For this project I really wanted to combine some of the patterns we used in lecture. In particular I wanted to make a pattern using an ascending color scheme.

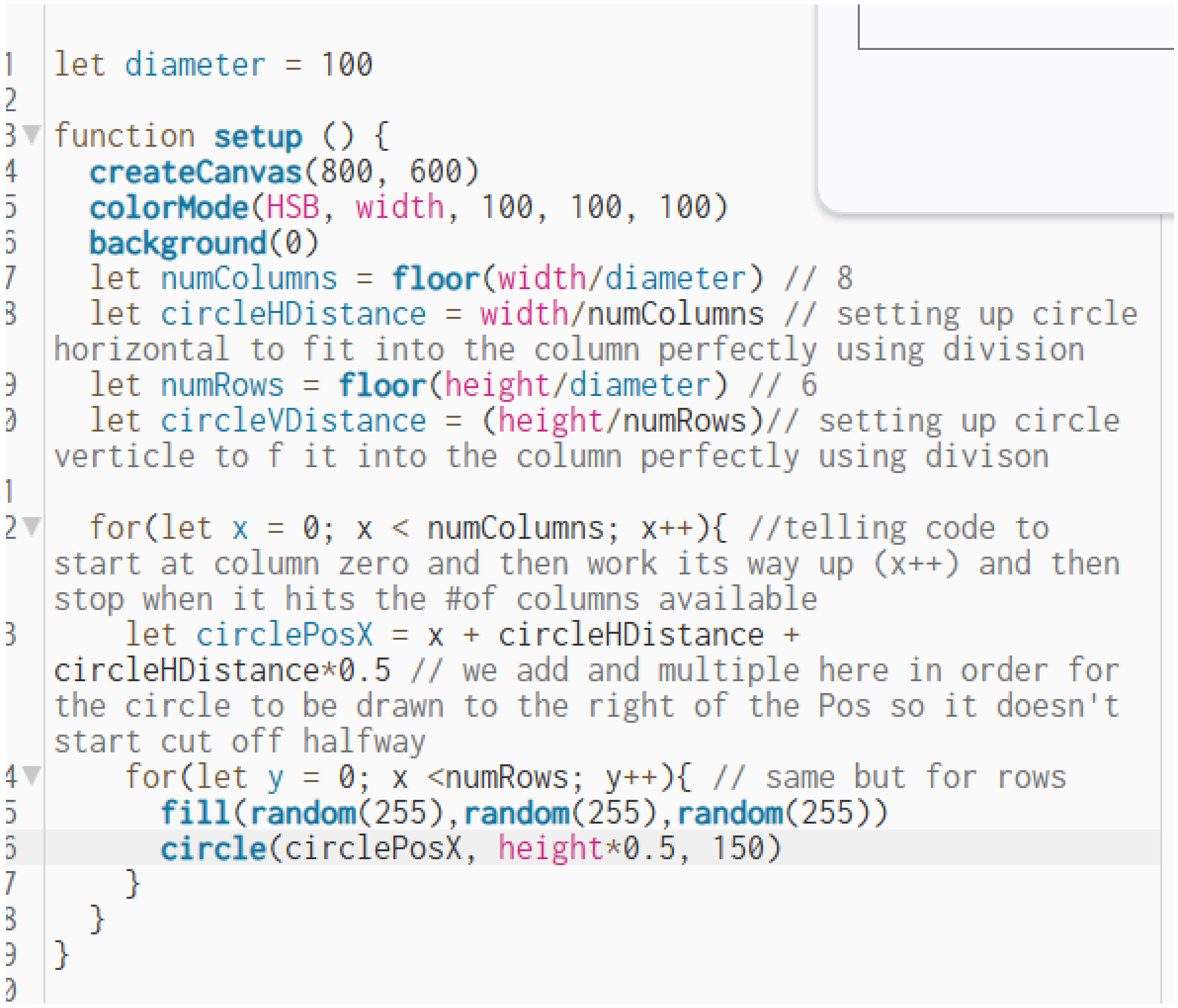
My first problem however was trying to interpret the code and figuring out how I could combine these two projects into one functioning code.

Reviewed the zoom lecture to understand how to write the code to make ascending colors based on the width of the canvas.

Problem: tried to combine code… ended up deleting my circle instead of actually making it work correctly.

Started over to get a better understanding of the functions for each segment of code, started by setting up rows, columns and color mode.

Wrote down notes from lecture to understand the code better:

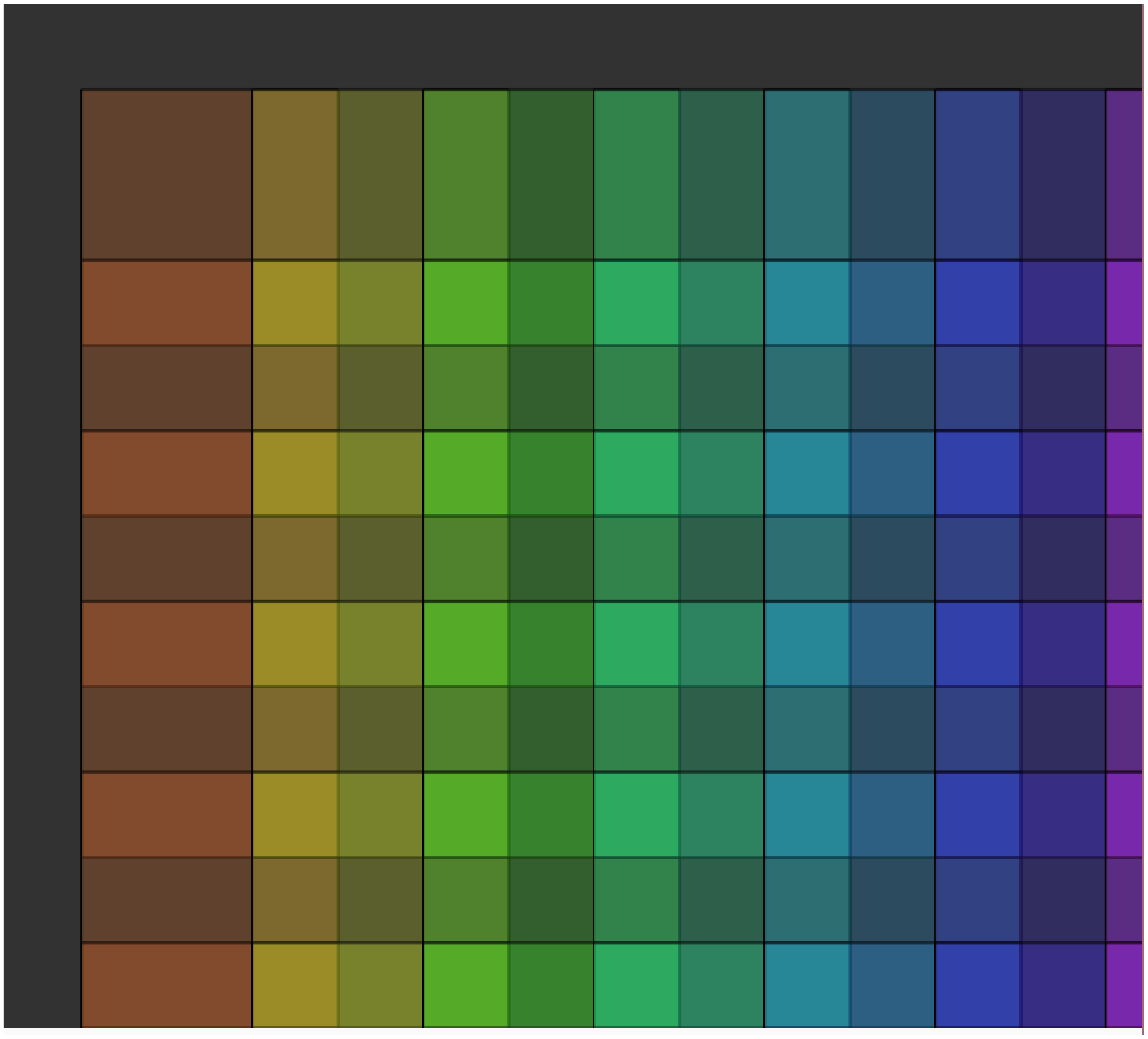


```
let diameter = 100

function setup () {
  createCanvas(800, 600)
  colorMode(HSB, width, 100, 100, 100)
  background(0)
  let numColumns = floor(width/diameter) // 8
  let circleHDistance = width/numColumns // setting up circle horizontal to fit into the column perfectly using division
  let numRows = floor(height/diameter) // 6
  let circleVDistance = (height/numRows)// setting up circle verticle to f it into the column perfectly using divison

  for(let x = 0; x < numColumns; x++){ //telling code to start at column zero and then work its way up (x++) and then stop when it hits the #of columns available
    let circlePosX = x + circleHDistance + circleHDistance*0.5 // we add and multiple here in order for the circle to be drawn to the right of the Pos so it doesn't start cut off halfway
    for(let y = 0; x <numRows; y++){ // same but for rows
      fill(random(255),random(255),random(255))
      circle(circlePosX, height*0.5, 150)
    }
  }
}
```

Problem: p5js crashed when I was working and I hadn't saved…

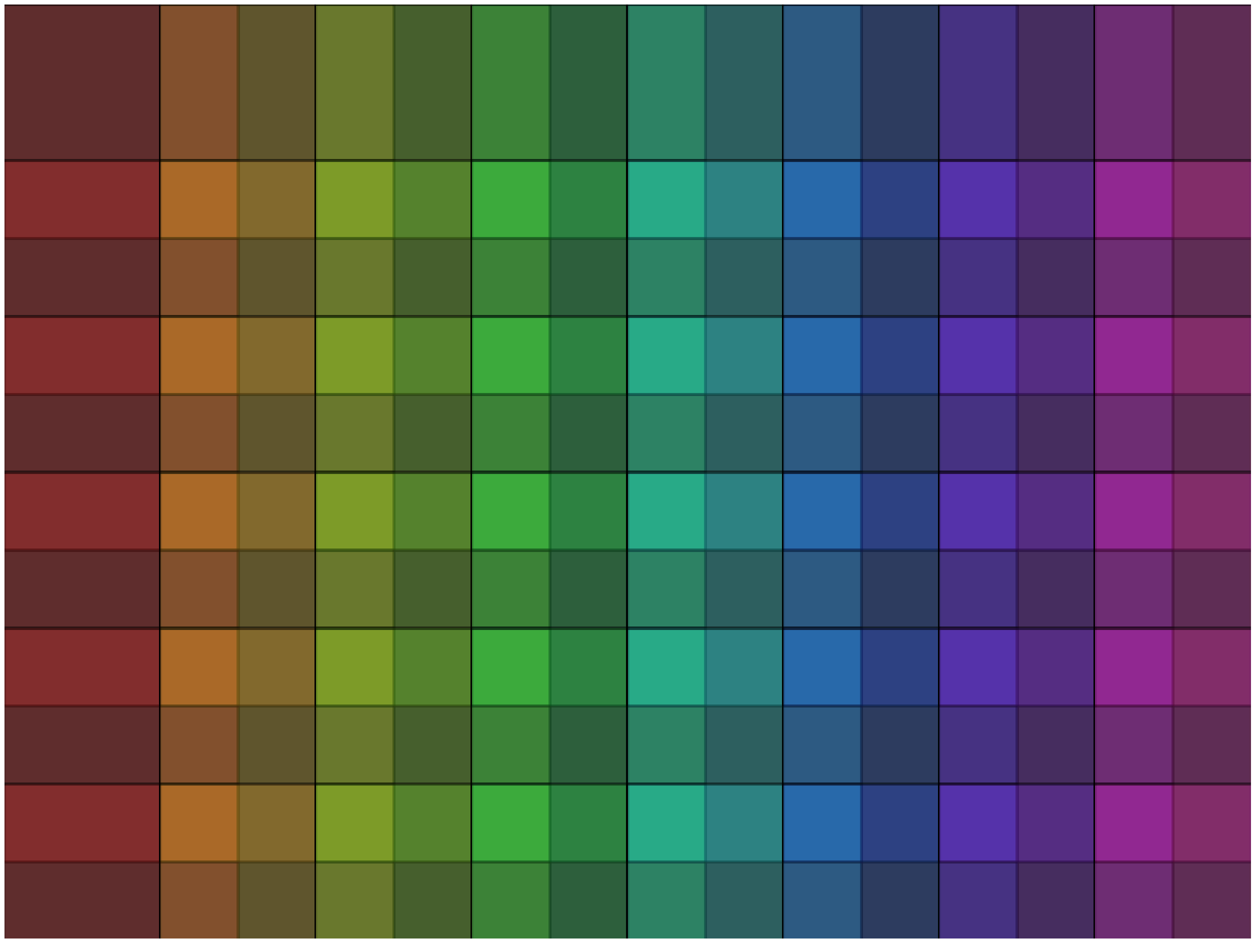
Started over again, this time utilizing code from one of my other saved projects.

Began looking at creating an overlapping square pattern with some opacity which I thought looked pretty cool.



Problem: the squares are not aligned properly and need to be shifted up and left, also the edges are very long for some reason?

Managed to fix the alignment issue by rewording some of the code to not give it as much space on the sides, still have an issue with the sides being rather long, perhaps I could shift it further up and left to compensate?



Found that by changing the diameter I could manipulate how many squares lined up, by changing it to a value of “2” I could make everything line up correctly.

Change the blend mode to “Lightest” to remove stroke and make a cleaner look. This interestingly also changed how the squares are arranged based upon the diameter value since I was using it as one of my values for the rectangles scale. This also gave me an idea.

Could I make it so the pattern changes based on user input through manipulating the diameter per mouse click? I think this would look rather interesting.

In order to do this I need to do a few things, first I need to set up a value where I can multiple my last size variable by the diameter and then a random number. For this purpose I chose to do a number between 1-10, as once you hit 10 things started to get a little odd. For this purpose I wrote a simple line of code:

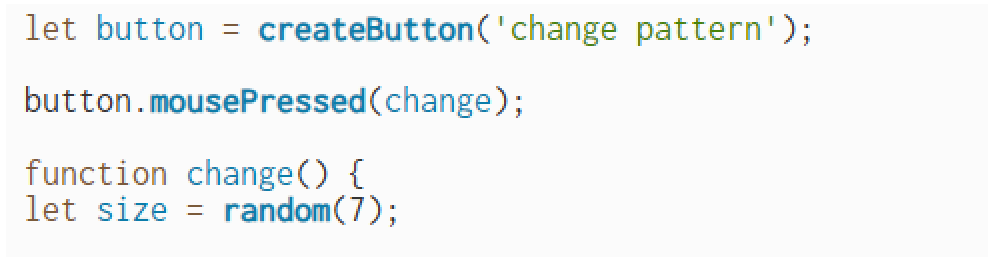
```
Size = random(10)
```

This would make it so I could have a random number generated between 1-10, I then used that for my rectangle size calculation by using rect(rectPosX, rectPosY, diameter*size) and it worked rather well! I got a different and interesting looking pattern for each time I re-ran the code.

Now that the function works, I want to add a button to make it so that users can change the color without having the re-run the code every time.

Managed to make a button that changes the pattern… but not in the way I want it to. It currently seems to multiple the current brightness of the pattern and eventually washes it out.

Here is the code I've used.

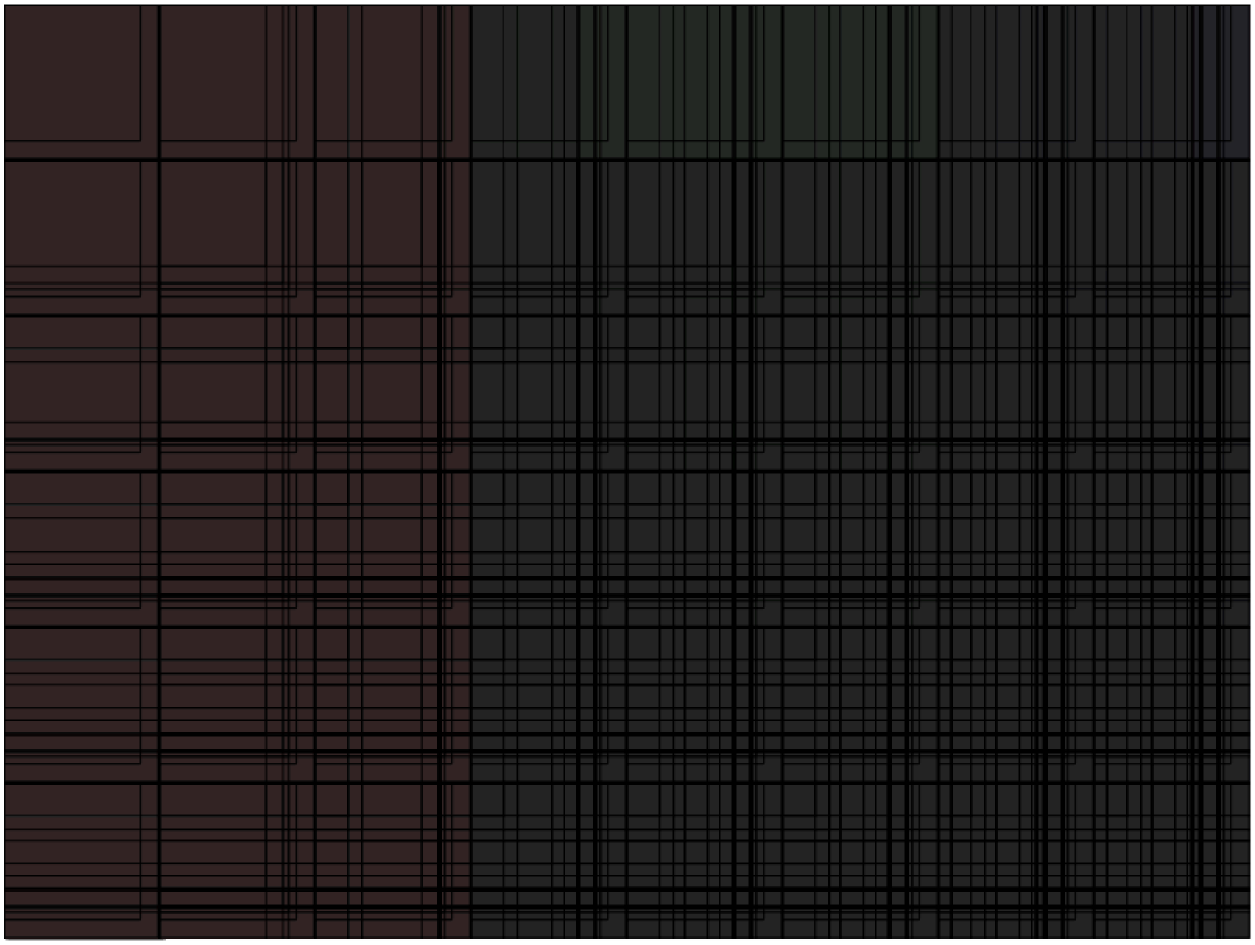


```
let button = createButton('change pattern');

button.mousePressed(change);

function change() {
let size = random(7);
```

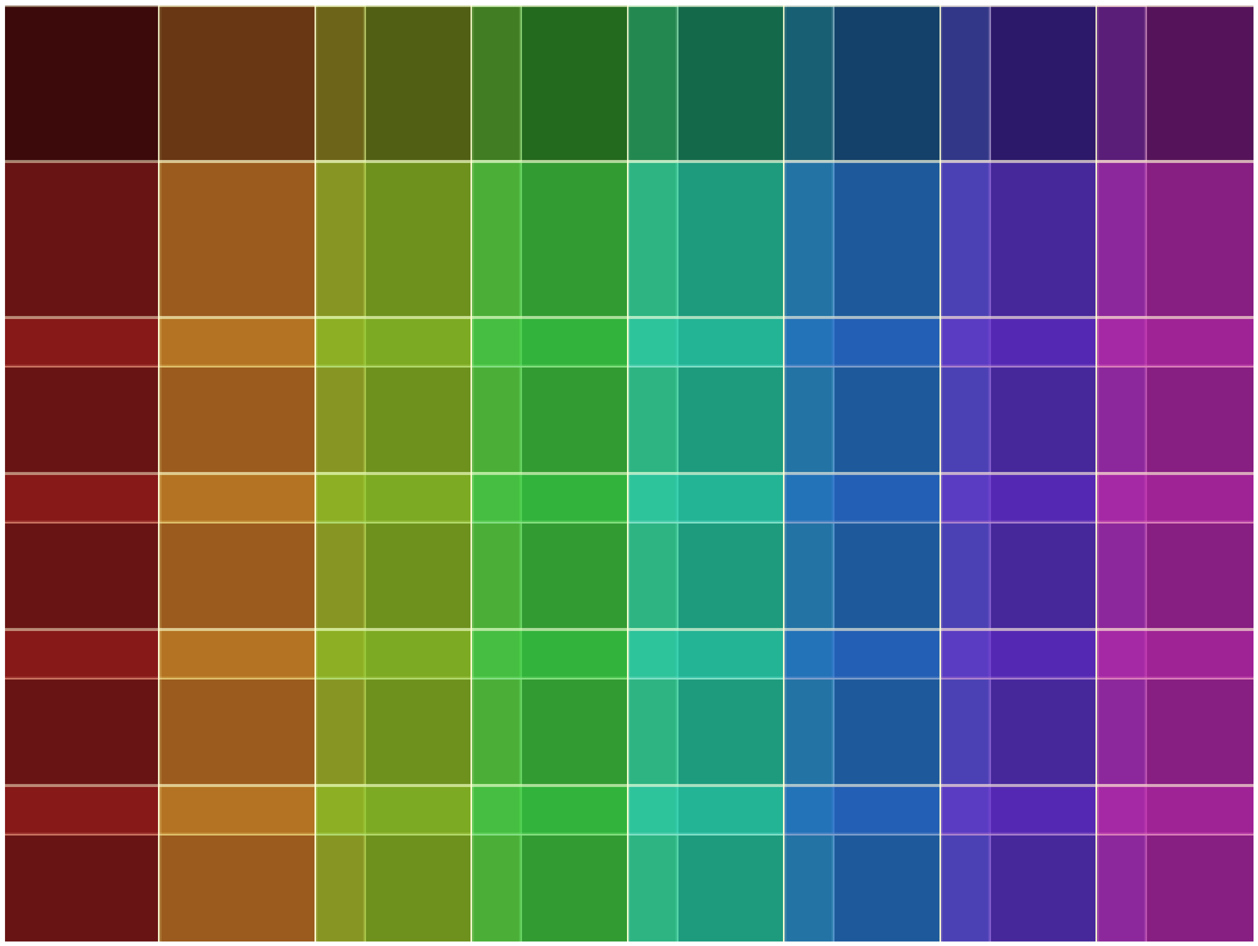
I did a temporary fix by changing the blend mode from light to normal.

Which seems to have prevented the problem from occurring… still not entirely sure why though.

Changing the blend mode to darkest actually makes a very interesting pattern where you slowly add lines and make it more and more complex. I think it looks really cool and might end up keeping this.



I’ve changed my mind like 3 times now, really liking how random and cool these patterns can be. I’ve now randomized the stroke colors for each button press as well which allows for some really cool patterns with lots of color to be made like this one below.



So what was my most interesting problem?

To be honest, a lot of my time was spent just trying to figure out both what I could do in terms of manipulating the pattern and what I wanted to go with. I knew I wanted some level of randomness included that could be utilized by whoever is interfacing with the program. However, the question for me was, how could I get something that would generate a random pattern but not look like a complete mess? There is a huge difference between randomly generating a pattern and just randomly generating a mess of colors and shapes.

I figured out the solution to this problem by extensively reviewing the previous lecture and dissecting what made the patterns created in class work. From there I looked up a few different commands on p5js, including how to code buttons, random sizes, and manipulate stroke. Then it was largely down to making sure everything worked together properly. The right things were randomized, the colors were decent, and the patterns looked like patterns. This was largely determined by trail and error, I'd run the code, adjust a few values, and eventually landed on what I thought was a decent solution.

Some of the cool patterns that were generated: