RNG Art

The artistic beauty (or artistic train wreck) of pseudorandomness

JOSH BICKING

Abstract Art

- Abstract art gets a lot of flak from various people, mostly those who rarely understand the value of art at all.
- To them, as well as several others, it simply looks like paint splattered randomly on a canvas. I used to think this myself, before I spent some time looking at an abstract painting.
- This accusation got me thinking. Would it be possible to generate art using a pseudorandom number generator and a graphic design application?

The Generation: Preparations

- Adobe Photoshop CS5 includes support for JavaScript, AppleScript, and VBScript. While these are all mostly the same (apart from some syntax variations and cross-platform restrictions), I chose to write my generator in JavaScript.
- JavaScript includes a function called Math.random(). In short, this function returns a long floating-point (decimal) number in the range [0, 1). This can be manipulated through multiplication and addition to output any range of numbers. While Math.random() is not a cryptographically secure random number generator, it's much faster than cryptographically secure number generators. Considering all the calculations that needed to be done, it was the practical choice.

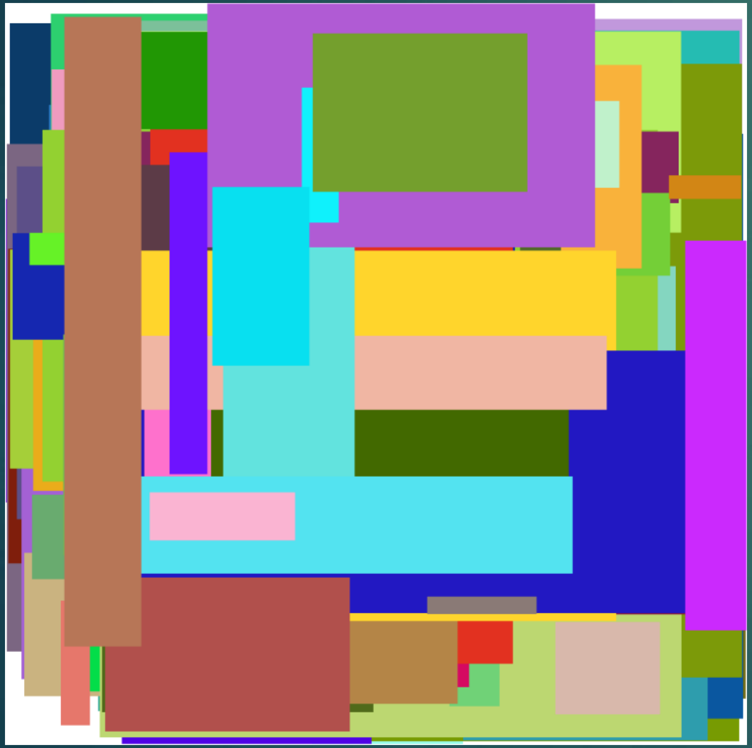
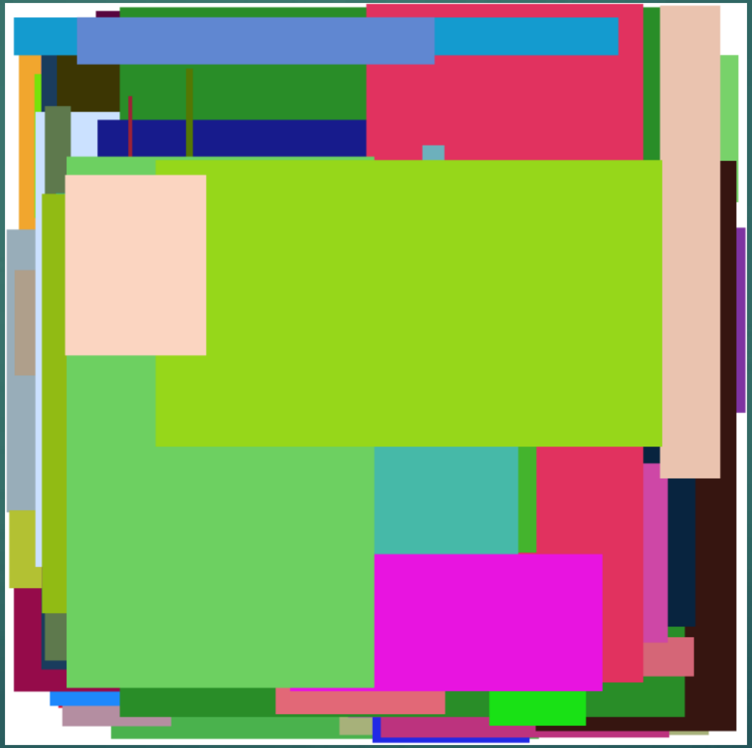
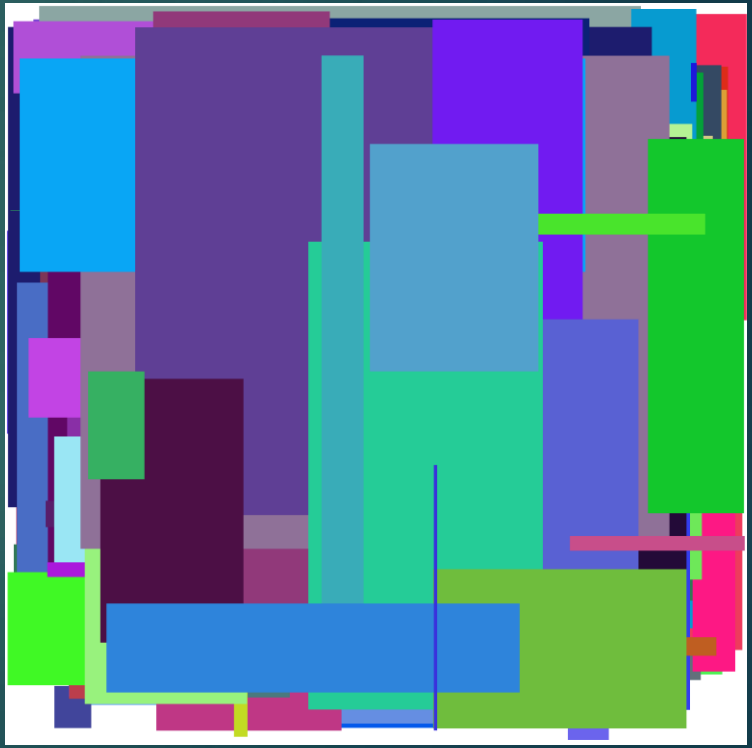
The Generation: Preparations

- I started with a white image sized 4000 pixels by 4000 pixels. From there, I would draw rectangles of various dimensions, orientations, and colors at random places across the image. I would repeat this process several times, and decide whether or not any of the images had any sort of visual appeal to me. Even if it didn’t, I would keep them for comparison purposes, and also to see if it had any visual appeal to anyone besides me. So to whoever views this, thank you for your eyes.

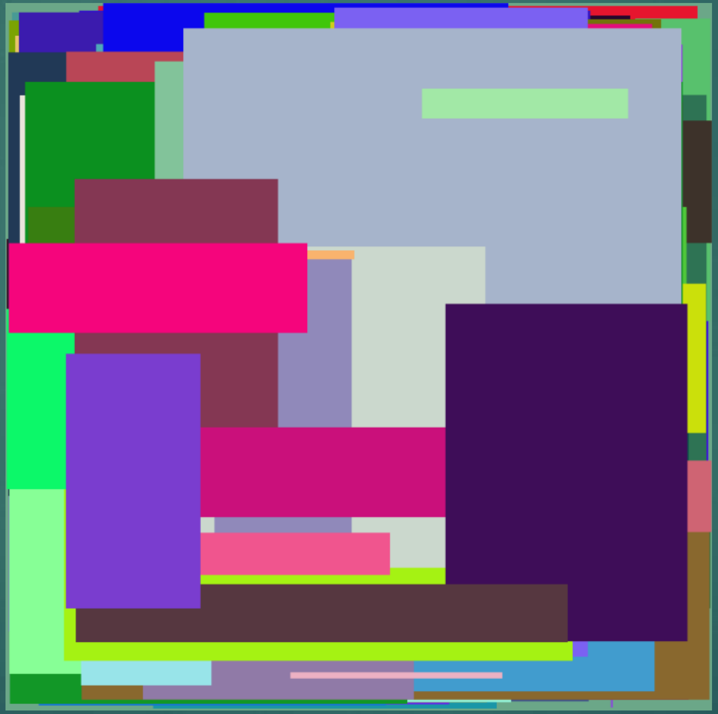
The Generation: Test One – The Generation-ing

- In short, the generation script performed the following tasks a specific number of times:
  - Choose a color between #000000 (black) and #FFFFFF (white)
  - Choose four numbers between 0 and 4000. These will be (x_0,y_0) and (x_1,y_1), the top left and bottom right points of the rectangle.
  - Create a new layer of the image.
  - Select a rectangular region specified by the two pairs of coordinates chosen.
  - Fill the region with the chosen color.

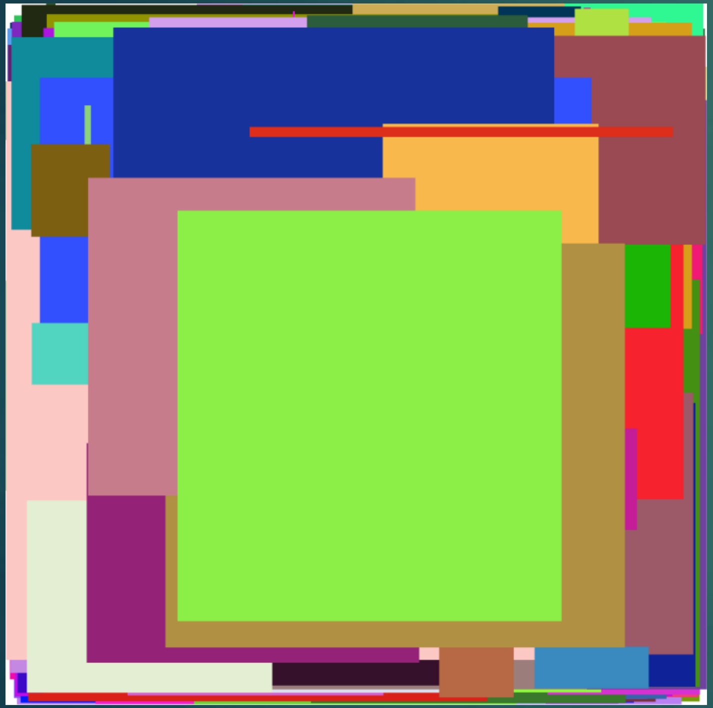
The Generation: Test One – The Results – 200 rectangles

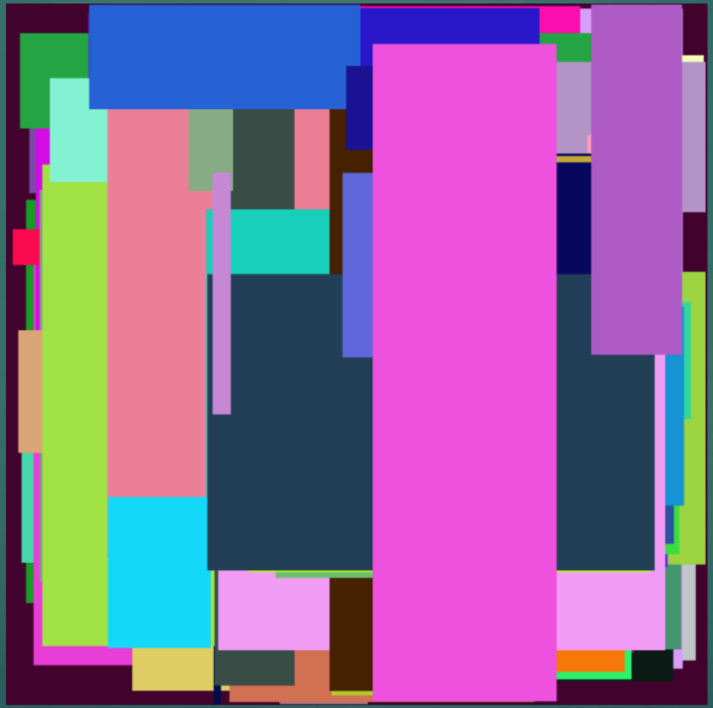




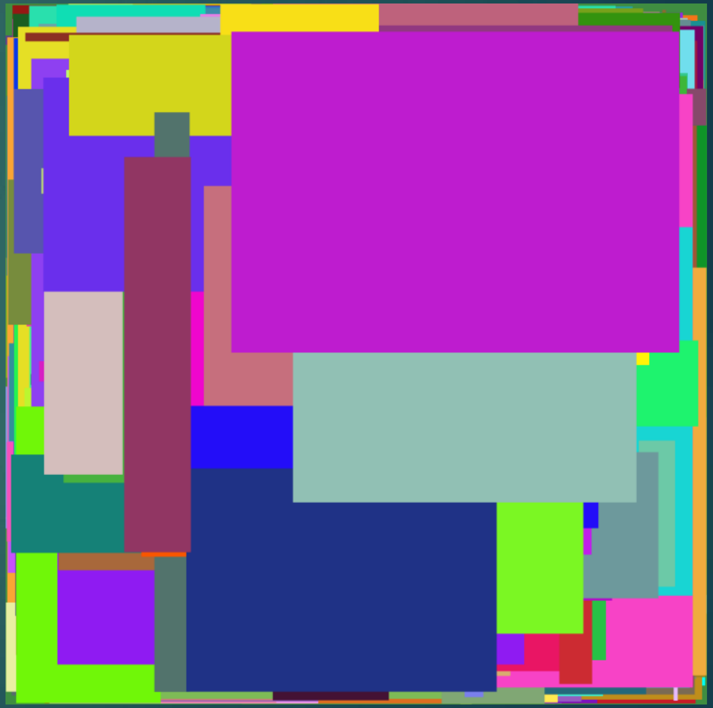
The Generation: Test One – The Results – 1000 rectangles



The Generation: Test One – The Results – 2000 rectangles



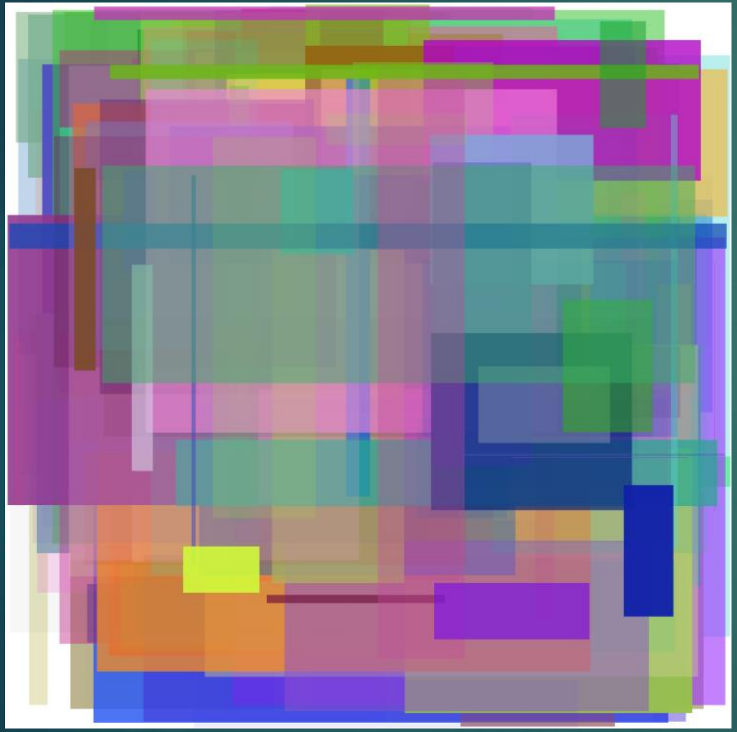
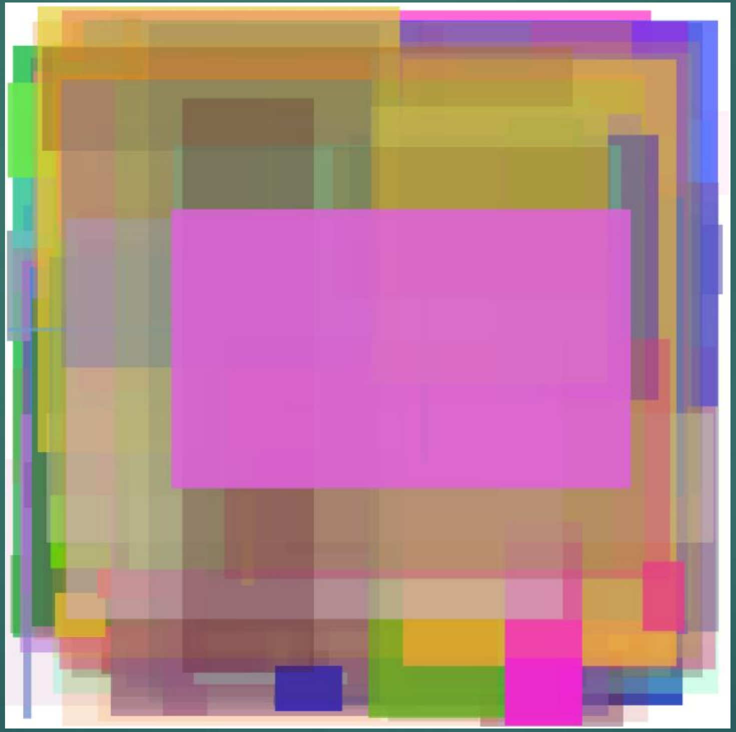
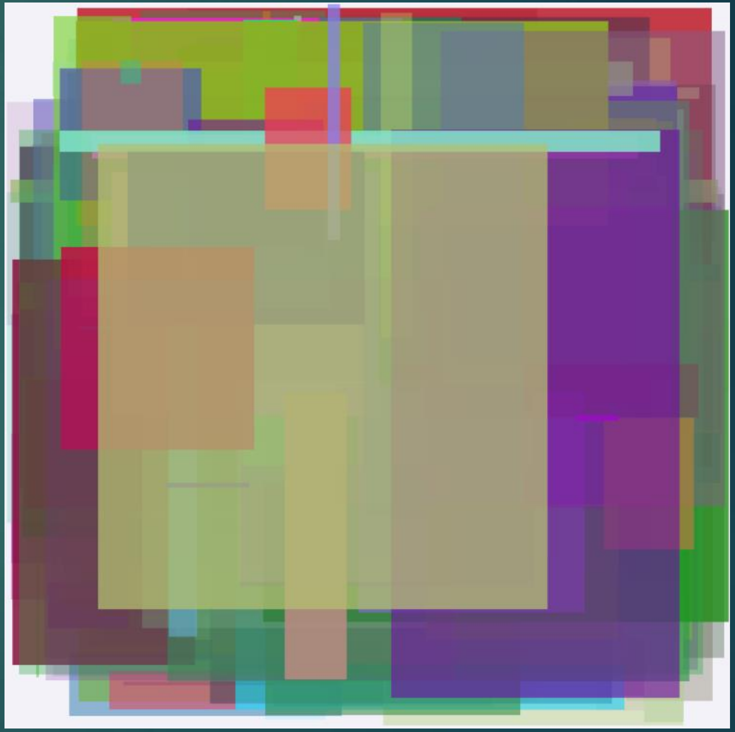




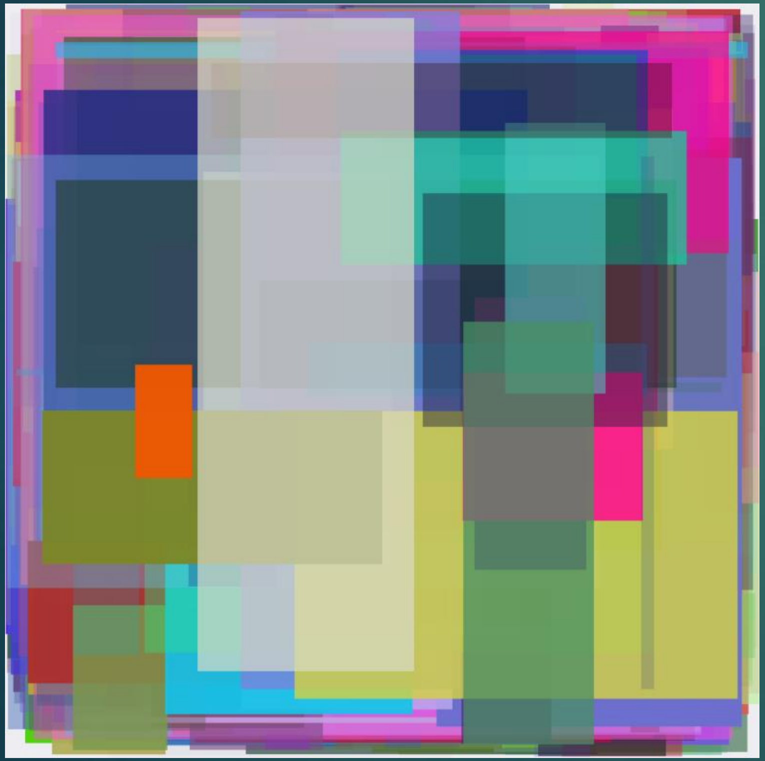
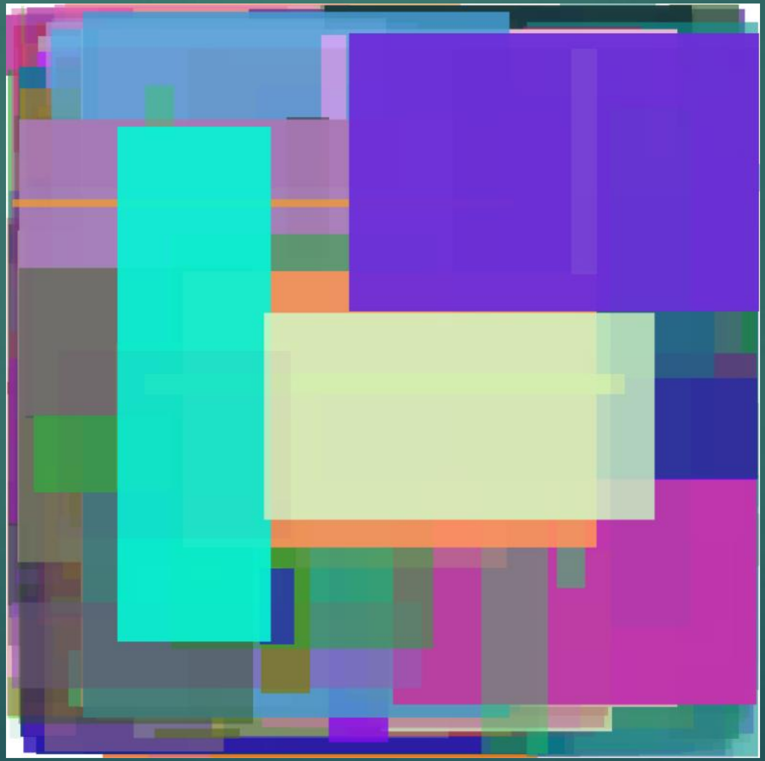
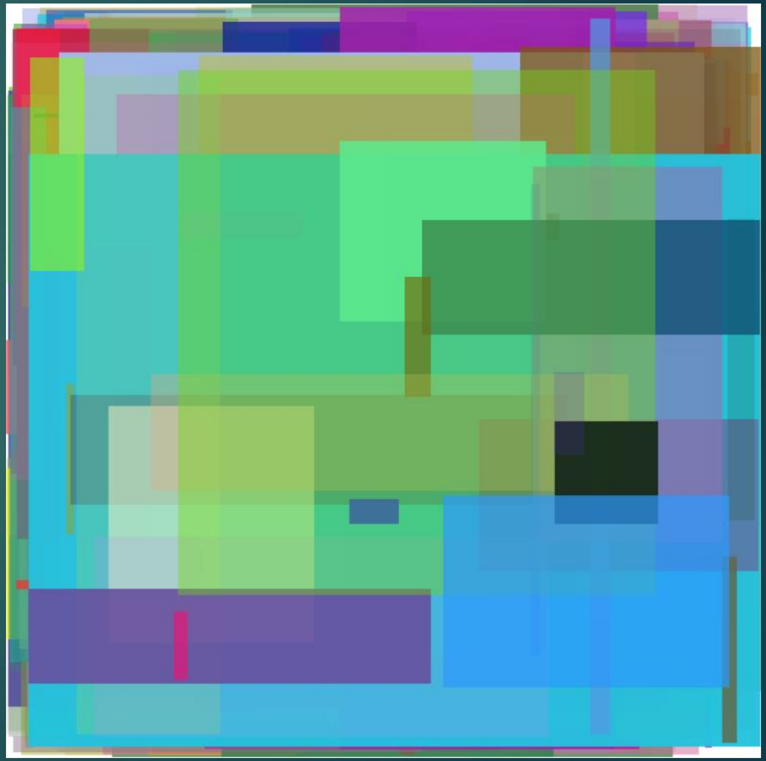
The Generation: Test Two – Less Covering

- I decided to mess with the opacity next. The second script performed the same actions as the first, except now, at the end of generating each rectangle, it chose a number between 0 and 100 for the opacity of the rectangle.

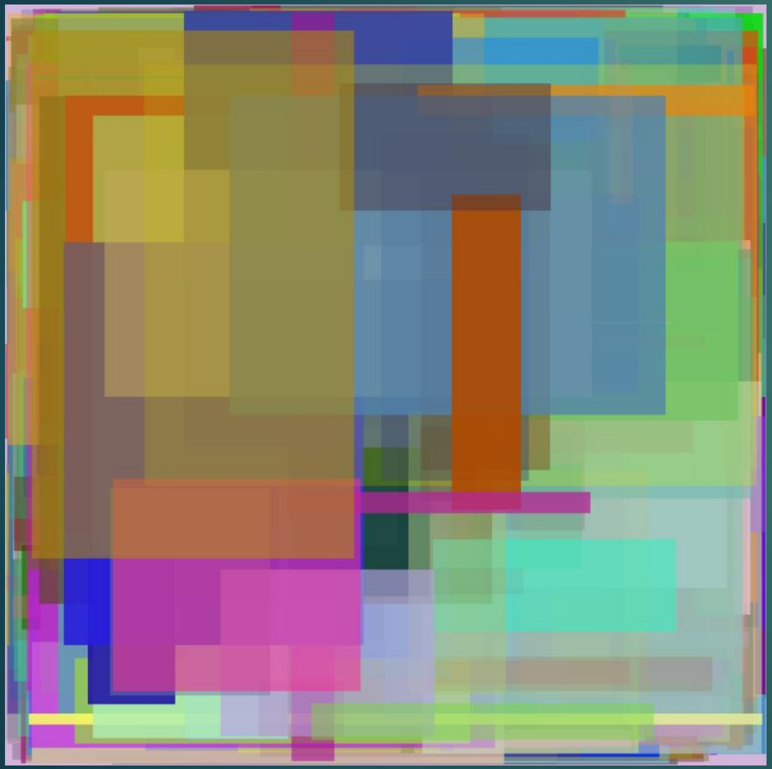
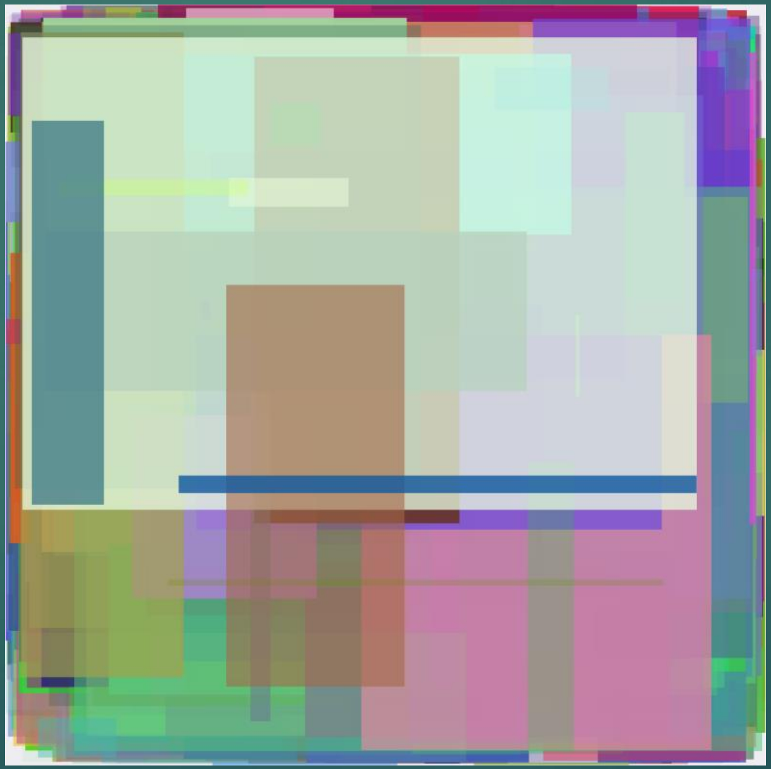
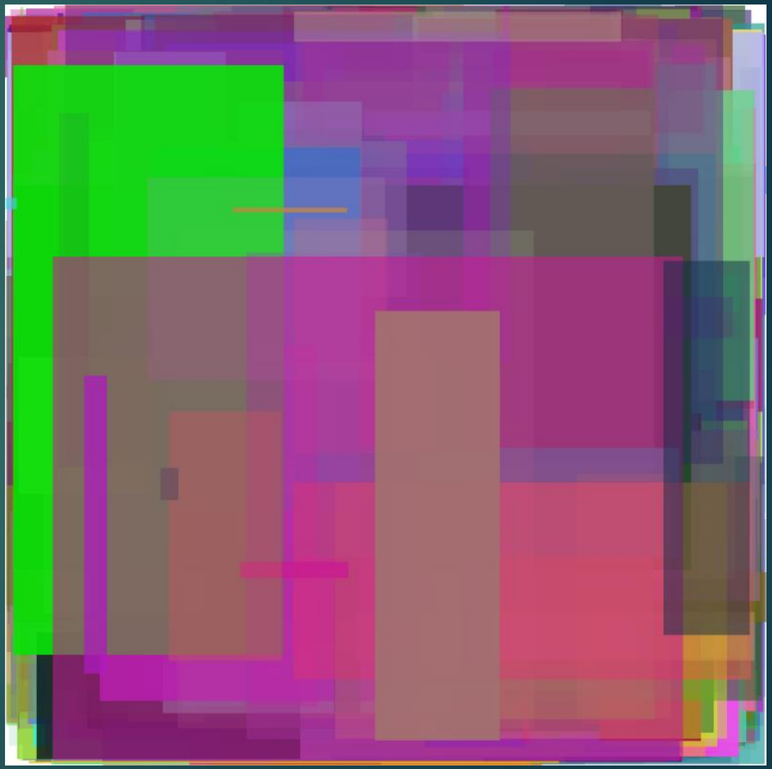
The Generation: Test Two – The Results – 200 rectangles

The Generation: Test Two – The Results – 1000 rectangles

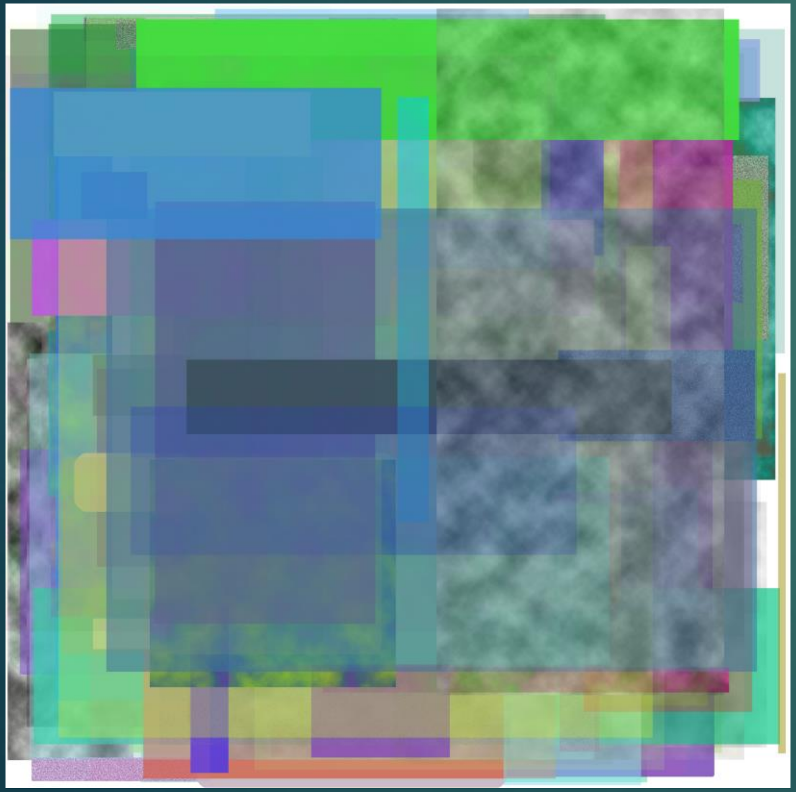
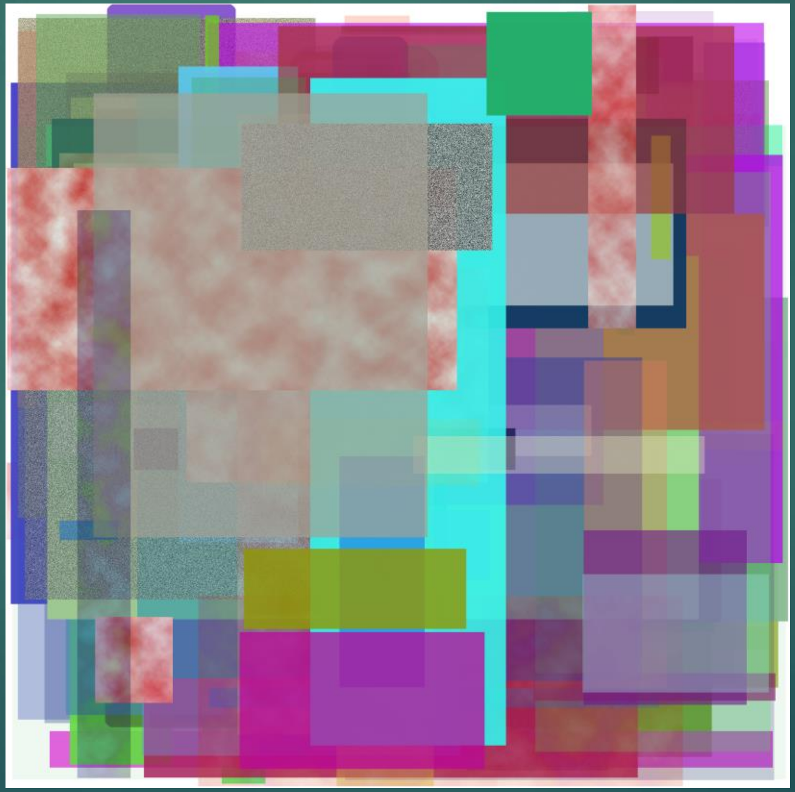
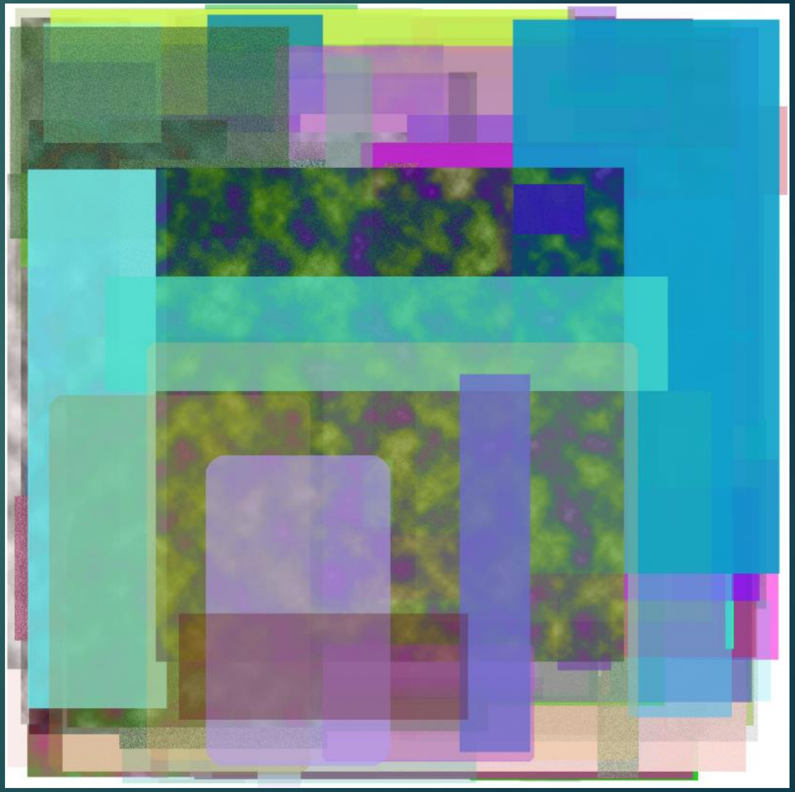
The Generation: Test Two – The Results – 2000 rectangles

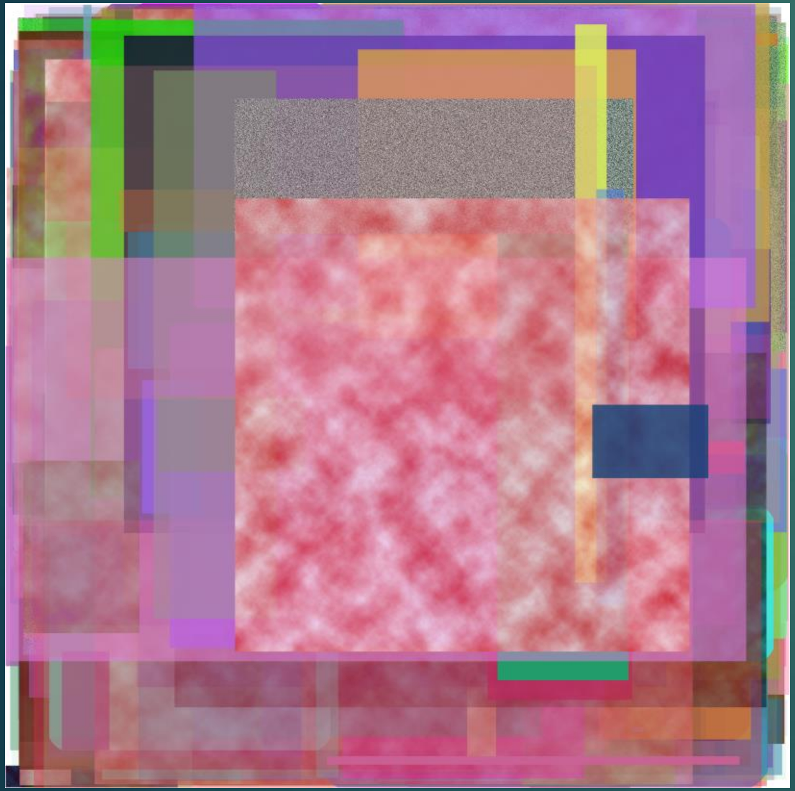
The Generation: Test Three – Filters

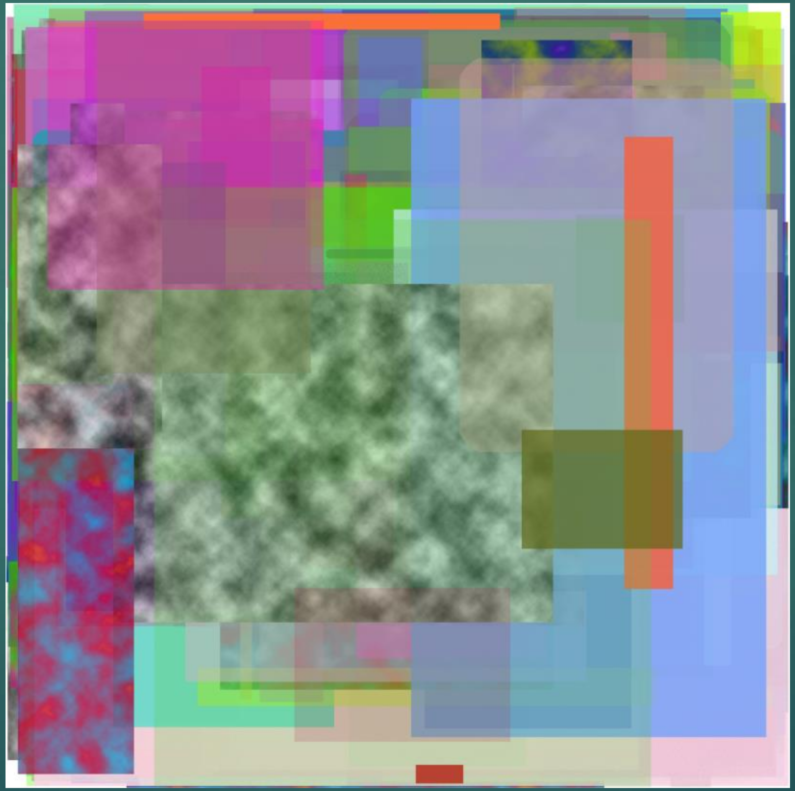
- While I thought the transparency added a new sense of depth to the image, I still wasn’t satisfied with how flat and bland each rectangle felt. For my third script, I chose 13 different filters that could be applicable to a solid colored rectangle, and the script chose a number between 0 and 13 to apply a filter to the rectangle. (0 meant no filter was added.)

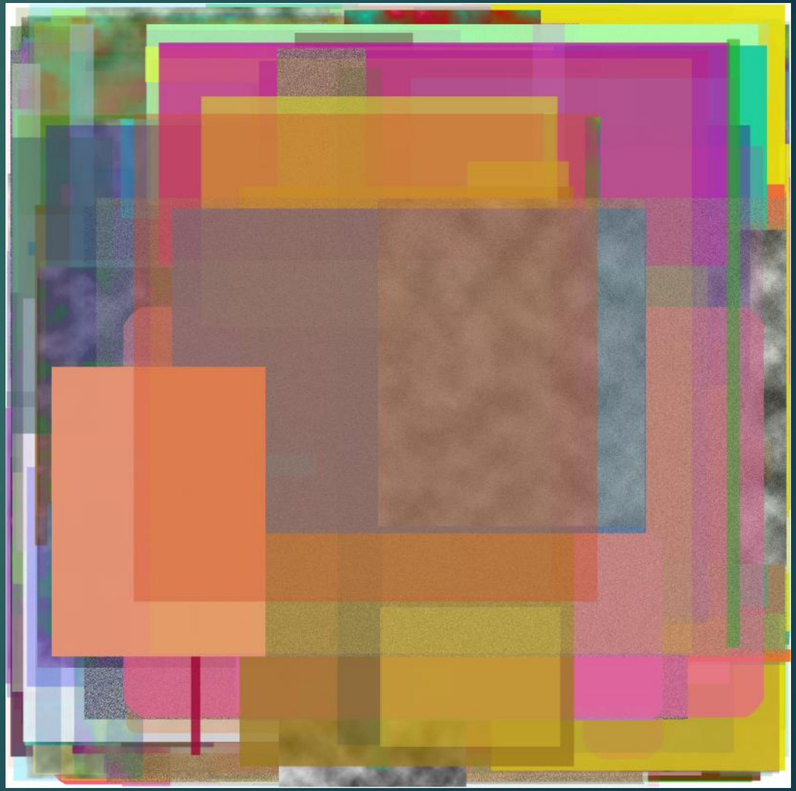
The Generation: Test Three – The Results – 200 rectangles

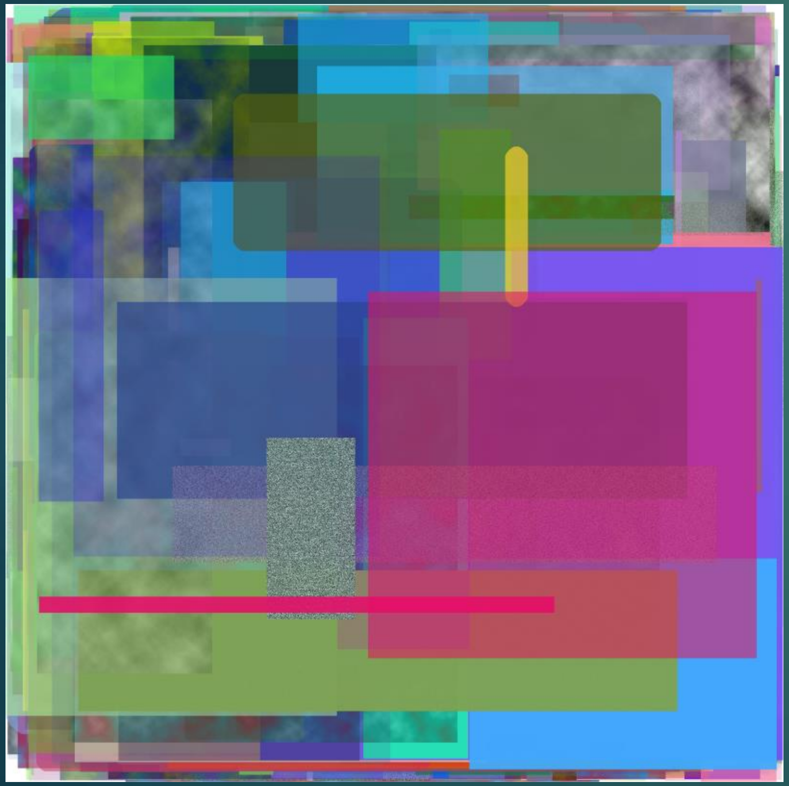
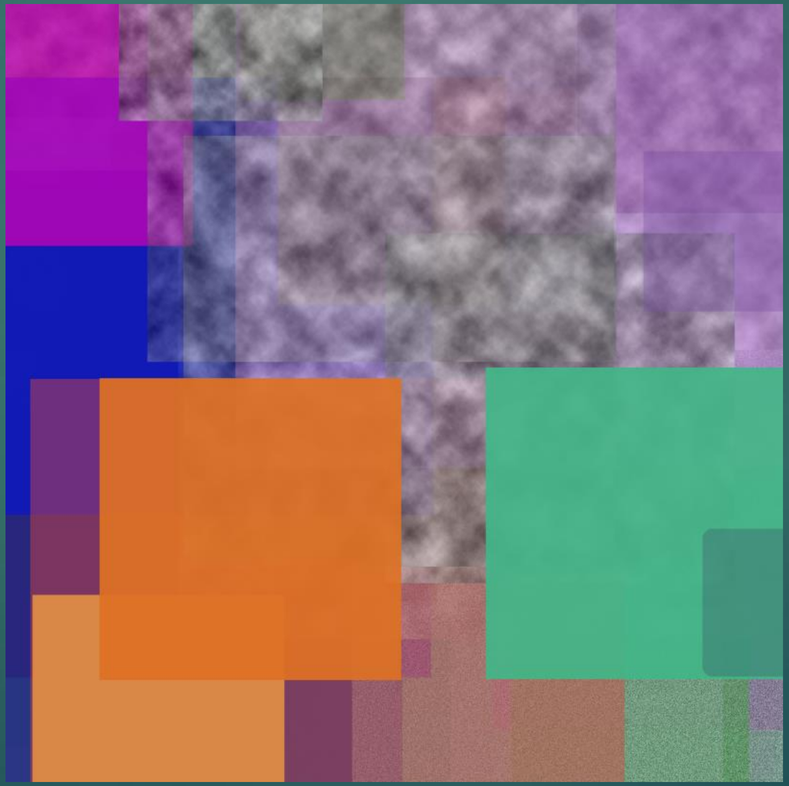
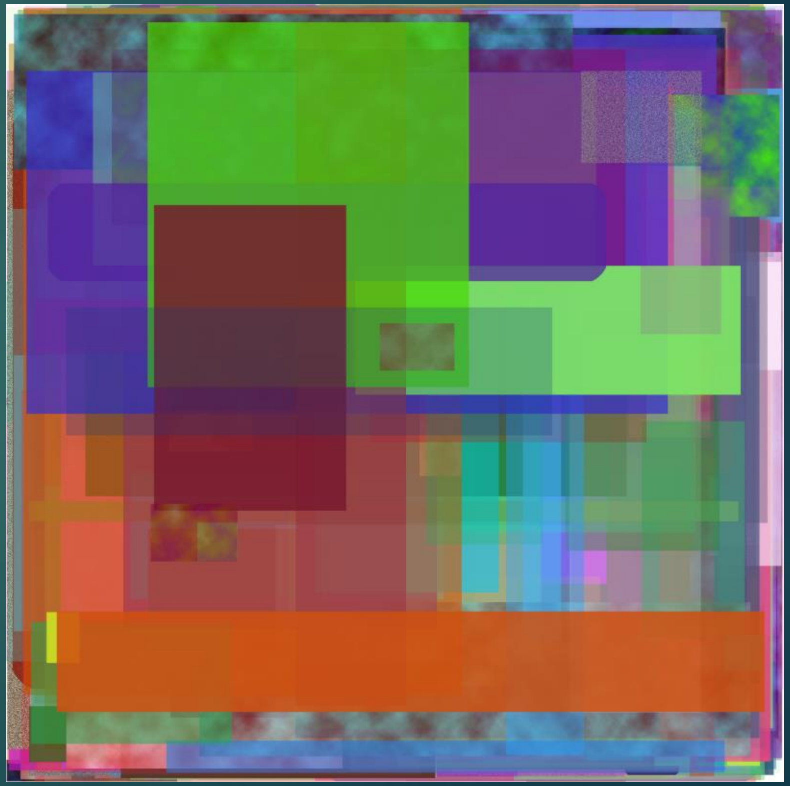
The Generation: Test Three – The Results – 1000 rectangles







The Generation: Test Three – The Results – 2000 rectangles

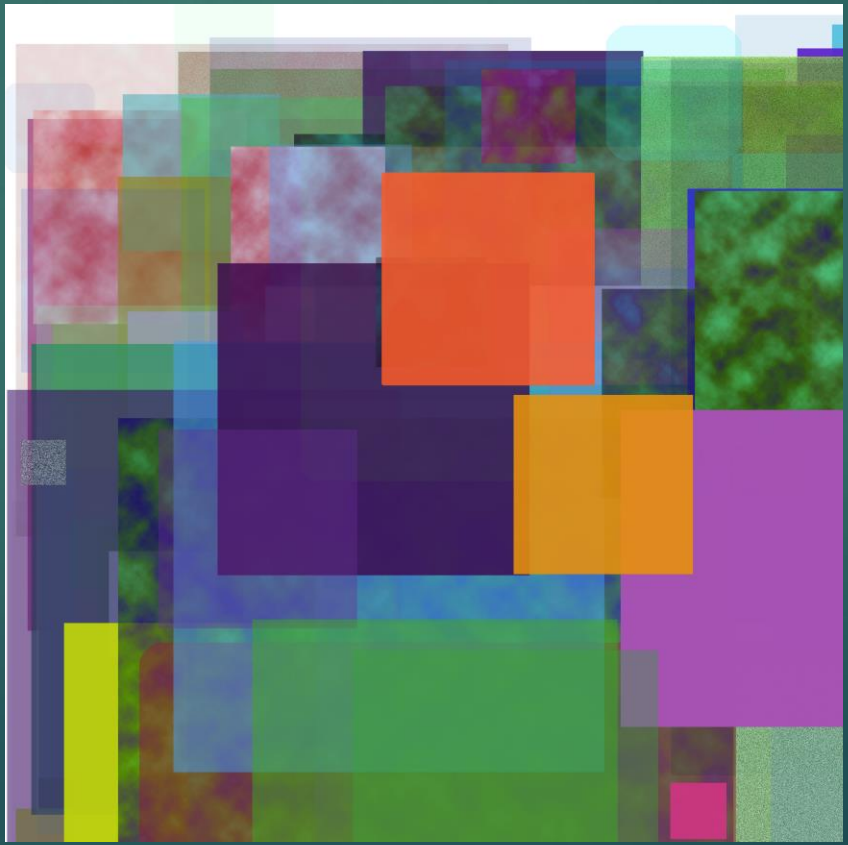




The Generation: Test Four – Squares

- At this point, I wondered if these pictures could use a little more structure. I decided to try squares instead of rectangles.
- Instead of choosing two points for a rectangle, the fourth script will:
  - Choose two numbers between 0 and 4000 to be the location of the upper left of the square.
  - Choose a number between 0 and 4000 for the length of the square.

The Generation: Test Four – The Fatal Error

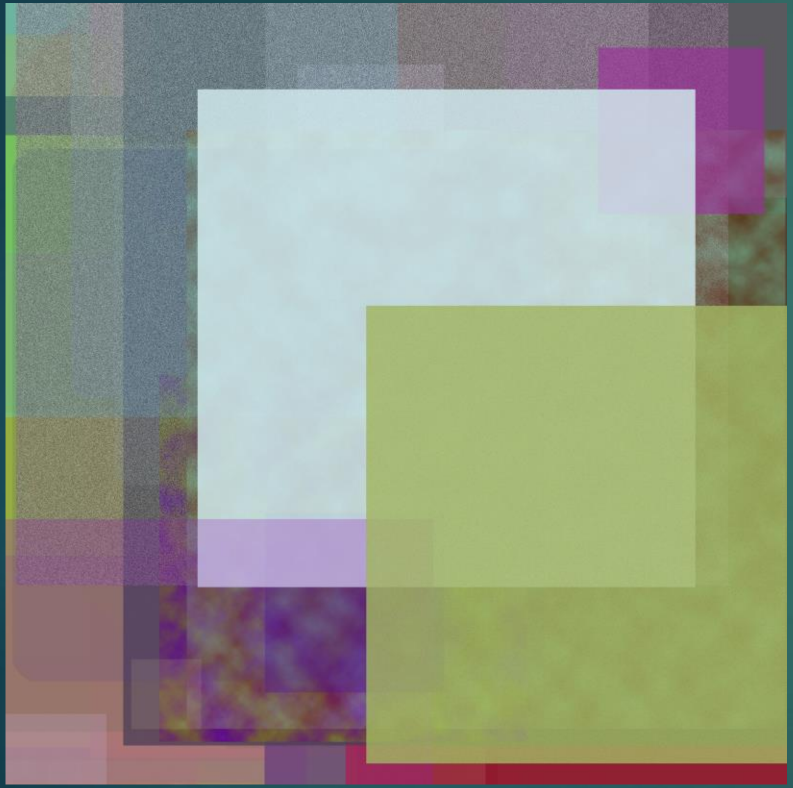
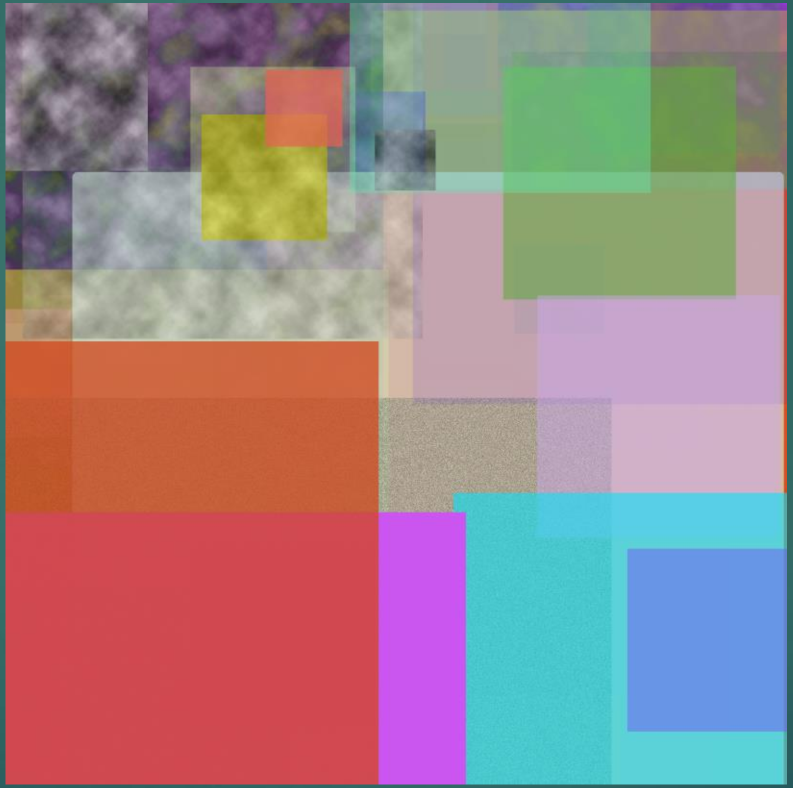
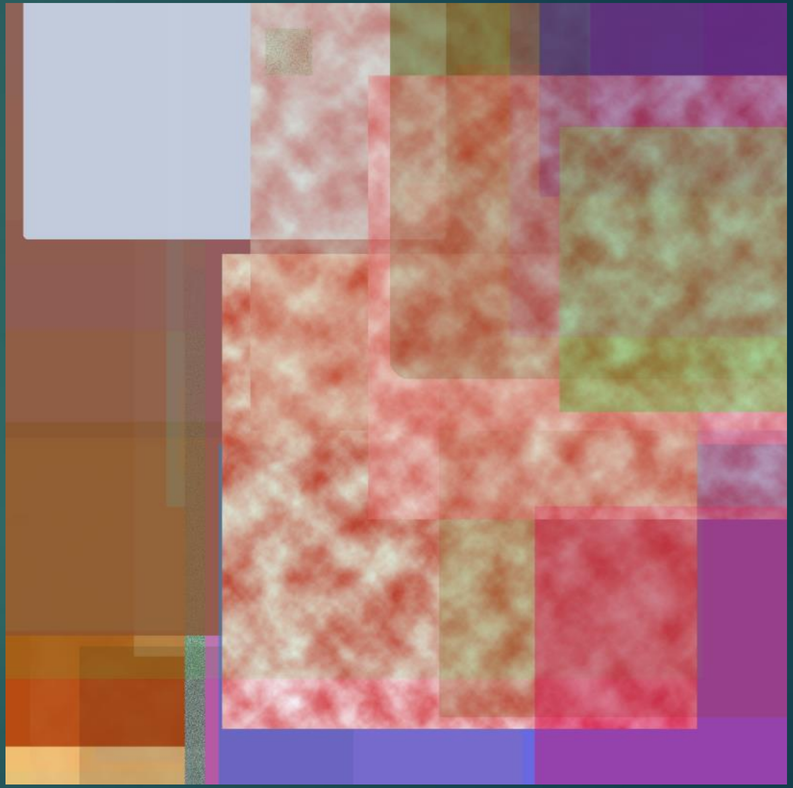
- After generating one 200 square picture, I could see that starting the square from the upper left leaves the top and left of the image blank.



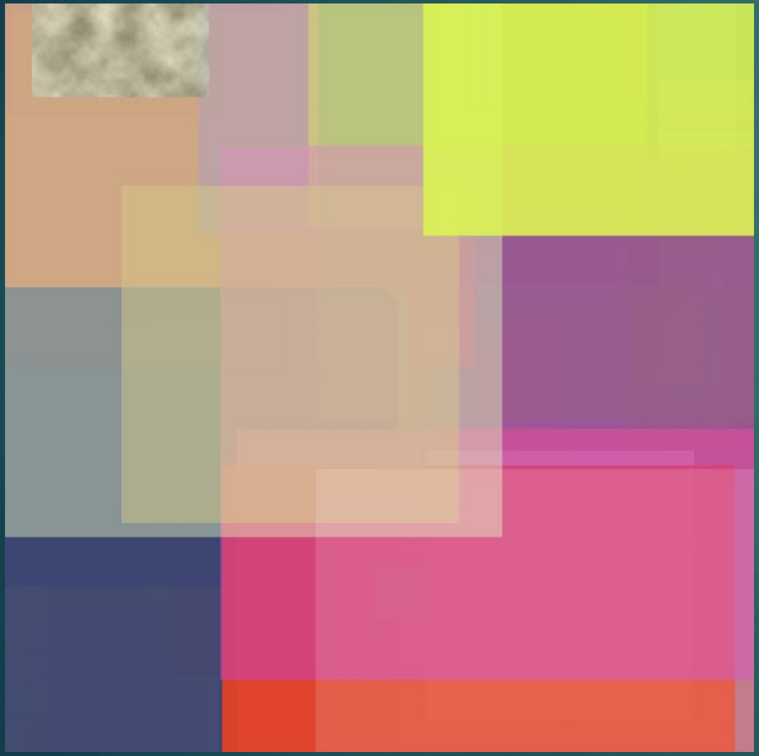
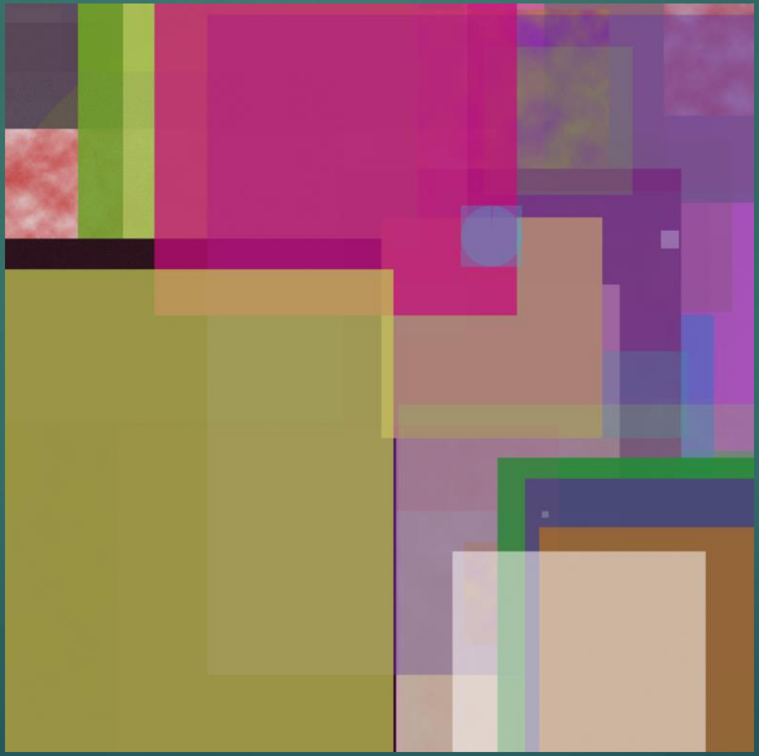
The Generation: Test 4.5 – More Squares

- This time, instead of using the chosen point as the upper left point, I used it as the middle of the square.
- I did this by modifying the chosen point. After it was chosen, subtracting half of the square’s length to both the x and y coordinates moved the center of the square to the chosen point.
- The results were much more balanced.

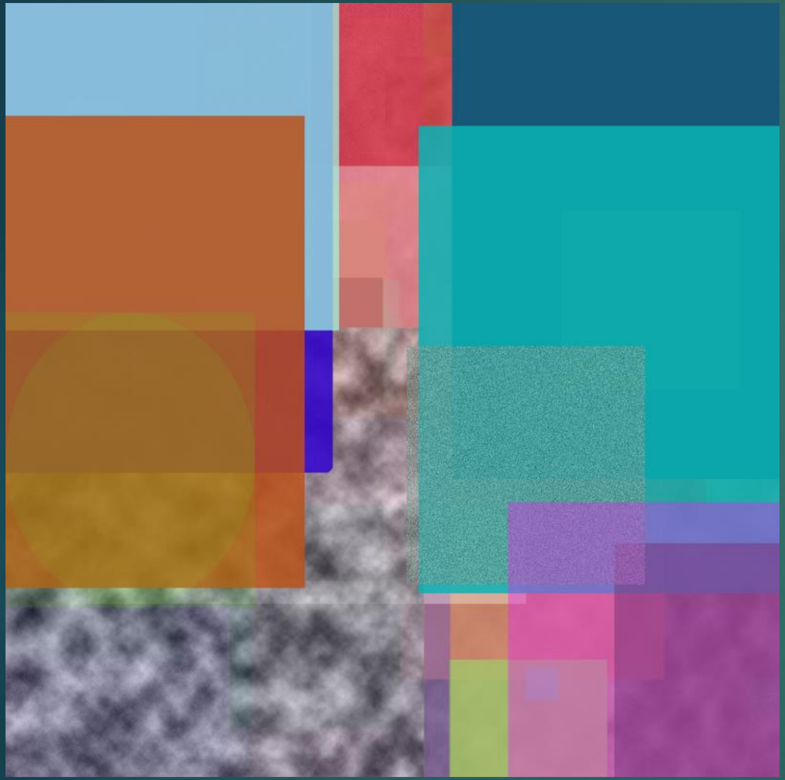
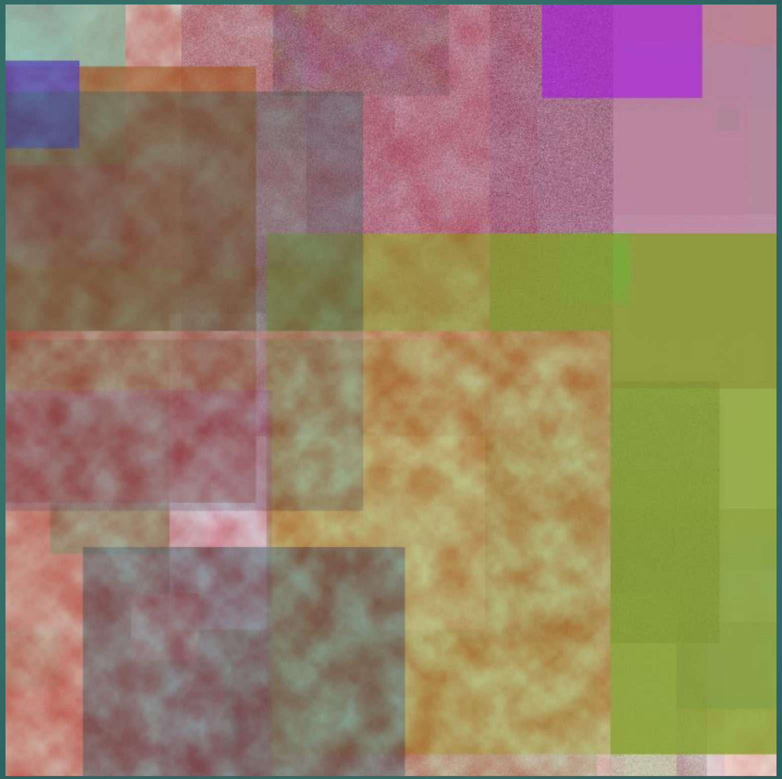
The Generation: Test 4.5 – The Results – 200 squares

The Generation: Test 4.5 – The Results – 1000 squares

The Generation: Test 4.5 – The Results- 2000 squares

- Due to time restraints, I was only able to generate two of those. They took a while.

Well, is it art?

- I’m not sure, really. Certain pieces have a nice visual balance to them, Others I can stare at for hours.
- I don’t really get anything out of it. They look nice, but they don’t send a message. Is that because I didn’t put sweat and blood into conveying a message to an audience or filling a void within myself?
- Or did I do that by writing the script? Some consider coding to be an art form, as some programs have an artistic balance and beauty to them.
- I certainly enjoyed making them and seeing how they came out, but whether or not it’s art, I don’t know. You decide.