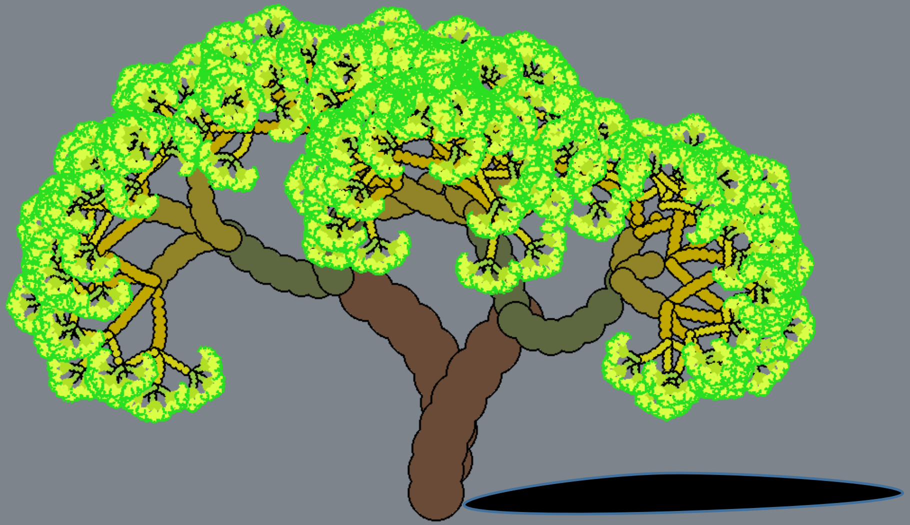
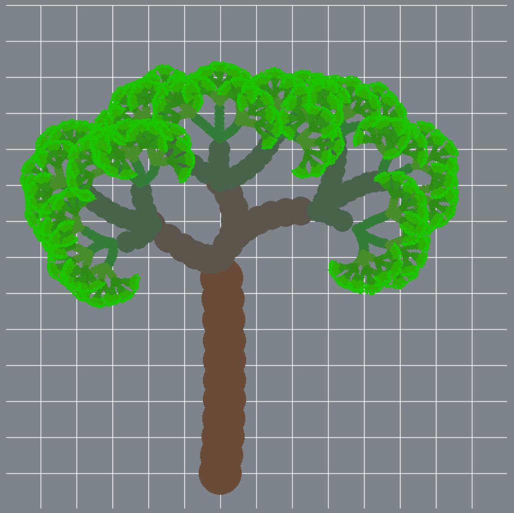
Fractis Arboretum



Introduction

- Draw trees using a simple and recursive method in Python
- Use 10 Levels of fractal definitions and a dose of randomness to have an organic appearance.
- From the same recipe, possibility to obtain an infinite number of trees, due to the randomness.

Surface o= Pi * R^2 = Pi * D^2/4



Conservation of Trunk surface (Leonardo da Vinci)

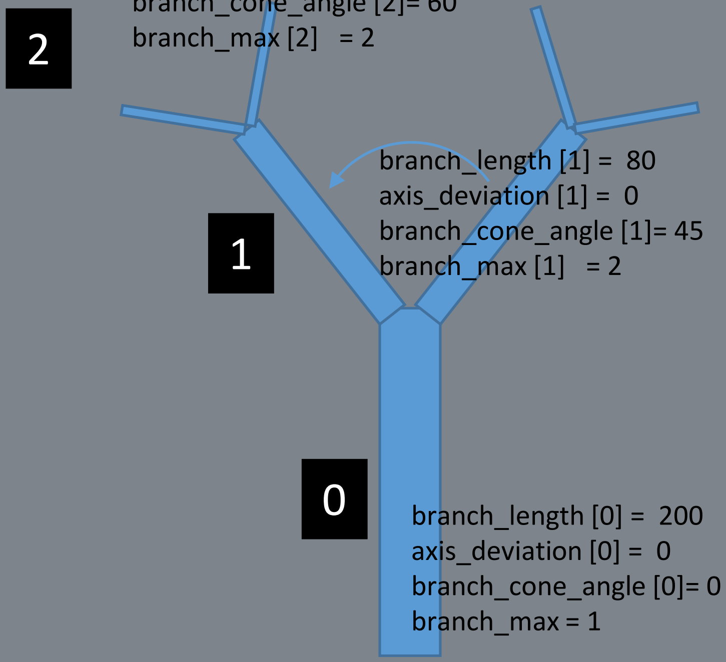
surface= Pi * diameter_ini ^2/4 /only one branch at level 0

Finding Diameter at any segment [n], from number_branch [n]

Surface= Pi * diameter [n] ^2/4 * number_branch [n]
Therefore
diameter [n] = SQRT(4*surface / (Pi*number_branch [n]))

Each tree is represented by up to 10 levels
Each level has properties such as

- Nb of Branches
- Branch length
- Angle from parent
- Radius Start and End
- etc.



branch_length [2] = 40
axis_deviation [2] = 10
branch_cone_angle [2]= 60
branch_max [2] = 2

branch_length [1] = 80
axis_deviation [1] = 0
branch_cone_angle [1]= 45
branch_max [1] = 2

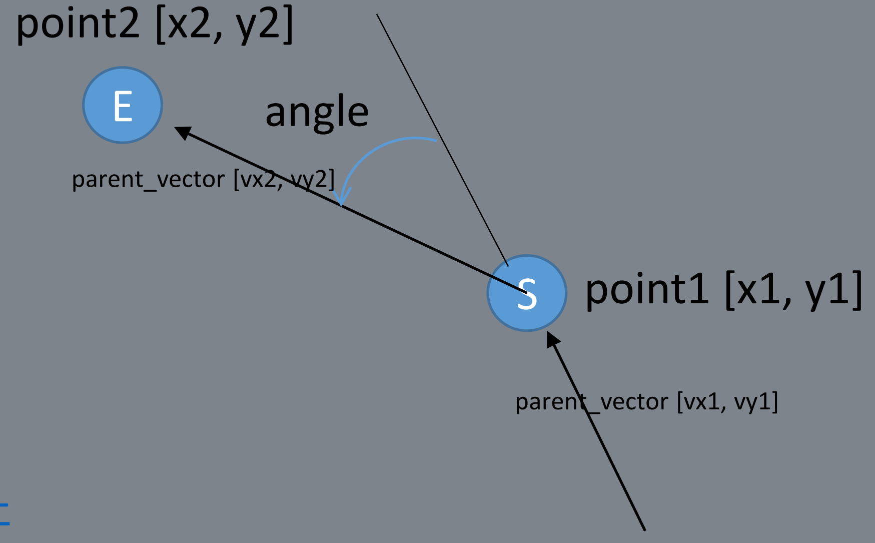
branch_length [0] = 200
axis_deviation [0] = 0
branch_cone_angle [0]= 0
branch_max = 1

[Level, Length, Rs, Re, Nb Branch, Length Random, Angle Random]

```
self.data = [
    [0,  375,    60,    30,   1,   0.2,   0.1],
    [1,  150,    40,   130,   2,   0.3,   0.4],
    [2,  100,    30,   100,   2,   0.4,   0.4],
    [3,   50,    15,   100,   2,   0.3,   0.4],
etc… ]
```

nodelist [level,node]
node = [x, y, parent_vector]
for instance node = [0,100, -0.5,-0.5,]

branchlist [level,branch]
branch = [node_start,node_end, property1,property2]
branch = [node003,node006]



point2 [x2, y2]

E

angle

parent_vector [vx2, vy2]

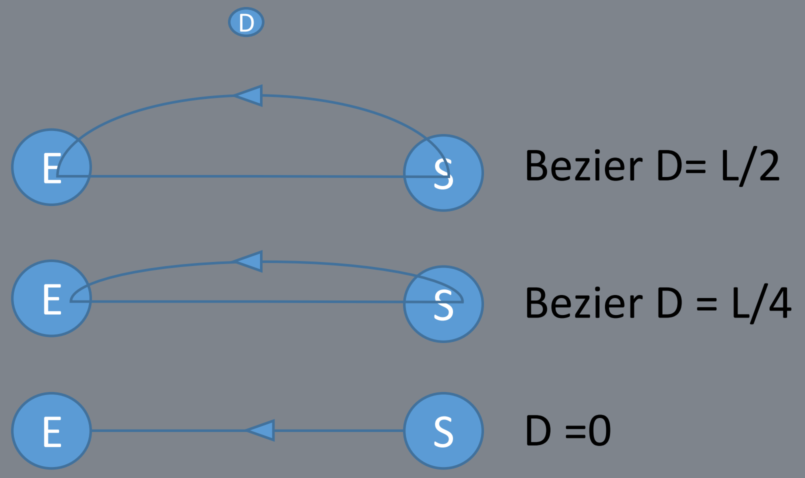
S

point1 [x1, y1]

parent_vector [vx1, vy1]

https://stackoverflow.com/questions/69804595/trying-to-make-a-bezier-curve-on-pygame-library
https://samgentle.com/playgrounds/bezier

A branch is a line between a start and end points
However, I have introduced a curvature parameters
The implementation is done by a three point Bezier curve which is very simple to draw



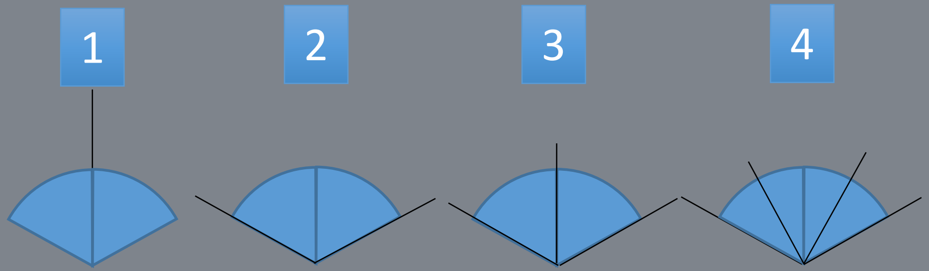
D

E S Bezier D= L/2

E S Bezier D = L/4

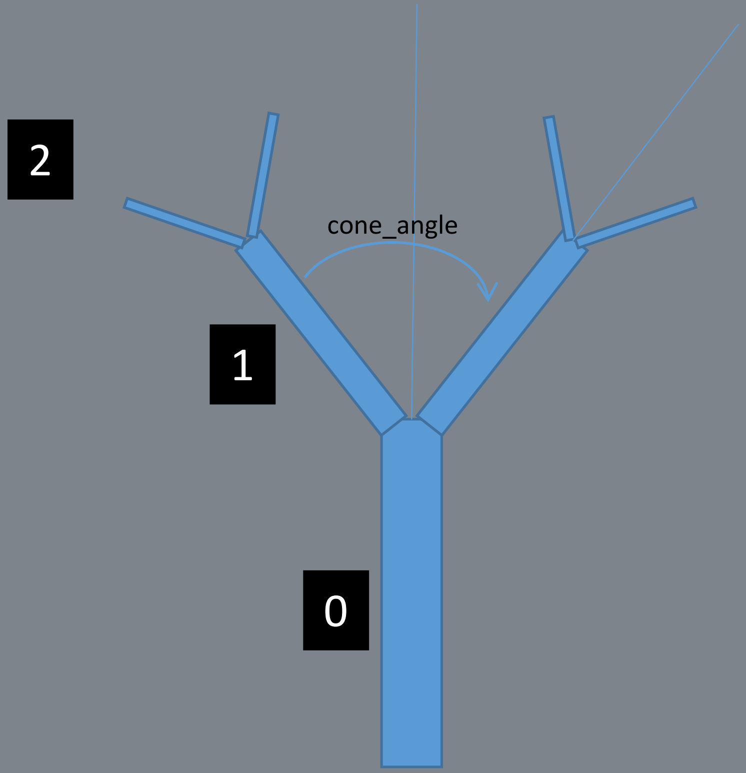
E S D =0

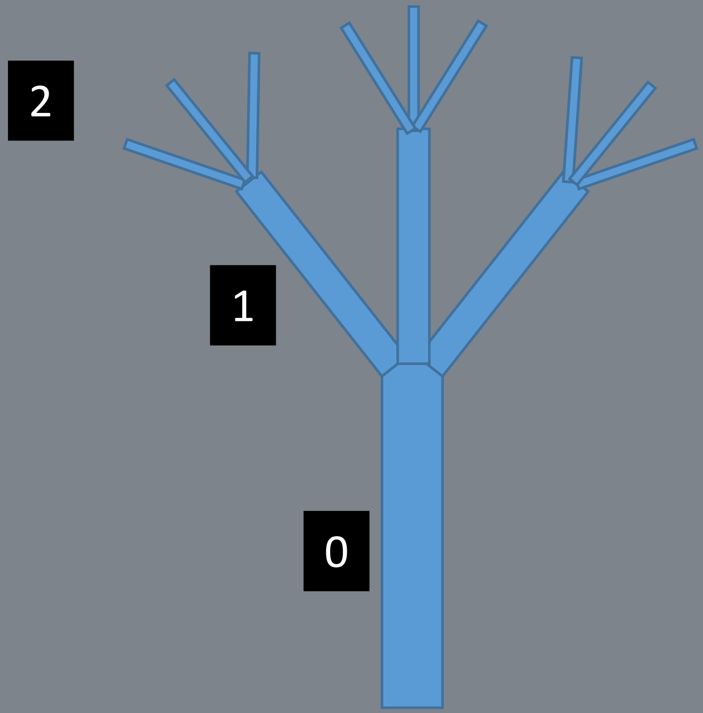
Some details about cone angle, which represents how the branches are positioned and distributed compared to the parent node.



The branches are distributed to cover the Angle Solid

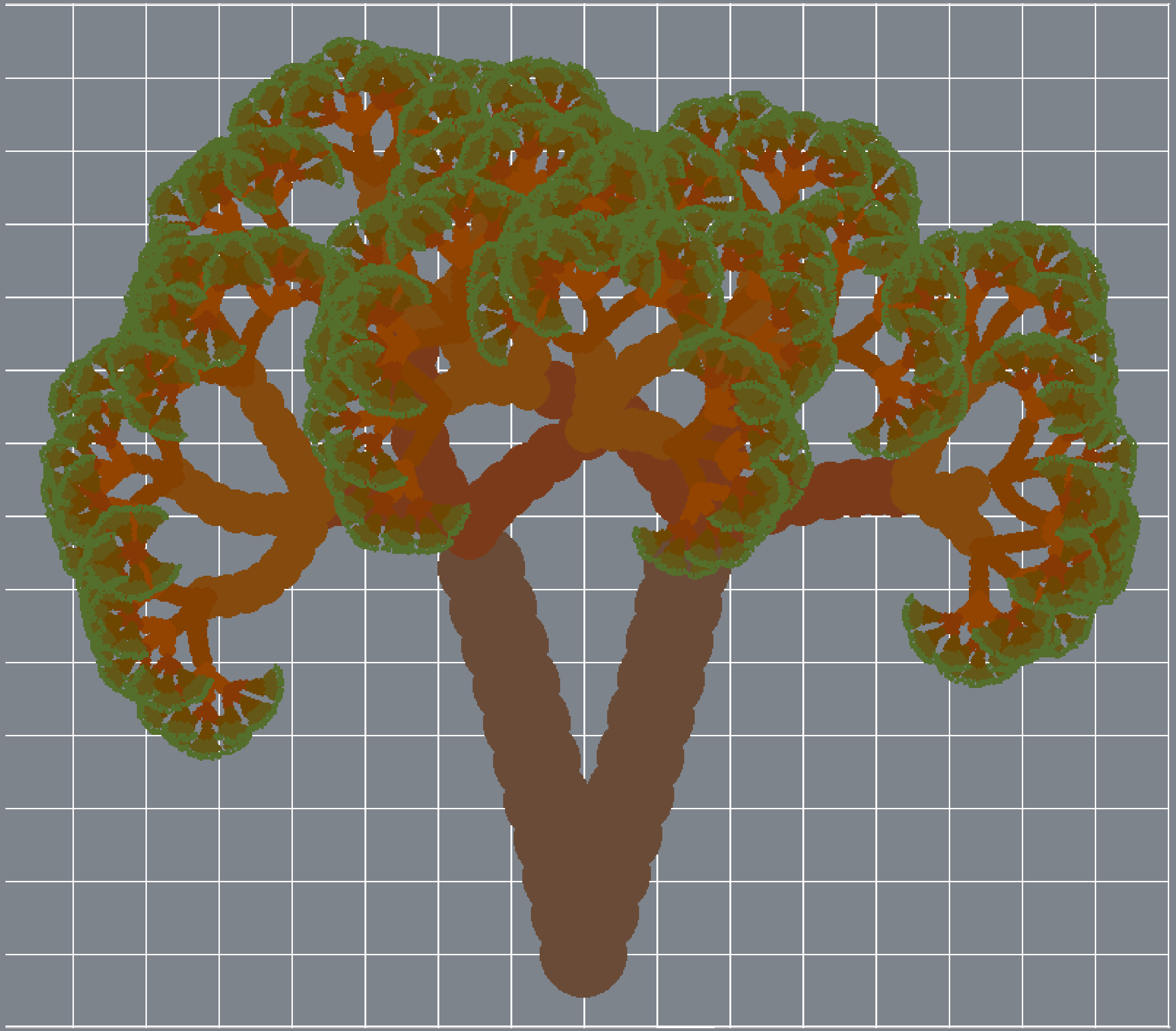
- If branch_node = 1 : Angle =0
- If branch_node = n :
- Angle = cone_angle/ (n-1)

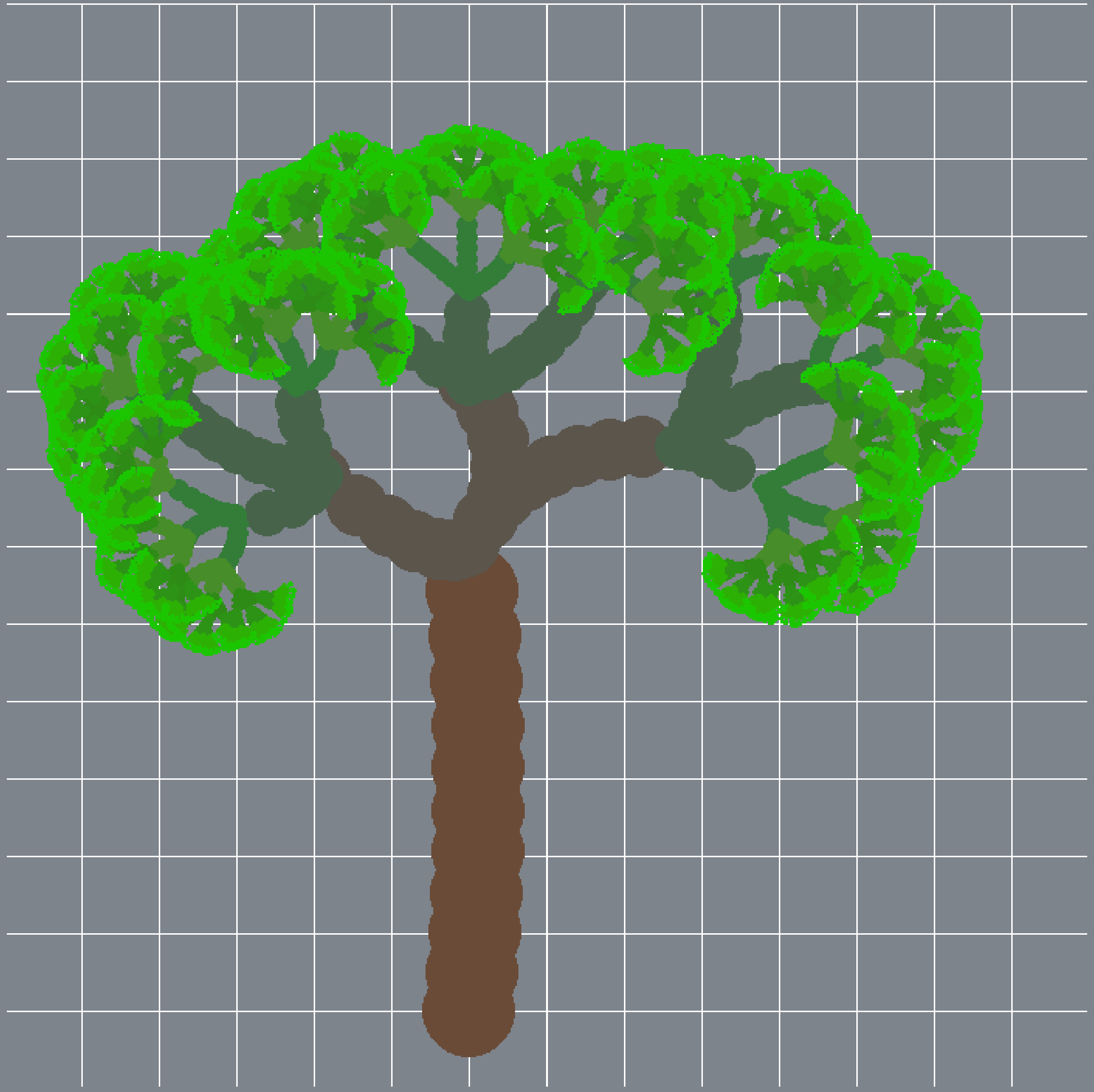


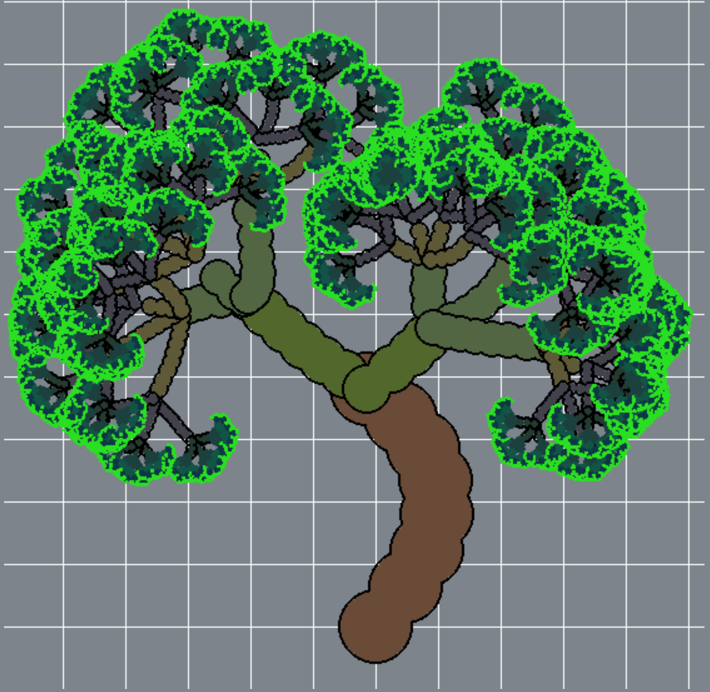


First attempts

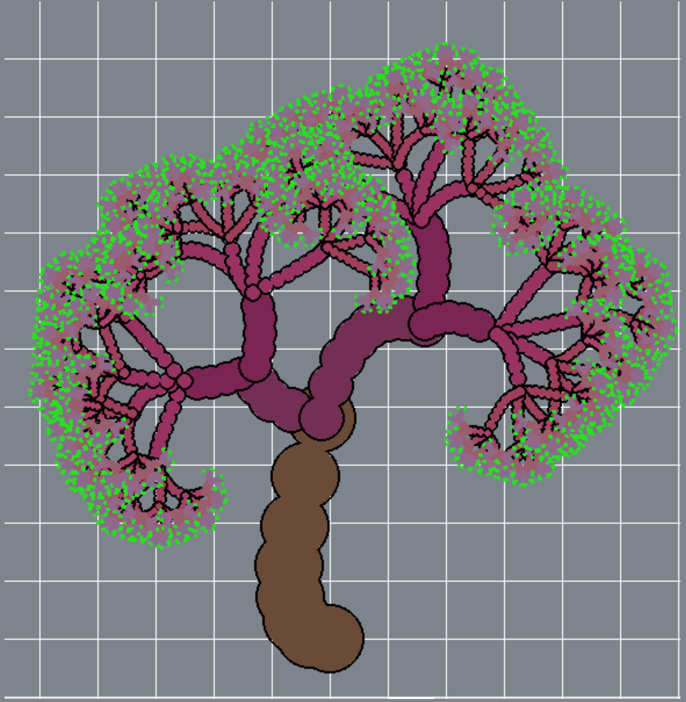
The drawing is done by making circles along the Bezier curves

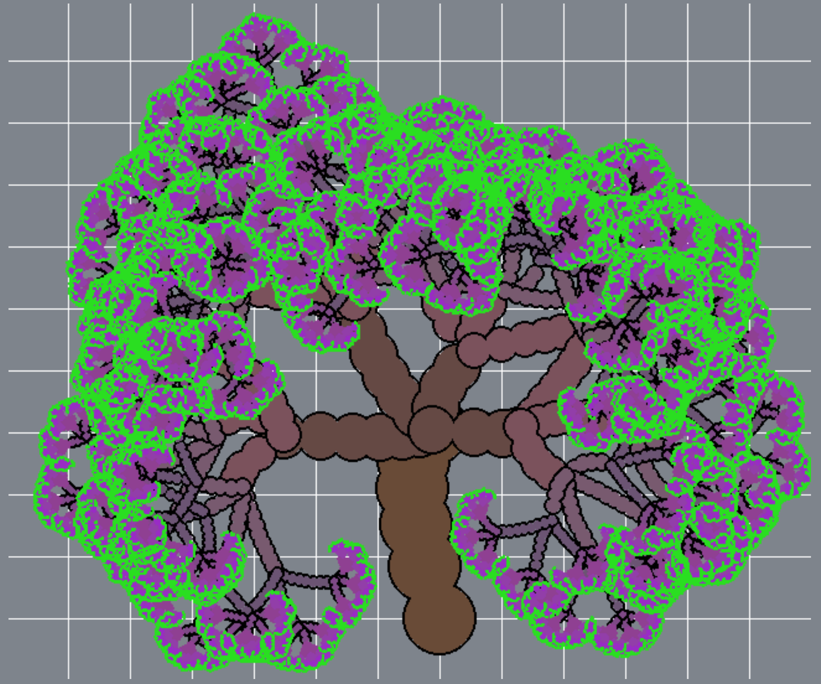


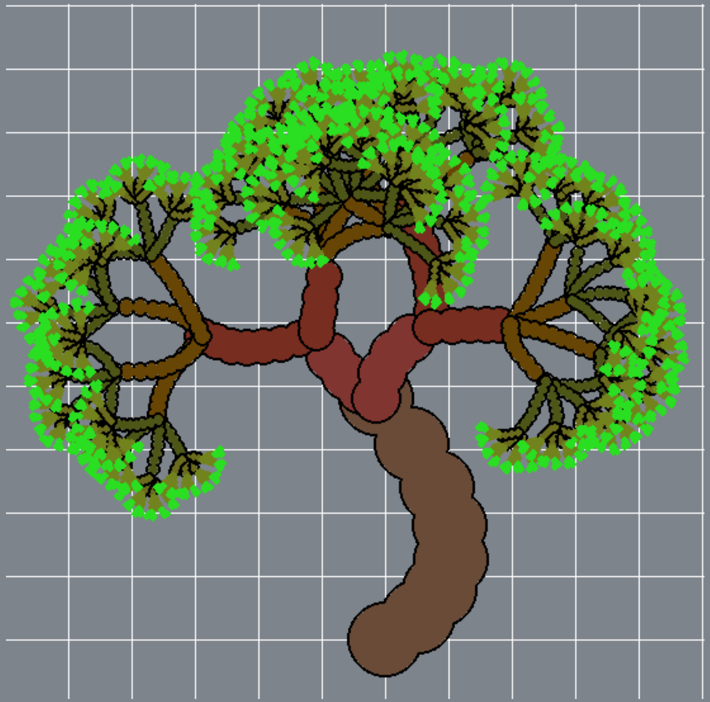


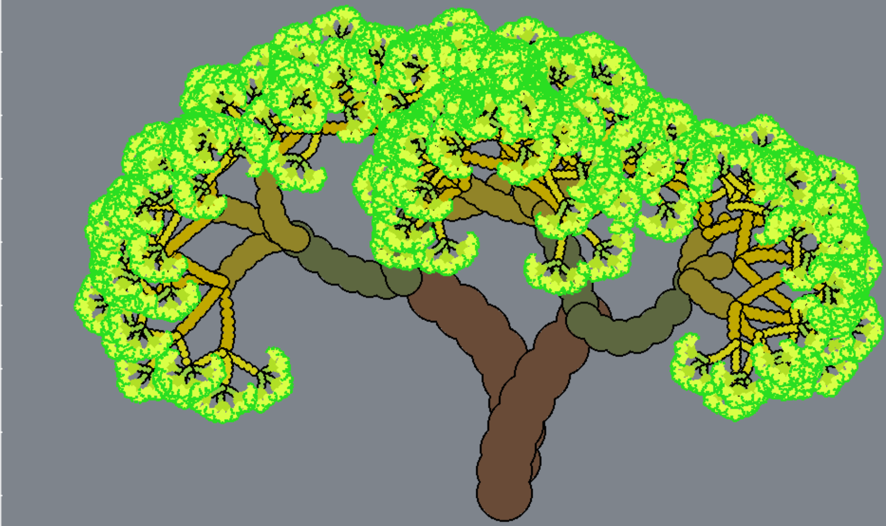


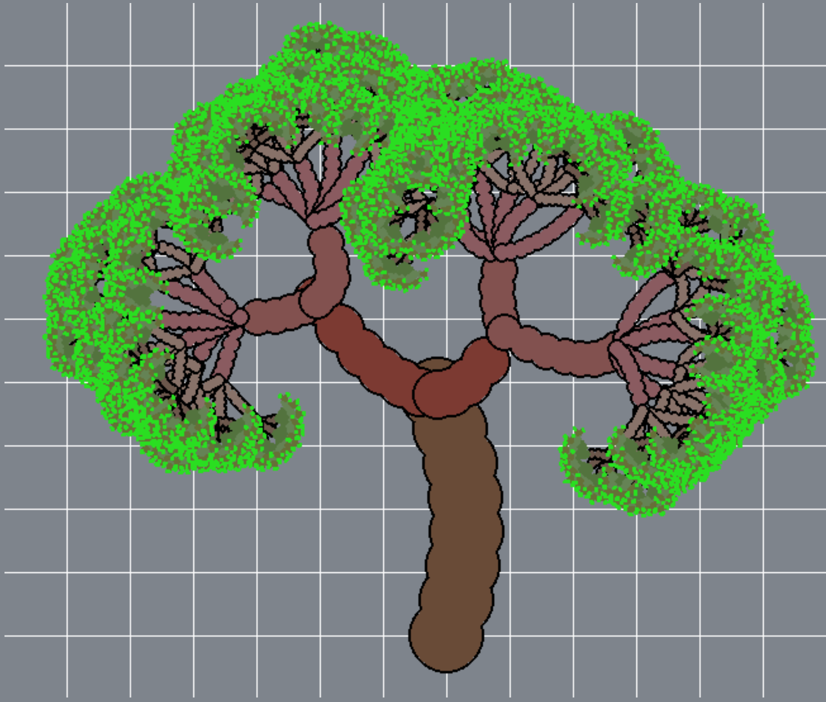
With Black Outlines



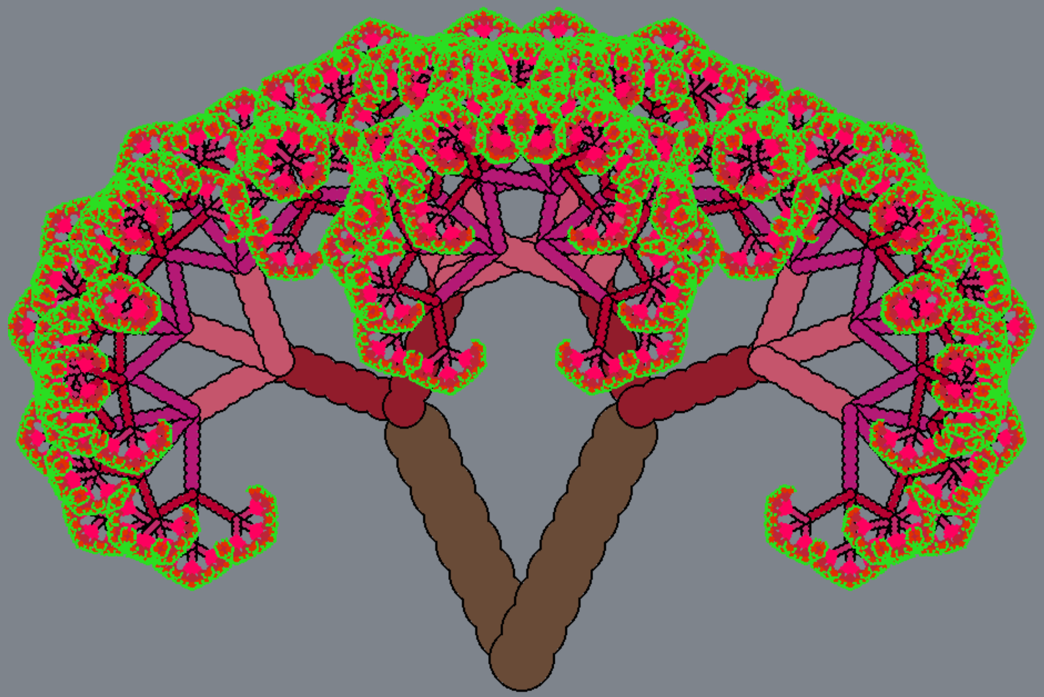




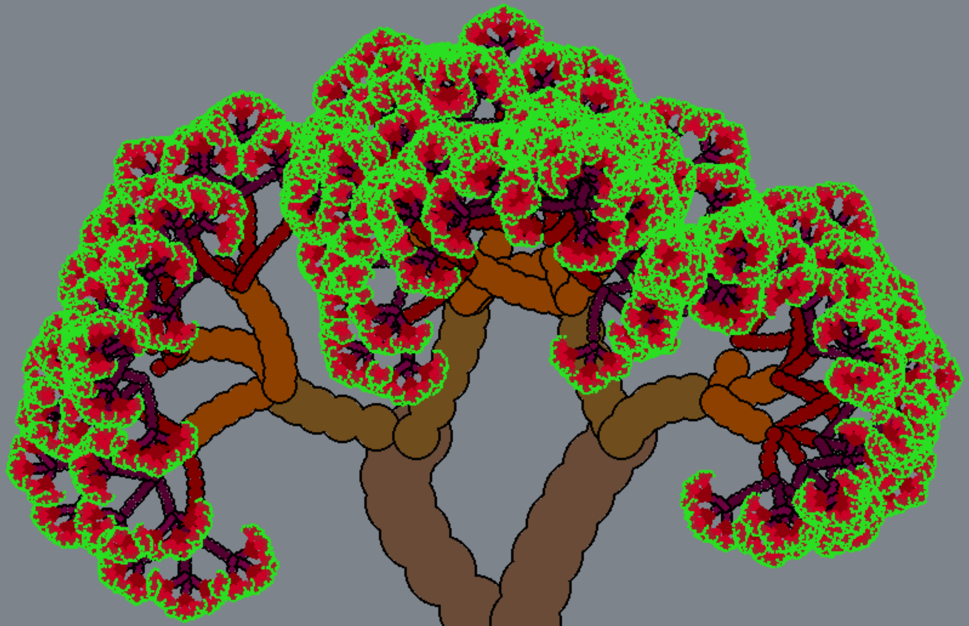


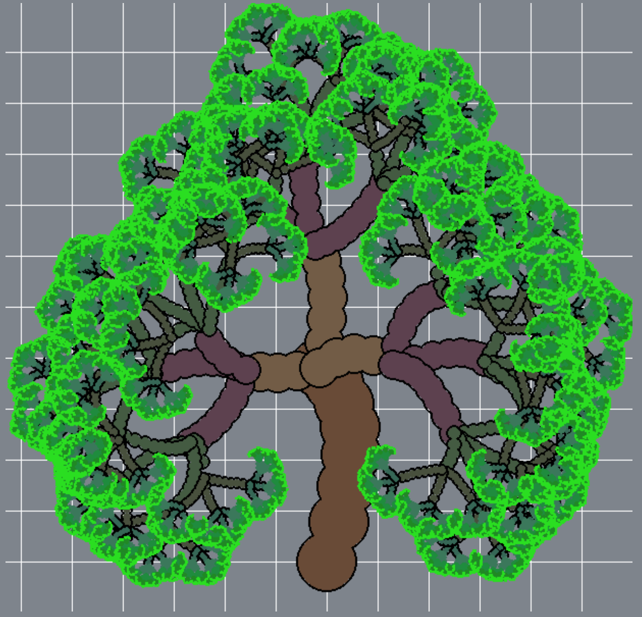


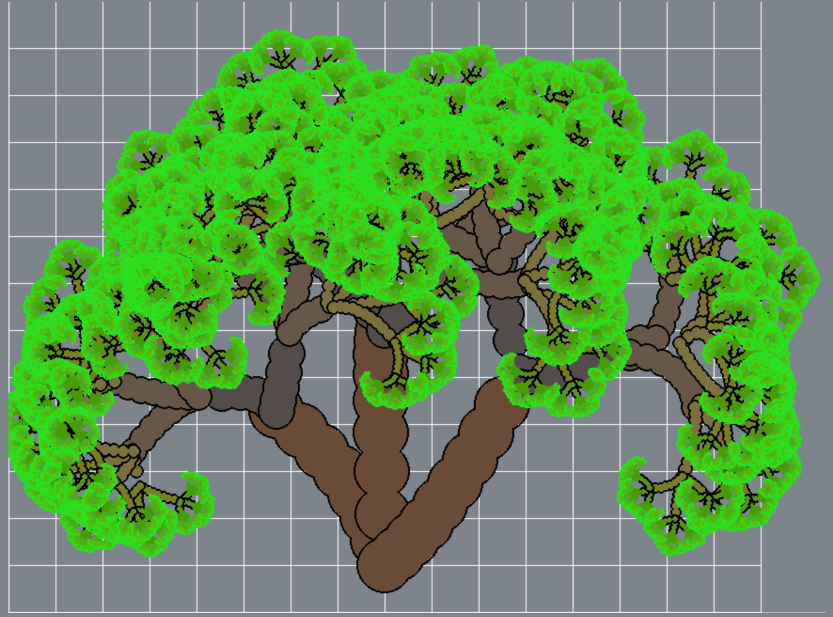
without noise in length and curvature

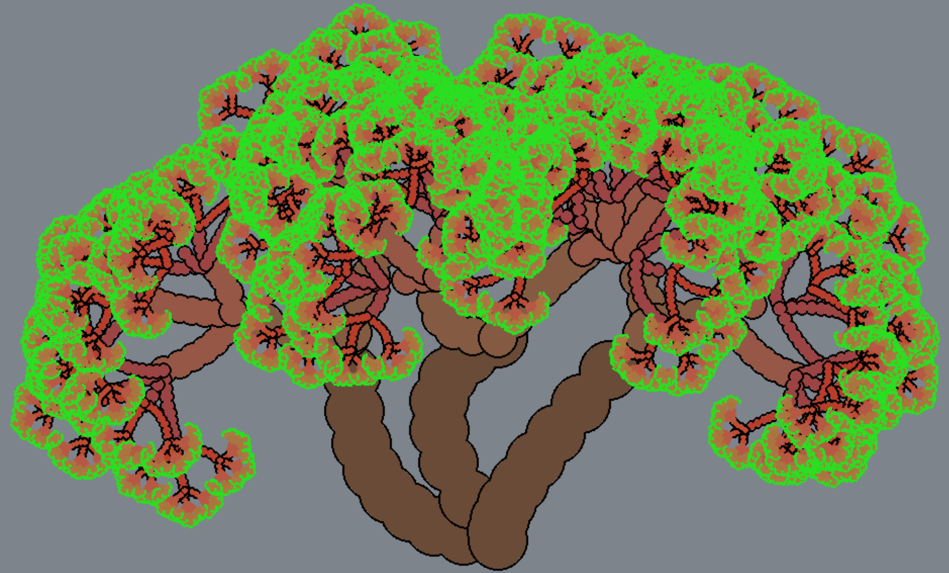


Adding noise, making things more “organic”

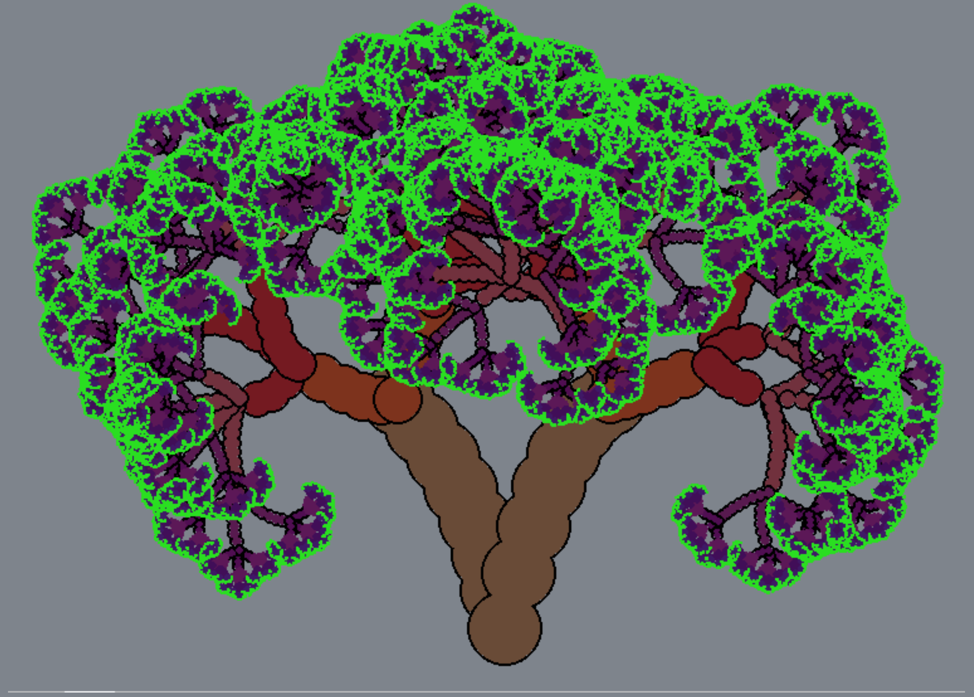


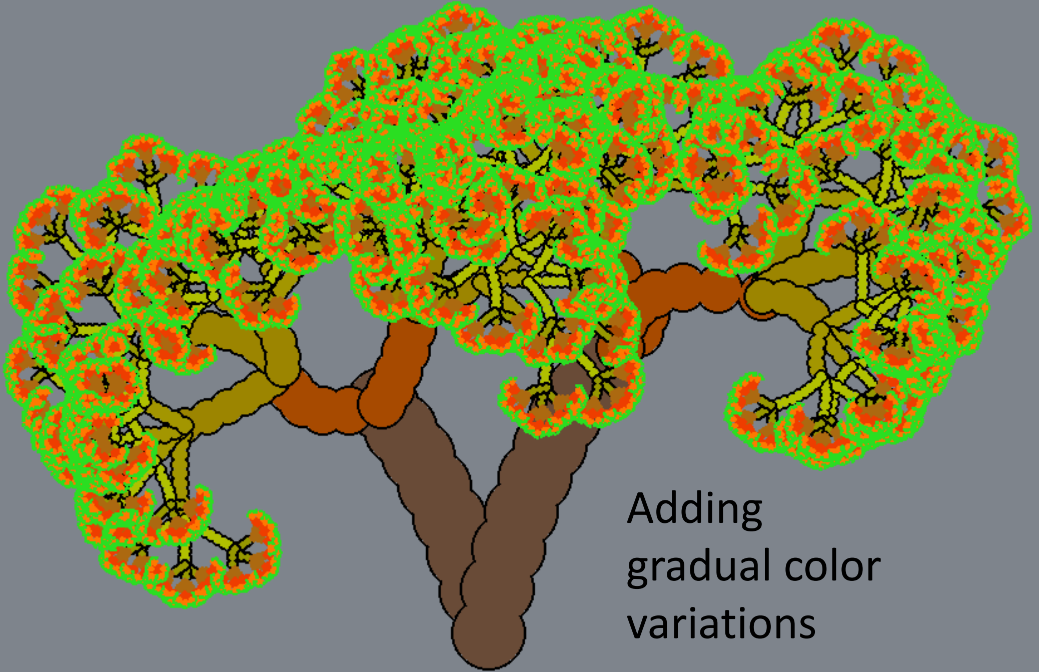




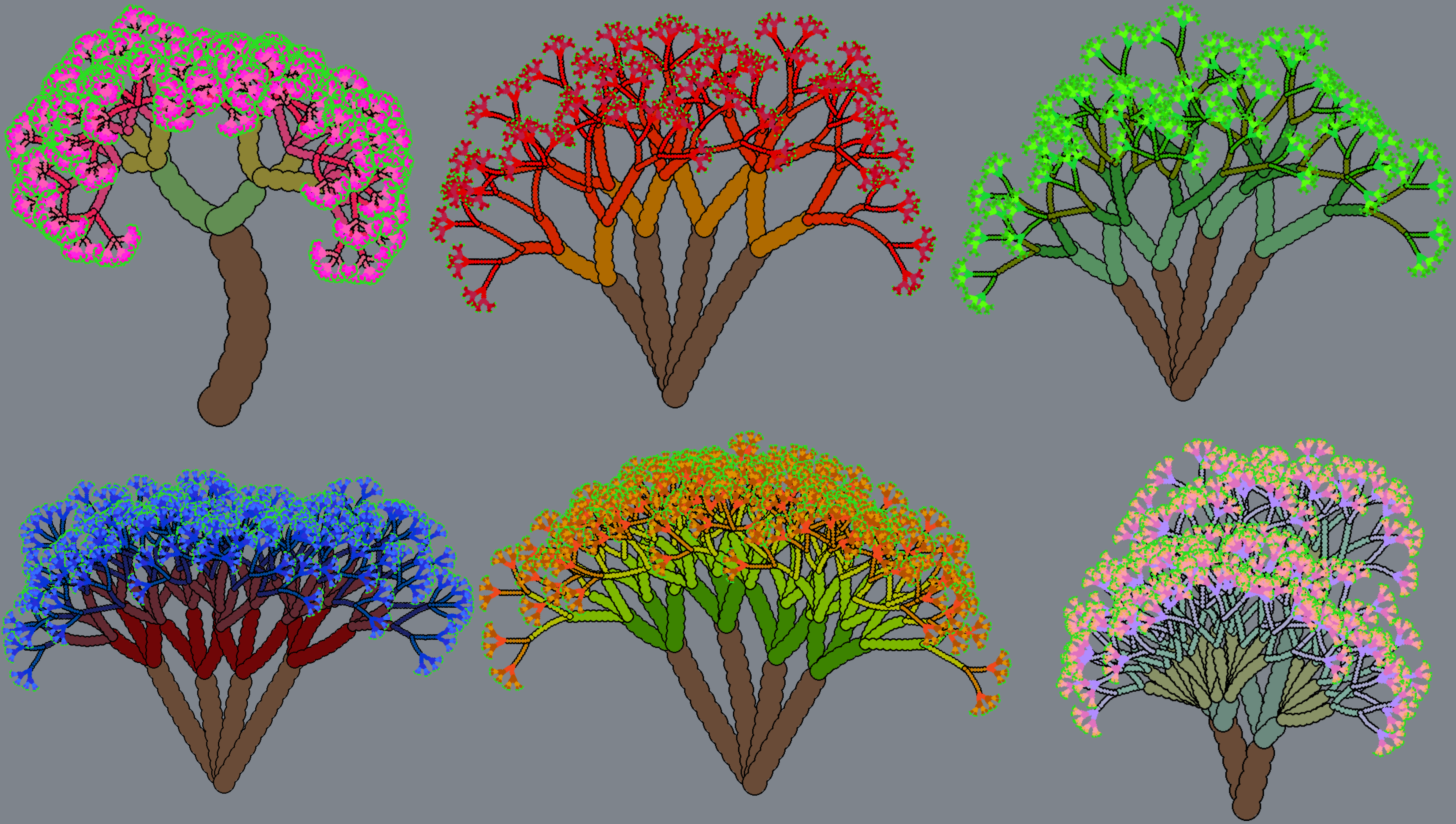


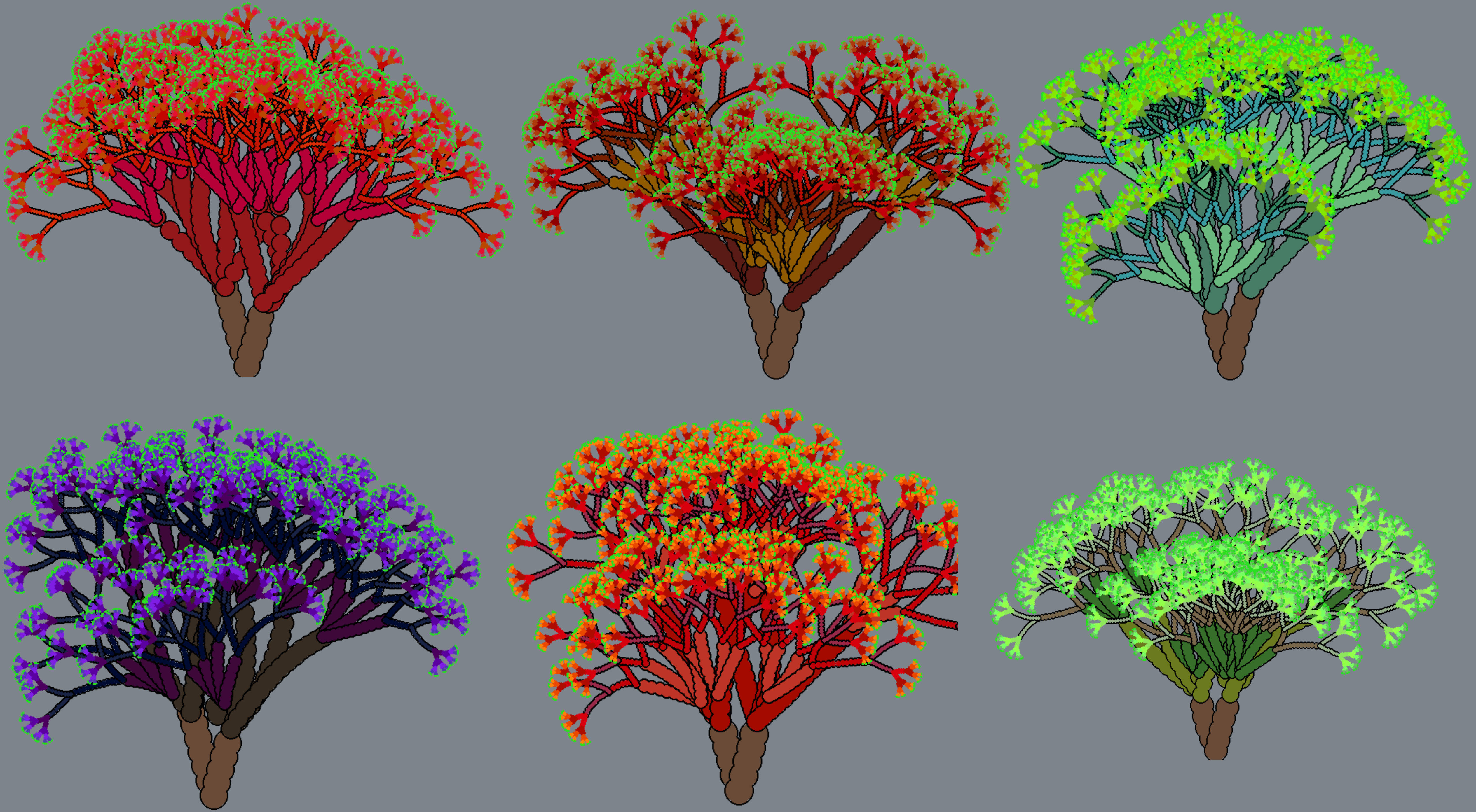
Adding, double and triple trunks

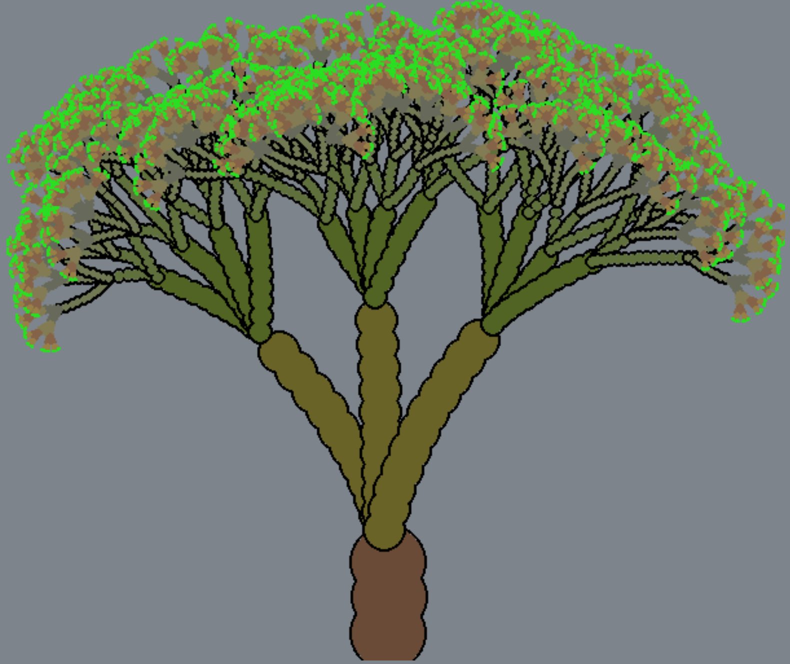


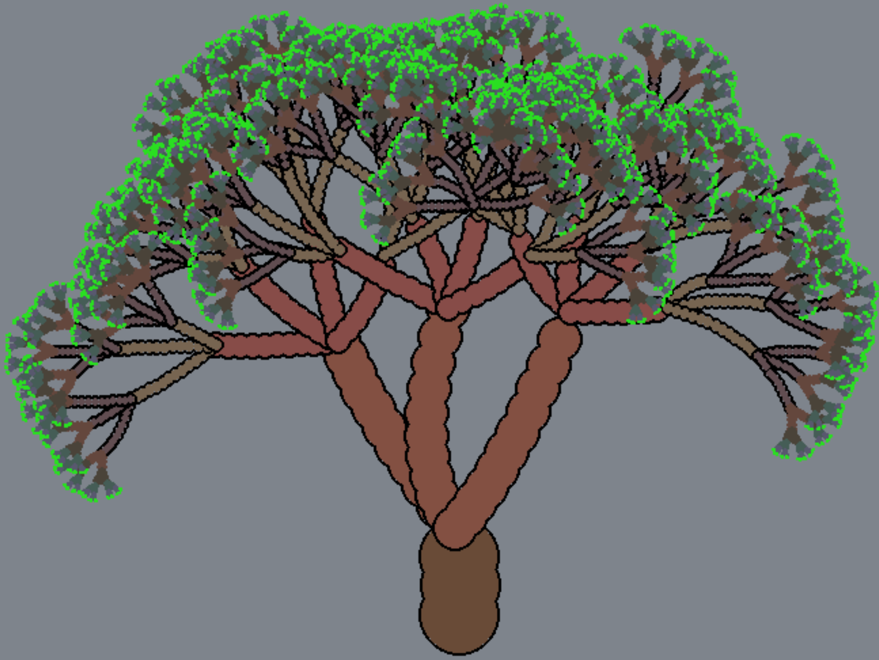


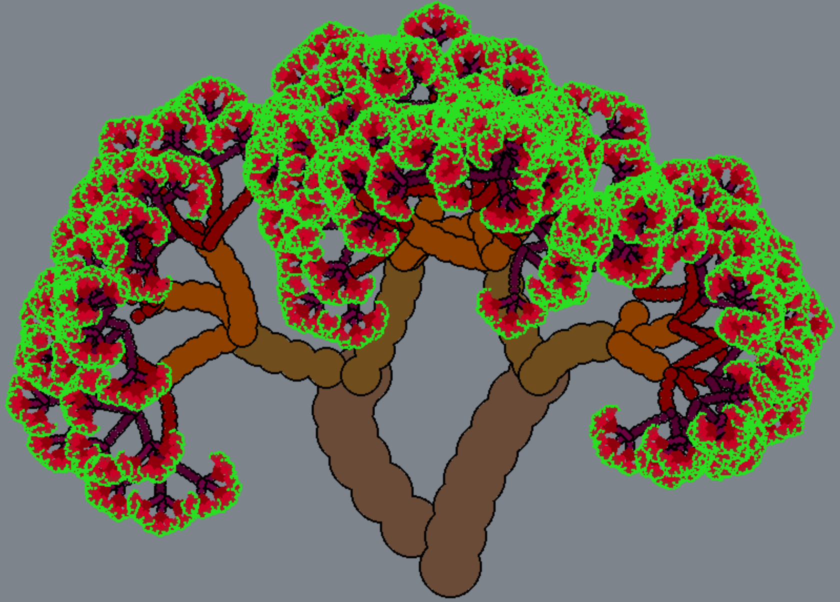
Adding gradual color variations

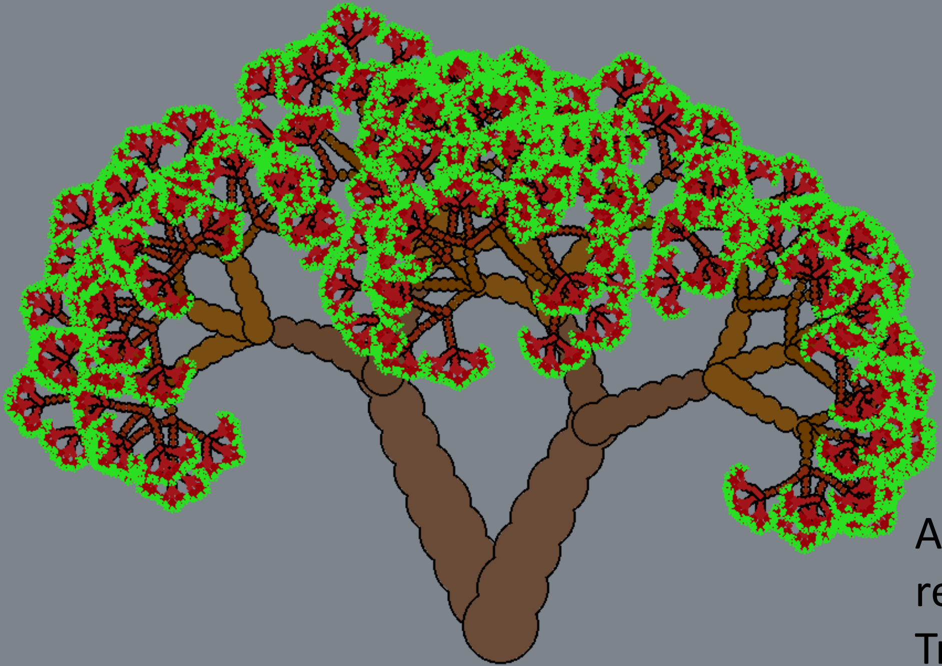


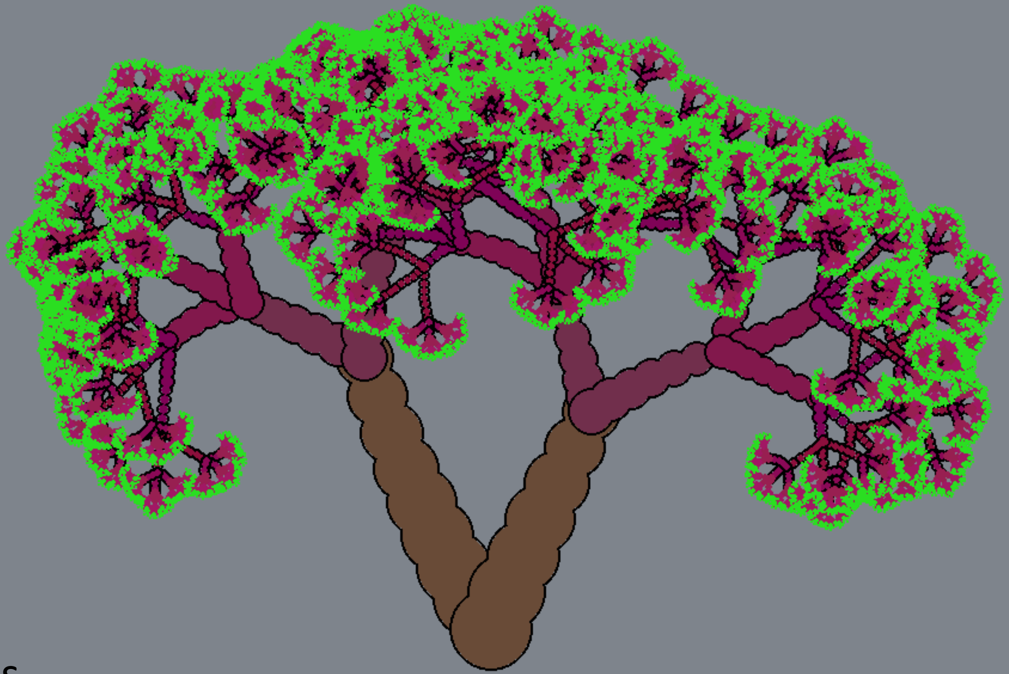




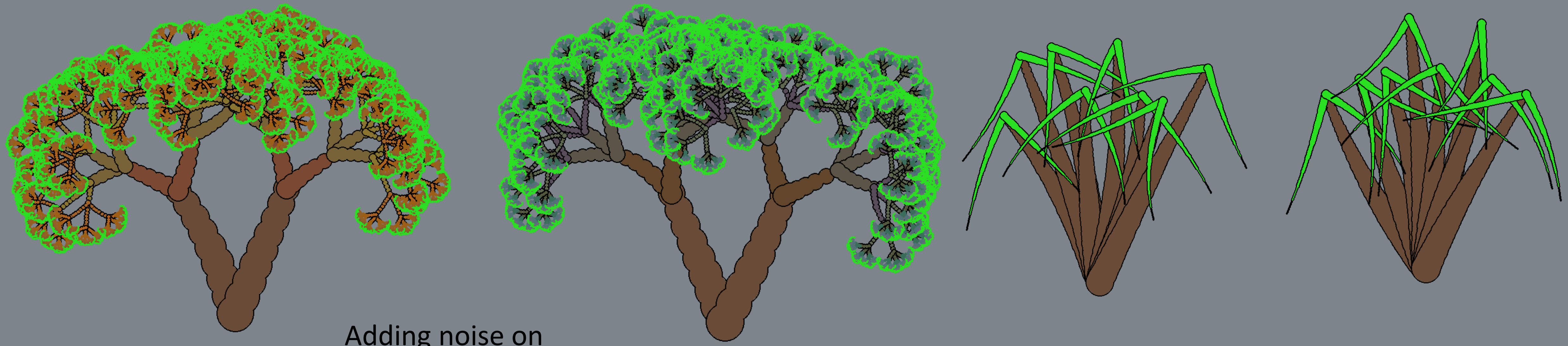




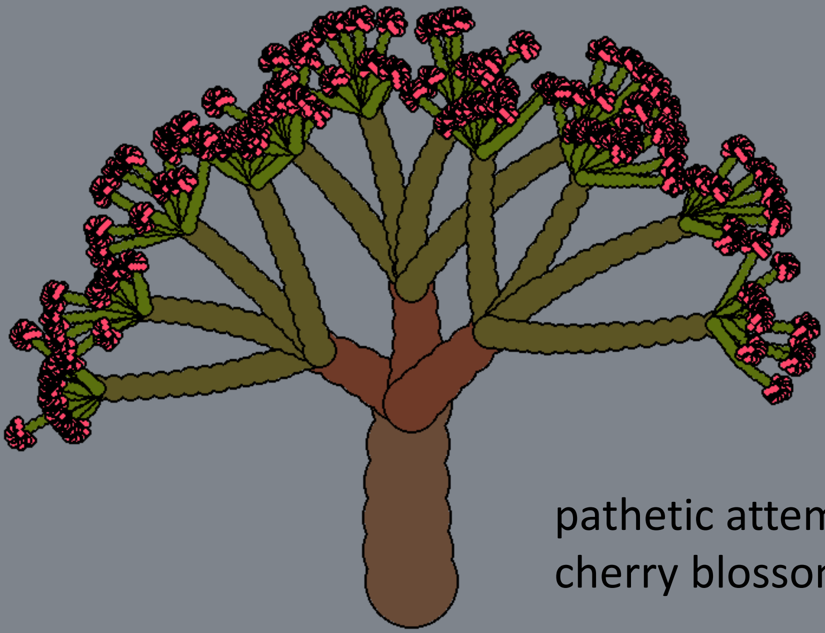


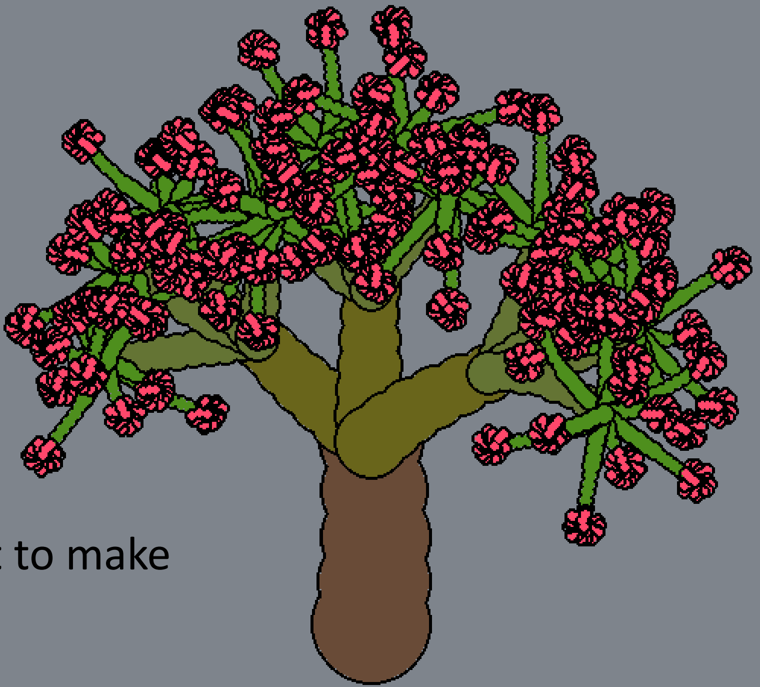


Adding
reducing
Trunk Sizes

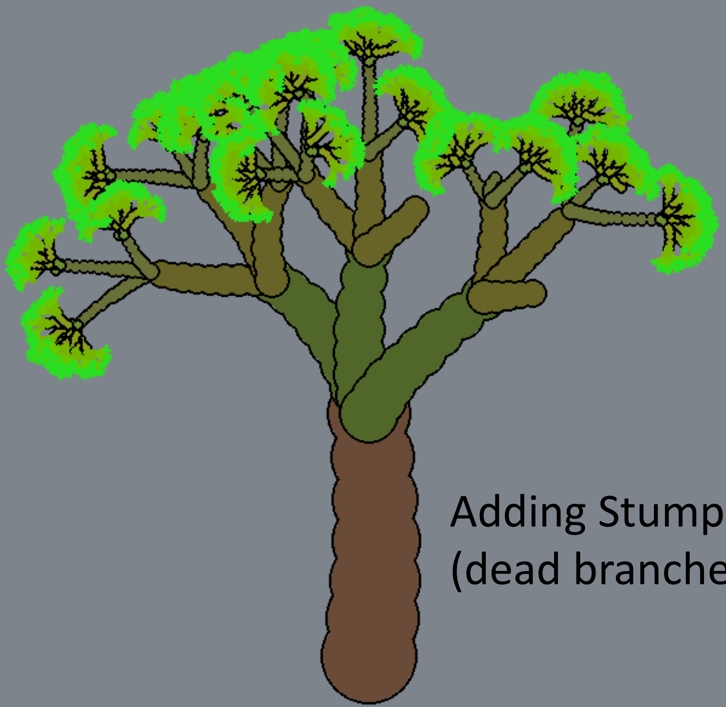


Adding noise on each branch angle

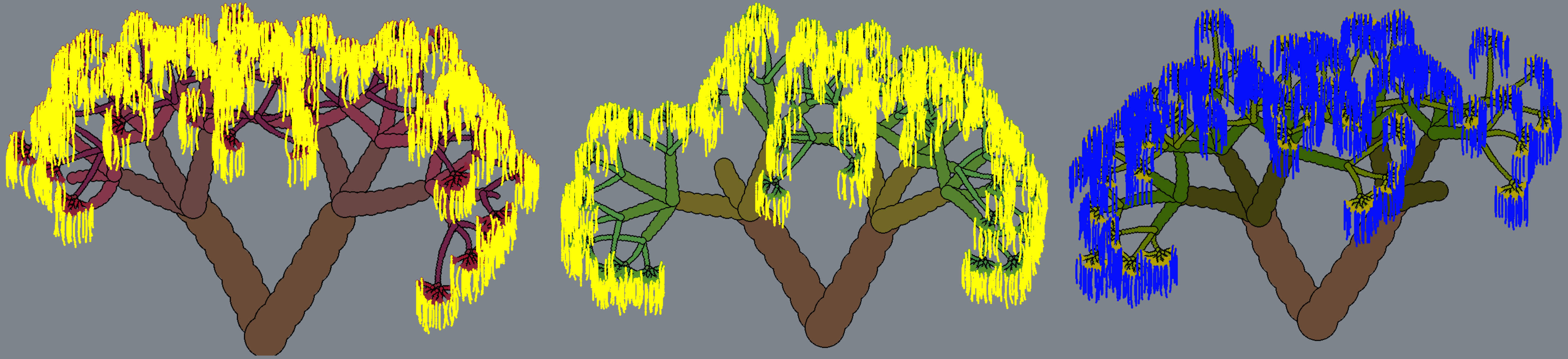




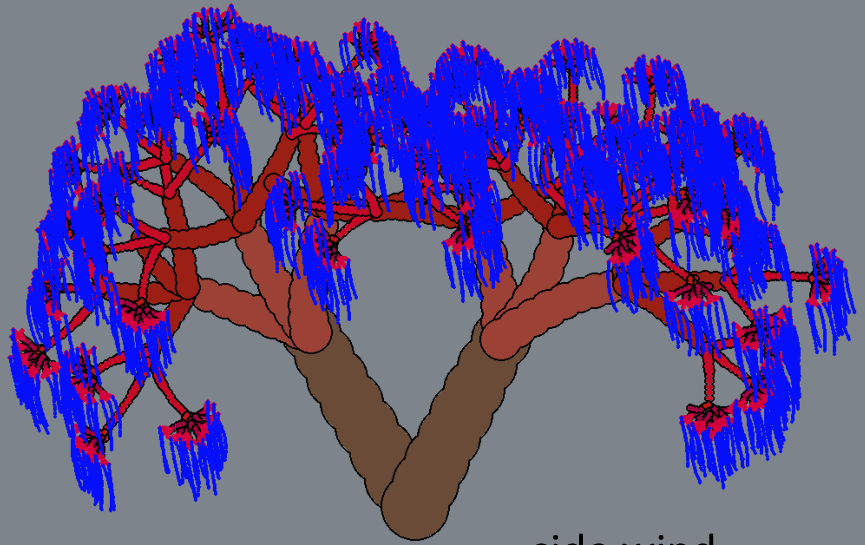
pathetic attempt to make cherry blossoms



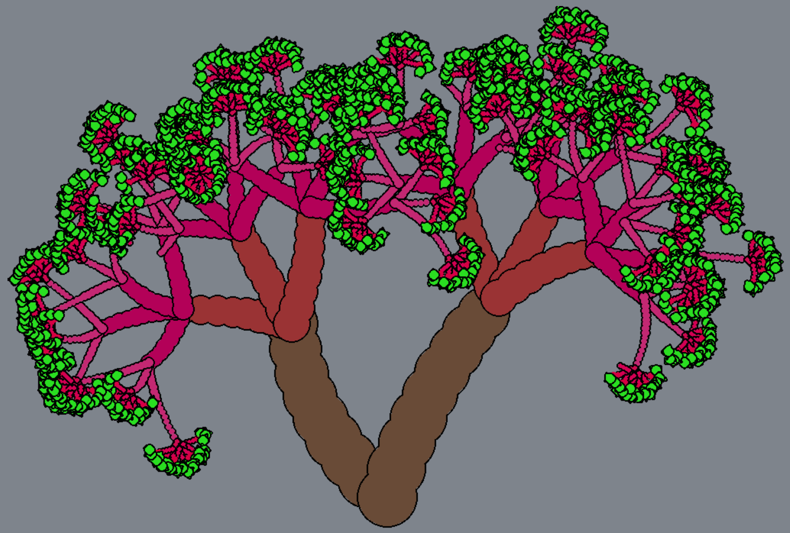
Adding Stumps (dead branches)



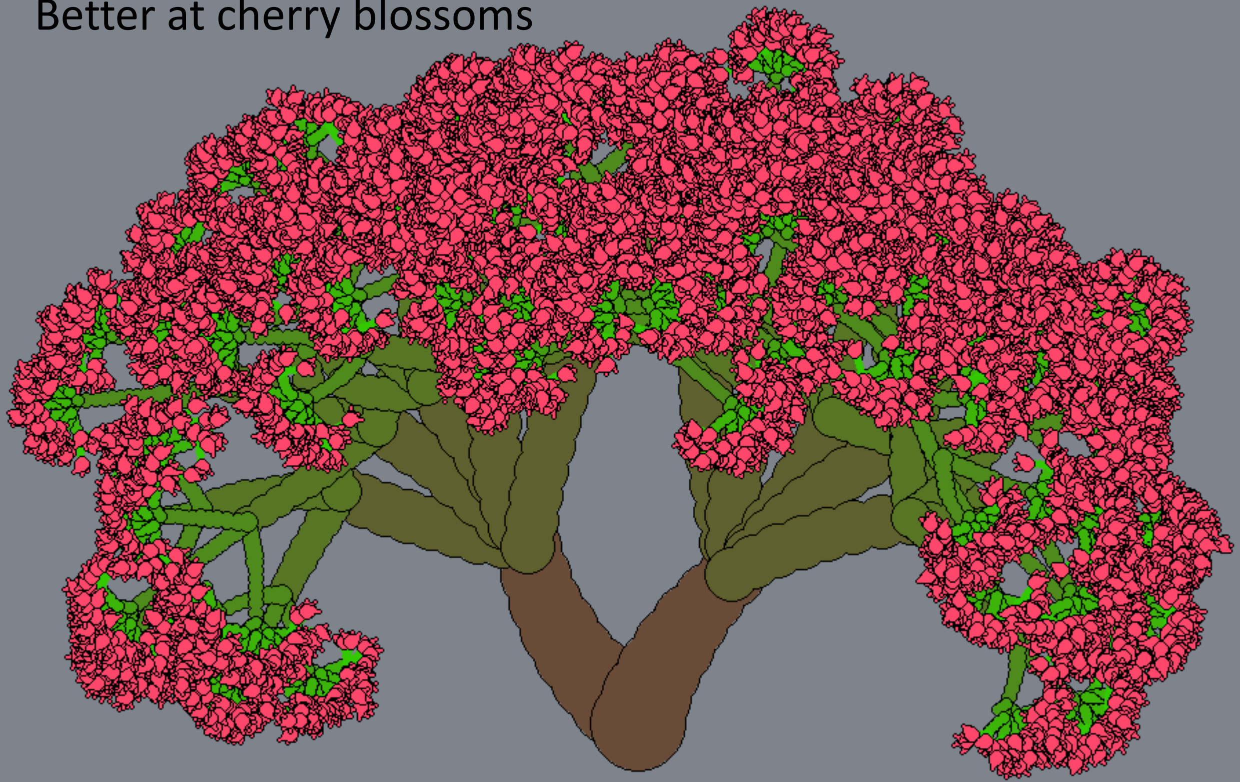
Adding Gravity Effect

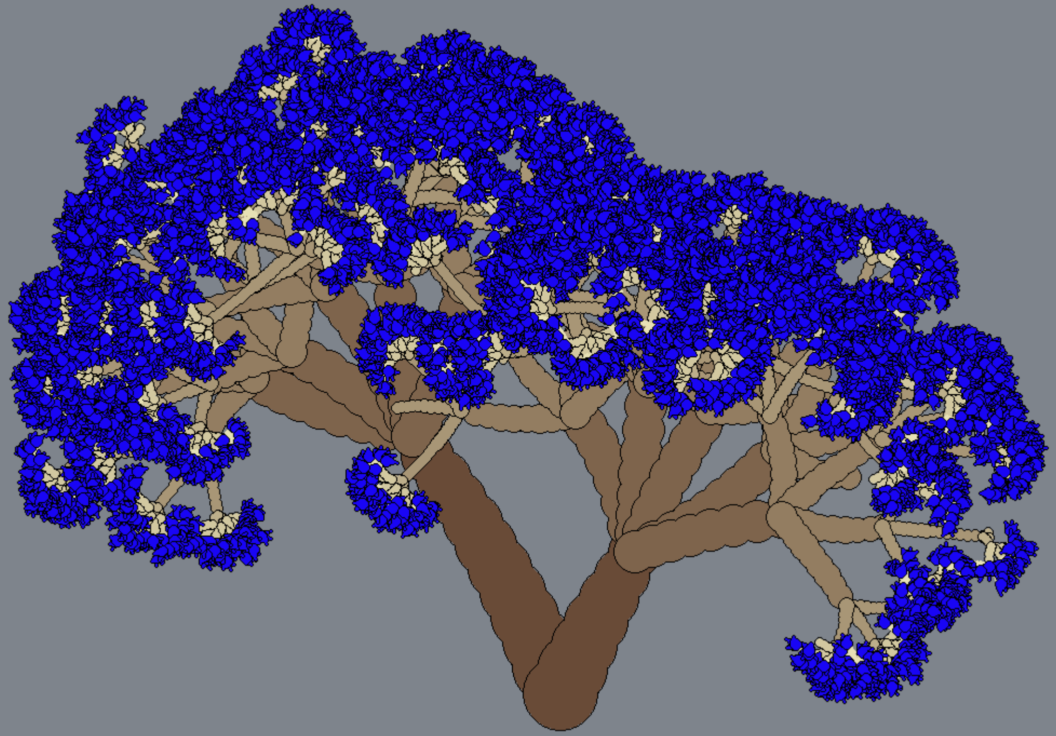


side wind



Better at cherry blossoms





with color interpolation
Trunk->leaves