Jay Annadurai

MAD SP23

T. Faas

22 Jan 2023

Exercise 2: Functions

[Link to Github: https://jannadurai.com/Exercises/Exercise%202/index.html](https://jannadurai.com/Exercises/Exercise%202/index.html)

**Introduction**

     Exercise #1 primarily covers the basics of p5.js and utilizing control structures to modify visual behavior such as changing object colors or object movement. I mainly chose exercises 1, 2, and 3 in chronological order, but I'm fairly certain the code in Exercise 3 can be easily expanded to accommodate Exercise 4. As for Exercise 5, while its novel, utilizes the same basic concepts of declaring variables, initializing them in setup(), and then modifying values after using a control structure within draw(). Exercise #2 primarily covers the utilization of functions on top of the p5.js basics to modularize code and increase utility and readability. There were only 3 available exercises and 3 that needed to be done so all were written. Exercise 2.1 draws a rocket at the mouse position and modularizes the rocket draw calls into a single function. Exercise 2.2 draws an inner circle with a randomly-generated color and an outer circle with the same color but the red component eliminated. Exercise 2.3 I'm not quite sure what the purpose of it is, but roughly simulates a loop of circles drawn in polar coordinate space?

**Algorithms & Planning**

     All files do contain a similar structure of declaring variables outside of the setup() p5.js function, initializing them with values within p5.js's setup(), and then drawing pertinent and dynamic elements within p5.js's draw() function. Functions are written in a separate section after the p5.js draw() section.

**Reflection**

     A repeated insight I've encountered while doing the exercises is that modularization can be difficult to implement immediately. For example, with the 'Drawing Drawer' exercise that repeatedly draws a composition, the most difficult aspect for me was actually creating the 'composition' in the first place. To get a sense of scale, I went to a random pixel drawing website and began just putting shapes on the pixel grid to get a general sense of space. Frankly it was a bit difficult converting absolute positions of the shapes to relationships in two-dimensional space between shape positions. For example, 'noseconeHeight = rocketHeight - fuselageHeight - finExtraHeight' is conceptually more difficult than just plotting the point. However, it comes with the advantage of creating an easily scalable composition that sensibly scales itself to the size of the display window! I experimented with ratios and proportions until I was both satisfied with the composition and the function scaled the composition as expected with its scale argument. For future work, I'd like to be able to isolate the background animation and the rocket drawing to remove the strobing of the random star generation within the background.   
  
     With Red Remove, the biggest issue I encountered was the nature of JS referencing the original object instead of assigning a copy of an object. While there were many ways I tried to troubleshoot the issue of why modifications to the outer circle's color affected the inner circle, even before the modification call actually occurred, ultimately it related to the aforementioned. Nevertheless, I rudimentarily converted the p5.js color object to a string, modified the string, and then parsed the string as a color once again.   
  
     The final exercise for Polar Points was relatively easy as Professor Faas already outlined the algorithm and the exact steps in the exercise description. However, I'm still confused by what the aesthetic or mathematical purpose of the exercise is. I played around with it and also built a rectilinear to polar conversion function just for the sake of a computation demonstration, but ultimately it was not the most useful exercise. Though, I will employ the idea of using vectors to carry coordinates in future exercises.