Implementation Log – Assignment 4
Sam Vance

I started this assignment coming up with different ideas. I considered an RPG style game or a platformer, but ultimately landed on a Crossy Road game, focusing on the gameplay of crossing the screen to complete each level while avoiding the obstacles.

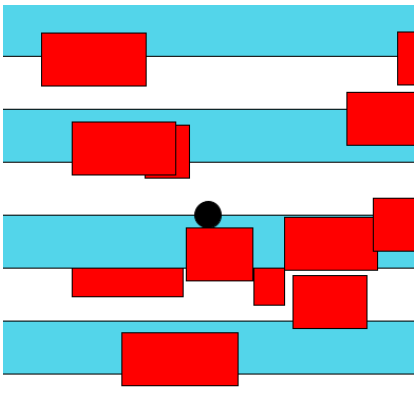
I began by writing out a basic Pseudocode to work off of. This helped me understand what I needed to work and what kind of functions I will need to implement, especially for my objects to work.

I began writing the program by adding my objects:

- Player – representing the user-controlled character that they can control using the arrow keys
- Obstacles – representing the shifting obstacles that will get in the player's way from completing each level. Their size and speed will vary
- Lanes – this is something I considered after the fact. Using an ArrayList containing Obstacle objects to place inside the lanes, allowing me to control how many there will be and their speed
- Game states – it wouldn't be a game without game states. I added two Boolean variables, gameOver and titleScreen. The game will constantly be checking if these are marked true, and if they are, will cease the gameplay and show the appropriate screen

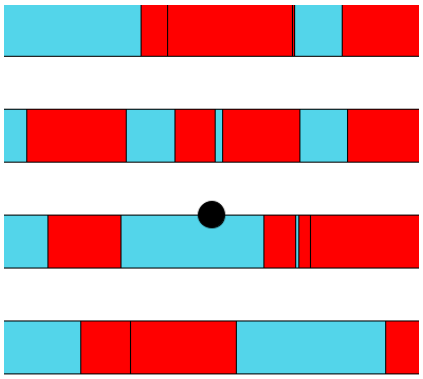
I began with inputting basic functions and getting a player-controlled character into the game. I made attempts at making a title screen, but I then focus on the Obstacles and Lane objects.

Problem 1 – Lanes and Obstacles



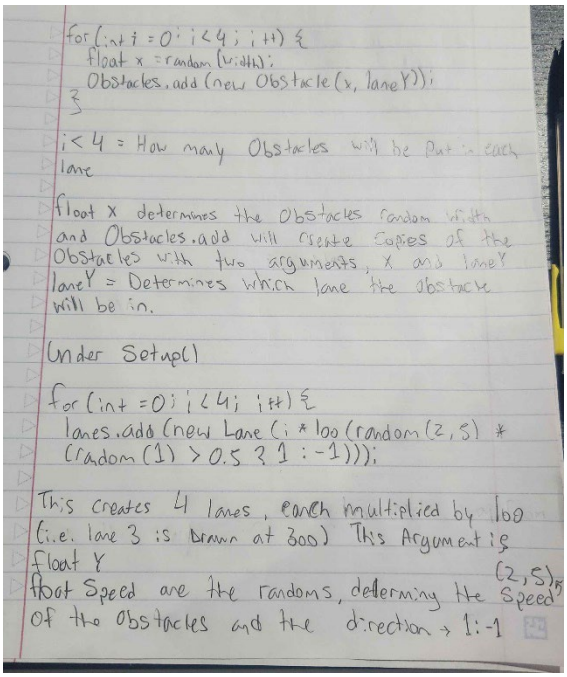
Problem: This was my first attempt at getting this to work. The obstacles were bouncing all over the place and were not set inside of the lane.

The idea is that my Obstacle class object would create the red squares and choose a random width. They would slide across the screen based on a incrementing velocity.



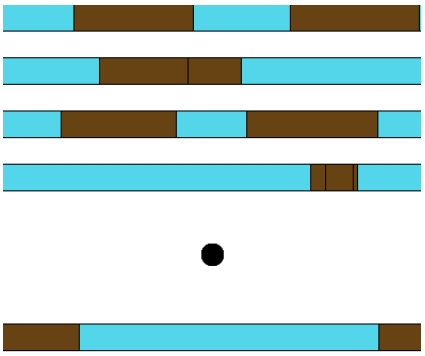
The Lanes would then call the Obstacles from an ArrayList and put them into each lane based on how many obstacles should be in each lane.

Solution: I added an update function in Lane and updated the previous Obstacle update function. Obstacle.update(float laneY) with position.y = laneY.



In Lane, I lock the obstacles to laneY so that the by position of each obstacle won't exceed laneY.
obs.update(laneY);

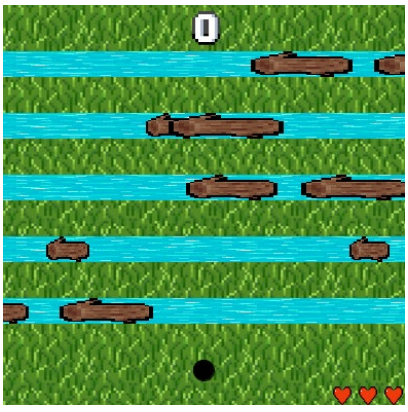
Problem 2 – Collision



Problem: I began working on the collision, trying to do a checkCollision Boolean function. The attached photo shows that when the player collides with an object, they make the lane and objects disappear.

Solution: I scrapped trying something new and went back to ole' reliable, Jeffrey Thompson. I used the circle and rectangle collision and it worked completely. I set the player to return to their starting position once colliding with the obstacle. I tweaked the program a little bit, because I was noticing the hitboxes were a little janky.

PImages and Gamification



Once I got the hard stuff out of the way and I had a functional game, I focused on making the game visually appealing. I relied a lot on PImages for the design of the game, unfortunately I could not find the backside of a pixel frog sprite.

I added a title screen and a game over press play to continue screen.

I added a score system to keep track of each level, and a life system.

Problem 3 – Game Over and Reset Game

Problem: I ran into a new problem when playtesting the game. In order to increase the difficulty of the game, I added a speed increase to the objects. However, by restarting the game after dying, the speed doesn't reset and stays at the same speed as you left it when dying.

I tried many things, such as adding resetSpeed functions to the lane and Obstacle objects, but they did not seem to work out at first. When restarting the game, the speed is reduced to 0 and then slowly builds up all at the same rate of speed, versus when you first boot up the game, the speed is already at set to its random value and increases from there.

Solution: I played with the program A LOT and eventually came up with this solution. Both Obstacle and Lane have a reset function that attempts to reset the speed by to a default speed. The Obstacle reset also chooses random positions for the logs so they are different every time. The Lane reset calls the Obstacle reset giving it that new default speed but randomizes it once more. I was struggling at the beginning because after 1 game over, the logs would all have the same speed. So to counter that, I added a random variable to the default speed. I initially called the reset function under the if(gameover) in the mousePressed function, even tried it in the draw function along with the level checker. But I eventually landed on the if(istitlescreen) conditional. Which is really should be the same as the if(gameover), except it works, sort of. I notice that the speeds aren't consistent with the initial speed when you first boot up the game, but after that initial game over, they all stay persistent and are capable of increasing with the level while being capable of resetting. It's not perfect and I would love to delve deeper into its intricacies and really figure out what is happening here, but for now, this is my solution.