HW03 Fractal Tree Report

Data Structures

Alex Darwiche

Describe your design and implementation approach:

- For this project, we implemented a stack to draw a fractal tree. The basics of my approach were first to define the stack: I used a stack of fractalTreePanelExtension objects where each object essentially corresponded with 1 branch being drawn. I would "seed" the stack with the first branch (or the trunk of the tree) and then set a depth, then let the methods fill and iterate through the stack to draw the tree.
- Step 1: Push the initial branch onto the stack with depth L
- Step 2: Pop then process the top branch (processing = drawing the branch)
- Step 3: Check the depth, if Depth is greater than 0, then push a left and a right branch with depth L – 1
- Step 4: Repeat steps 2 and 3 until the stack is empty.

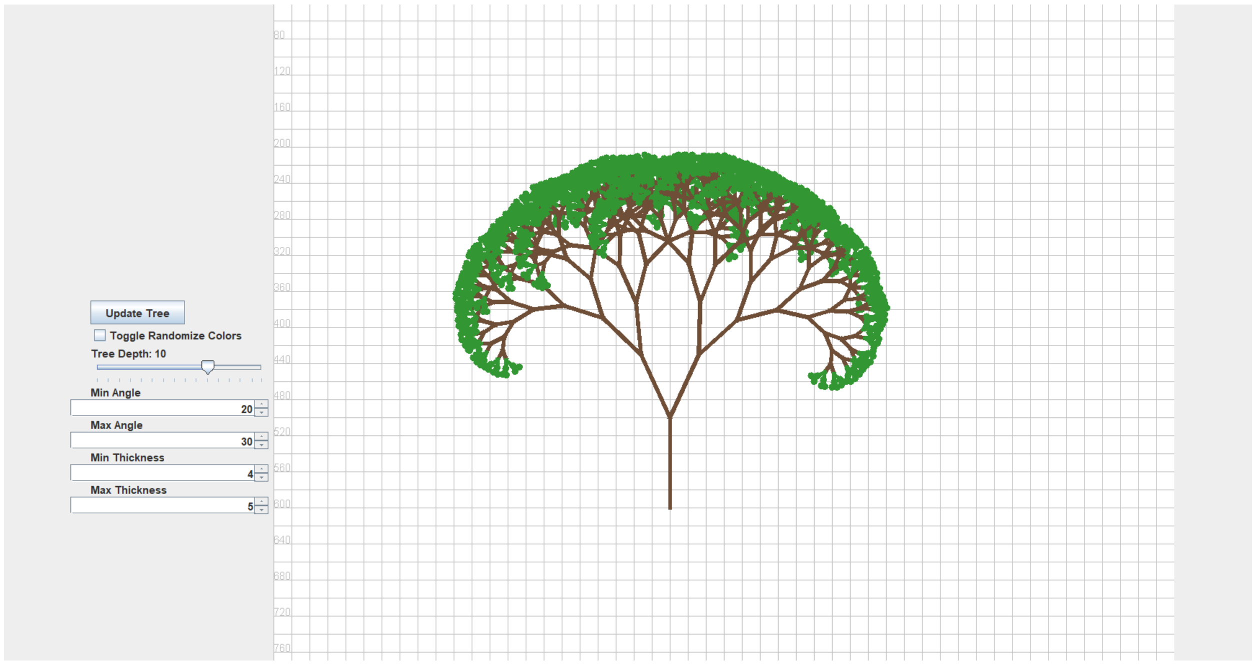
Complexity Analysis of the iterative method compared to the recursive baseline:

- Ok, so recursive vs iterative time complexity. The iterative approach is better from a time complexity perspective in most cases that I've read about online. However, in this case I believe the time complexities to be the same, namely O(2^d) where d is the depth of the tree. Essentially each time you add a new layer of depth, the numbers of nodes to create doubles for each algorithm.
- The additional complexity in recursion might come from the fact that you'd need to "unravel" once you've hit the dead end. Essentially traveling back up to the latest method call that hasn't been hit yet. Iterative approach with a stack doesn't require this backtracking as we simply pop whatever is currently on the top of the stack. I'm not 100% sure how this translates in time complexity, but I think this makes iteration a touch better on memory and performance.

Describe Testing Methodology and Results:

- First, to test the UI and the dynamic parameters I worked in the Java User Interface, making changes to each parameter and seeing how that affected things. I started first with tree depth, changing the tree depth from 5 to 10 to 15. The tree filled in each time, showing the increases in depth. The second test I did was with the angles. I made the angles very large (60 degrees) and confirmed that the tree grew outwards very quickly. This confirmed that the adjustment of the angle parameter was appropriately changing the tree graph. I repeated the same process with thickness and randomizing colors, again confirming that the adjustments in the UI were reflected in the outputs.
- Last testing I did was with Junit, ensuring all the methods that I filled out or created were functioning as intended. All 7 tests passed.

Screenshot



Tree Depth: 10

Min Angle 20

Max Angle 30

Min Thickness 4

Max Thickness 5

-