1 Administrative Topics

- We take the quiz.

2 L-Systems Project 10

Today we talk about how to understand the stochastic L-systems that Bruce supplied. We also talk about how to create an NPR style.

2.1 Symbols in drawString

To make sysTree2.txt and sysTree3.txt work, we need the following symbols supported by Interpreter .drawString.

- F and f must both mean “forward by the (modified) distance”
- L must draw a leaf whose size can be scaled by a modifier.
- ! must handle the width. If there is no modifier, then it should decrease the width by 1 (being careful not to reduce it below 1). If there is a modifier, then it should set the width to the value of the modifier.

To draw these trees, a good value for distance is 1.
2.2 Understanding sysTree3.txt

Here is the first example L-system:

base (50)F
rule F (50) f ![30]−F@g(5)L>][40]+F@g(5)L>!F
rule (x)f (1.5*x)f (1.25*x)f (1.7*x)f
rule (x)− (x*1.2)− (x*0.8)− (x*1.05)− (x*1.0)− (x*0.95)− (x*0.87)− (x*1.12)−
rule (x)+ (x*1.2)+ (x*0.8)+ (x*1.05)+ (x*1.0)+ (x*0.95)+ (x*0.87)+ (x*1.12)+

And the observations we made about it:

- Turns, forwards, and leaf-sizes are modified by modifiers.
- The structure of the tree is deterministic (we always move forward and make two branches (or subtrees) to the left and right, and the grown the branch (or subtree) up.
- Distances and angles are stochastic
- The scales in front of F and f should be of comparable sizes.

2.3 Understanding sysTree2.txt

Here is the second example L-system:

base (5)!(100)F
rule (x)F (x)F ![+2x/3F<g(5)L>][!−2x/3F<g(5)L>]
(x)F ![−2x/3F<g(5)L>][!1x/2F<g(5)L>]
(x)F ![+2x/3F<g(5)L>][!1x/2F<g(5)L>]

(Note, I wrote the three alternate replacements on 3 lines, just so it would fit on this paper – this is not how the file is organized.)

And the observations we made about it:

- Turns, forwards, leaf-sizes, and widths are modified by modifiers.
- The structure of the tree is stochastic (we branch forward and left, or forward and right, or left and right).
- Distances are decreased as we branch.
2.4 How to make a mess of going forward without making a mess of going forward

We want each forward to be based on the true forward, so we must keep track of the locations we would have visited if we were doing plain old forwards. Why? Because we want to make sure we don’t inadvertently change the size or shape of our shape.

- Ask the turtle its location and save it in variables \( x_0 \) and \( y_0 \).
- Pick up the turtle and move forward by the appropriate distance.
- Ask the turtle its location and save it in variables \( x_f \) and \( y_f \).
- Draw the line in the appropriate style.
- Pick up the turtle and move it to \((x_f, y_f)\).

If your NPR style changes the orientation of the turtle while it is drawing the line, you will want to make sure you change it back.

Suppose we want to make a dashed style. We should have a dashLength field that indicates the length of each dash. Then our dashed line should have dashLength dashes separated by spaces of length dashLength. If twice the dashLength doesn’t divide evenly into the total distance, then we need to be careful to put the turtle at its proper location.

```python
num_dashes = int(math.floor(distance/self.dashLen))
(x0, y0) = turtle.position()
turtle.up()
turtle.forward(distance)
(xf, yf) = turtle.position()
turtle.goto(x0, y0)
for i in range(num_dashes / 2):
    turtle.down()
    turtle.forward(self.dashLen)
    turtle.up()
    turtle.forward(self.dashLen)
turtle.goto(xf, yf)
```