites with lower z-indexes. We are only supposed to discuss a solution to the problem.

The solution that I would use is a bit complicated but would work for everything. I would start by declaring two extra constants called Z\_INDEX\_MIN and Z\_INDEX\_MAX. These would hold the minimum and maximum z-index used by all the sprites respectively. Finally, I would add a variable called spritesMap to the drawSprites(), which is an empty map/

For the code, I would first make a for loop that inserts keys from Z\_INDEX\_MIN to Z\_INDEX\_MAX with an empty array as a value into the map. Then, a second for loop (using the same variables) would go through the array of sprites as it currently does, except instead of drawing the sprites, it would push the sprite to the appropriate array in the map. Finally, a third for loop (using the same variables) would grab the arrays from the map and draw them in order, from lowest to highest z-index.

There are probably better ways to do this, but this ensures that the sprites are always drawn in the correct order with no potential undefined arrays. Maps are useful here because they guarantee that the keys will be iterated in the order of insertion (in this case, lowest z-index to highest z-index).

**Chapter 7, Problem 1:**

This problem asked me to modify the runner’s run behavior so that she only runs when she’s on a platform. This involves detecting when the runner is on a platform, and there are a couple ways to go about this. The easiest way to do this is to get the position of the bottom left and right of the runner as well as the platform bounds (using the left and top of the platform as well as the hOffset). Once that’s done, we can run a for loop on every platform in the game. If the runner’s feet are within the bounds of the platform, then play the running animation. Otherwise, do not. This is how I implemented the check. I added those variables and the for loop to the runBehavior execute() function. It checks to see if the player is in the platform bounds and, if so, sets the runnerOnPlatform bool and breaks. Otherwise, it never gets set and the function returns early.

**Chapter 7, Problem 2:**

This problem asked me to modify the snail’s behavior so that it shoots a bomb every second. The book directly talks about this behavior and how to change this. The snail’s shooting behavior is directly tied to which animation frame it currently has. When the snail sprite is instantiated, it is given a CycleBehavior() behavior with an interval of 1500ms. This is the amount of time that the snail waits before looping through the shoot animation again. Changing this to 1000ms makes it run its shoot animation every second, thus resulting in a bomb shot every second.

**Chapter 7, Problem 3:**

This problem asked me to replace the runner with the new runner sprite we put in the sprite sheet in problem 6.2. Since we are directly overwriting the runner, the only thing we need to do is replace the runnerCellsLeft and runnerCellsRight array with the flour bags. Ideally, I would modify the draw functions directly to allow for mirrored drawing, but I didn’t know if I should touch those or not, so I kept it contained to the sprite sheet and JS file. Additionally, I don’t modify the RUNNER\_CELLS\_WIDTH or RUNNER\_CELLS\_HEIGHT variables because they work for our purposes. Changing RUNNER\_CELLS\_HEIGHT also makes the run animation code no longer work because the default position of the runner is no longer low enough to contact the platform, and that is a big change that could cause breaking issues.