Intro to Generative Art
Problem Set 3

Harvard University
Taught by Sudhan Chitgopkar

This problem set is optional, but *highly encouraged*. You may work in groups and/or solicit outside help including but not limited to Generative AI tools. All outside sources/code used *must* be properly credited in the code documentation submitted.

1 Remedial Chaos Theory

In the process of developing (and playing) the Chaos game, we introduced restrictions to produce different fractals while keeping all other elements of the game constant. The application of these restrictions gives us the restricted chaos game. *Using either known restrictions or by creating your own, generate a fractal through the restricted chaos game.*

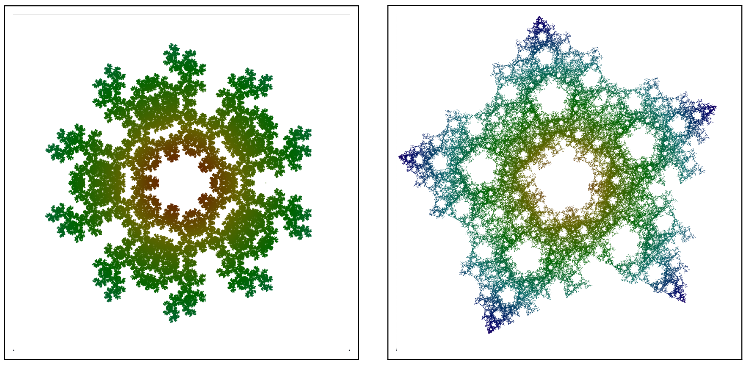

Figure 1: Two versions of the restricted chaos game played in a pentagon

2 Snow (Hey Oh)

Though we visualized snowflake-like fractals through the chaos game, ”snowflake” fractals need not be generated stochastically. For example, the Van Koch snowflake (briefly discussed in lecture) emerges from a process that is very naturally recursive.

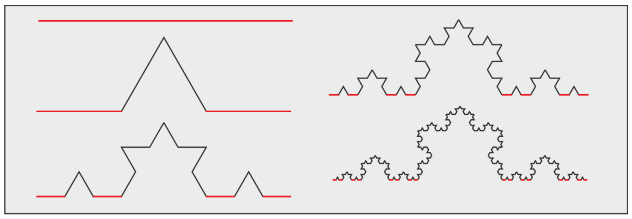

Figure 2: Generation of the Van Koch snowflake
Source: Math StackExchange

To extend our discussion of using recursion for fractal generation in class, *create a animated visualization of the Van Kock snowflake's generation.* You may find the code we used to recursively create fractal trees useful:

```
float angle = PI/8;

void setup() {
  fullScreen();
  colorMode(HSB);
} //setup


void draw() {
  background(140, 20, 255);

  translate(width/2, height);
  angle = map(mouseX, -width/2, width/2, 0, TWO_PI);

  branch(300);
} //draw

void branch(float len) {
  //if branches are too small to see anymore
  if (len < 2) return;

  //taller branches get greener and thinner
  strokeWeight(map(len, 2, 300, 0.5, 3));
  stroke(100, map(len, 2, 300, 255, 0), 100);

  line(0,0,0,-len);
  translate(0,-len);

  //recursively make first branch
  pushMatrix();
  rotate(angle);
  branch(len * 2/3);
  popMatrix();

  //then second branch
  pushMatrix();
  rotate(-angle);
  branch(len * 2/3);
  popMatrix();
} //branch
```