Michael Zachor

I think I spent more time actually programming than learning concepts and syntax, and if I had spent more time learning the concepts my final project would have been better. Specifically for my project it would have been helpful to have a more in-depth understanding of arrays vs. arraylists, because the main problem with my code I think is that it would have worked better with an arraylist rather than an array. Throughout the semester I found myself using Processing a lot more than p5.js, simply because I think Processing is a lot more simple and user-friendly. OOP is meant to be more organized than procedural programming I think, and therefore it's helpful for bigger projects. But they are pretty similar, because both of them involve creating new objects.

For my final project, the concepts that solidified were classes and motion. At first I tried just making one class for all the people, but I realized that it was easier to make a different class for each floor, and then I realized that the more classes I have, the easier my code is to control and the more organized it is. That was my big breakthrough, because it finally got my code running a bit more smoothly. The biggest thing that I learned is that it's helpful to plan out your project beforehand. If I had known how complex my code was going to become, I would have chosen to use arraylists instead of arrays, because then I could have done things like deleting an object once it's offscreen, and instead of initializing a set number of people, just adding one every time I needed one. It was really hard to resolve my bugs. Because of how spread out my code was, it was hard to figure out what was causing the problem, and even once I found out which specific lines of code were causing the program, it was hard to decide which specific person was causing the program, since I had so many of them running at the same time. I was able to resolve some of the bugs myself, but Alex at the PTC also helped me out a lot. The reason I wasn't able to get my game to the level that I wanted to be at by the due date was because I was stuck with a lot of bugs, and I didn't know how to fix them. I tried lots of solutions and I spent a lot of time reading through my code slowly, reading it as though I was the computer, but because of the complexity and all the different calls to functions and classes I had in my code, it was hard to figure out which part specifically was causing the problem, so that's why I was stuck in the same place for a while. Even after Alex at the PTC looked at my code for two hours and figured it out, even he had trouble telling me where the bugs were, because there were so many places where they could have been. Overall, I'm happy with my progress on this project, and I'm glad I was able to fix a lot of the problems in the two extra days I got, and even though it doesn't look like it, I spent a lot of time on this project, and I'm proud I was able to get it working eventually, even if there's still a lot I could do to make it better. The important thing is I learned so much through this project. The way I approach code in the future will be completely different than it was before this project, and that's all thanks to my experience. I was able to realize that not all projects have simple solutions, and sometimes you really do have to start all over. I would have done that if I had more time, but I'm happy with what I was able to accomplish.

I will absolutely continue programming. Even though my final project wasn't what I hoped it would be, I learned so much in the progress. I learned that I need to map out my plans for the program before I start coding it, and I learned that it's always better to organize your code right

from the start rather than just typing out code and seeing if it works. I loved this class, even though I struggled through it at times, and I love all the possibilities that Processing allows you. Watching Daniel Shiffman's YouTube videos and all his examples really gets me excited and inspired, and I would love to make programs visualizing music data or recreating real-life physics to create examples of the real world. Coding is a world of its own that I just entered basically three months ago, and I'm really excited to see the places it takes me.

My final project is an interactive game. The concept is that the user is an elevator operator and he/she has to get the people who appear on screen onto the floor that they want to get to. They do this by clicking on an elevator door to get the elevator to that floor, where the door opens. The game is called Elevator Operator. Ideally, the level that I created would be a tutorial level, since there are only two floors and one elevator, and there would be two more, harder levels, with more floors and two elevators. Therefore, it would become a strategy/time game, where the user would have to get a certain amount of people to their desired floor before the clock runs out. The way the game stands now, though, it's more of just a simulation game, where the user is playing as the person who controls the elevators, and there's not really much of a goal, besides getting ten people to the floor they want to get to.