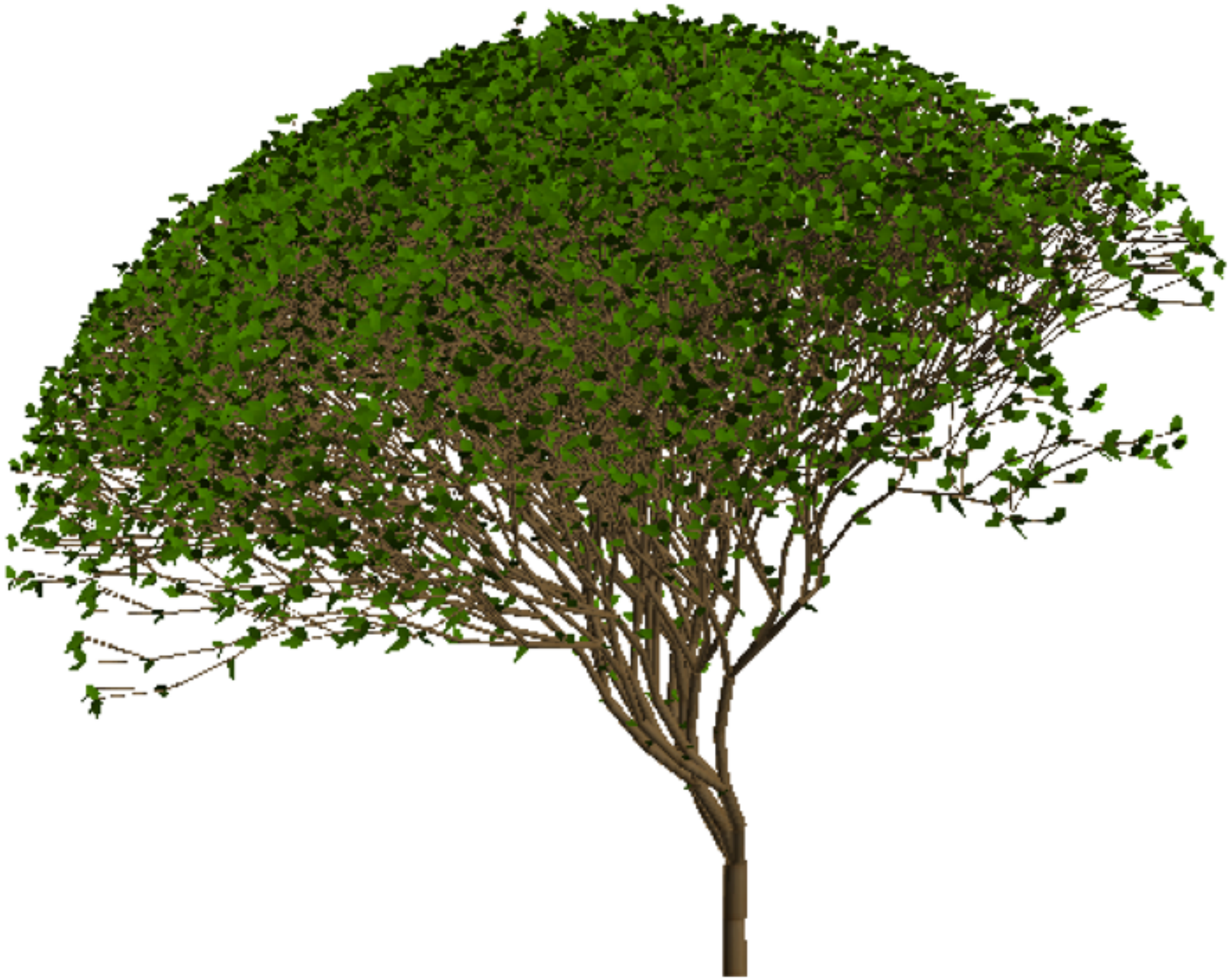


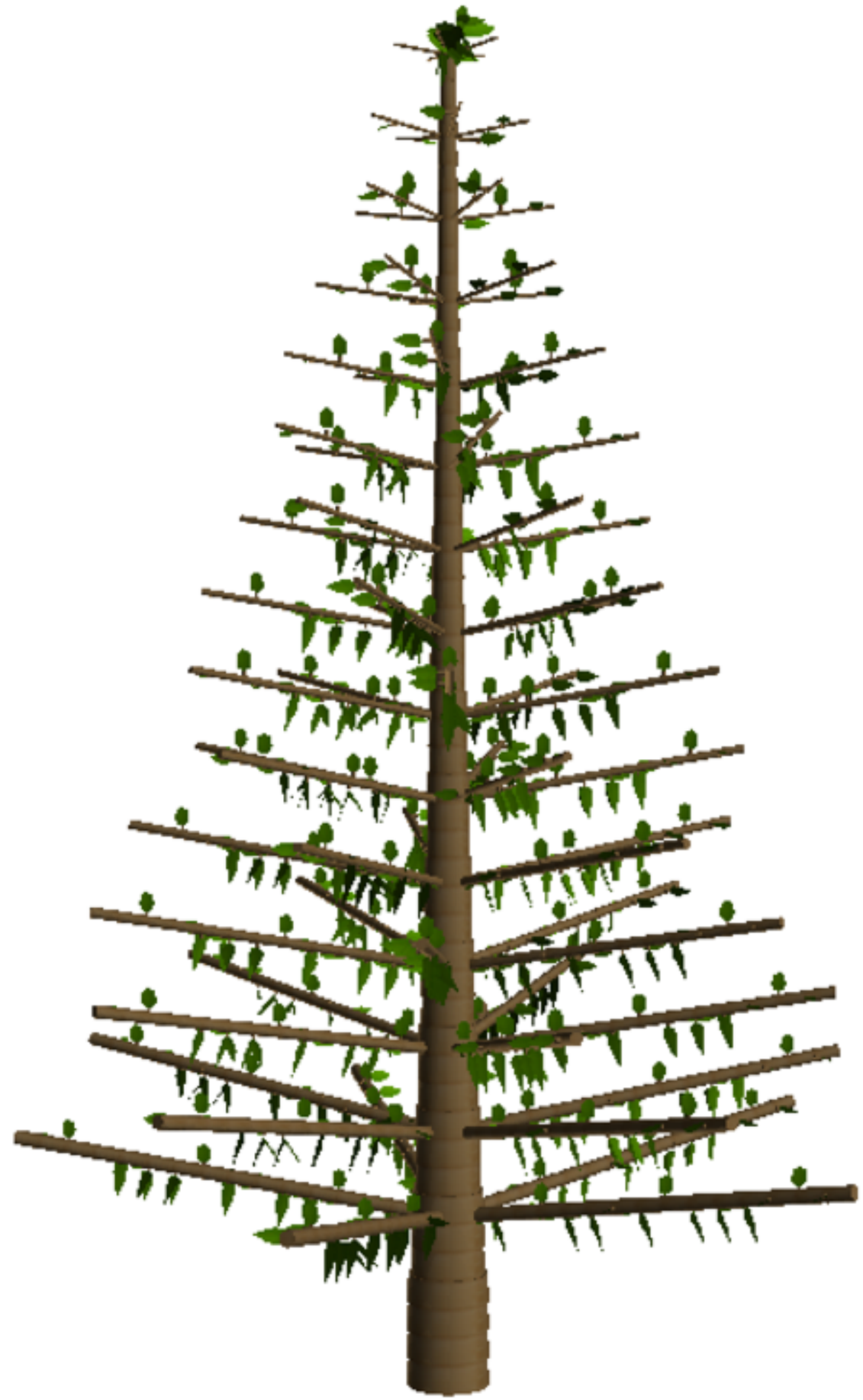
Generating Trees Using

L+ System

Abbas Naderi (abiusx@cs.virginia.edu)

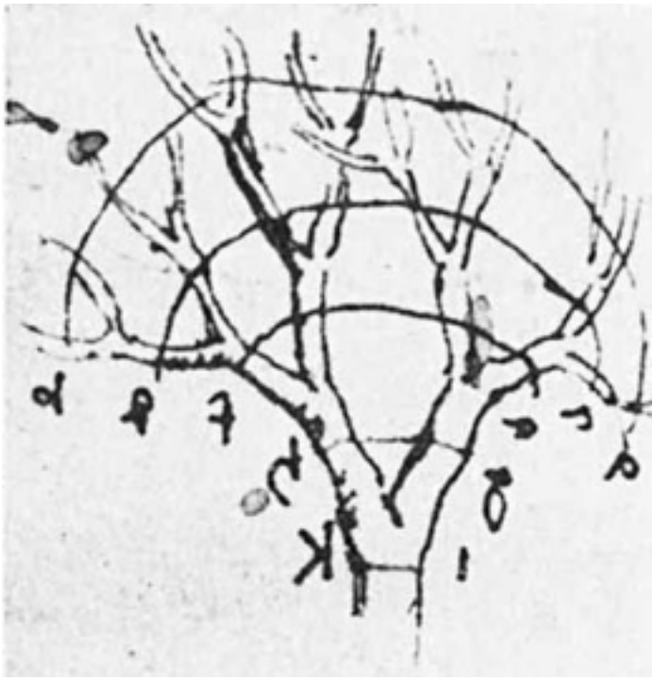
Covered Topics

1. Modeling Trees
2. Turtle Graphics
3. L-System
4. Building Blocks
5. L+System
6. Optimizations



Modeling Trees

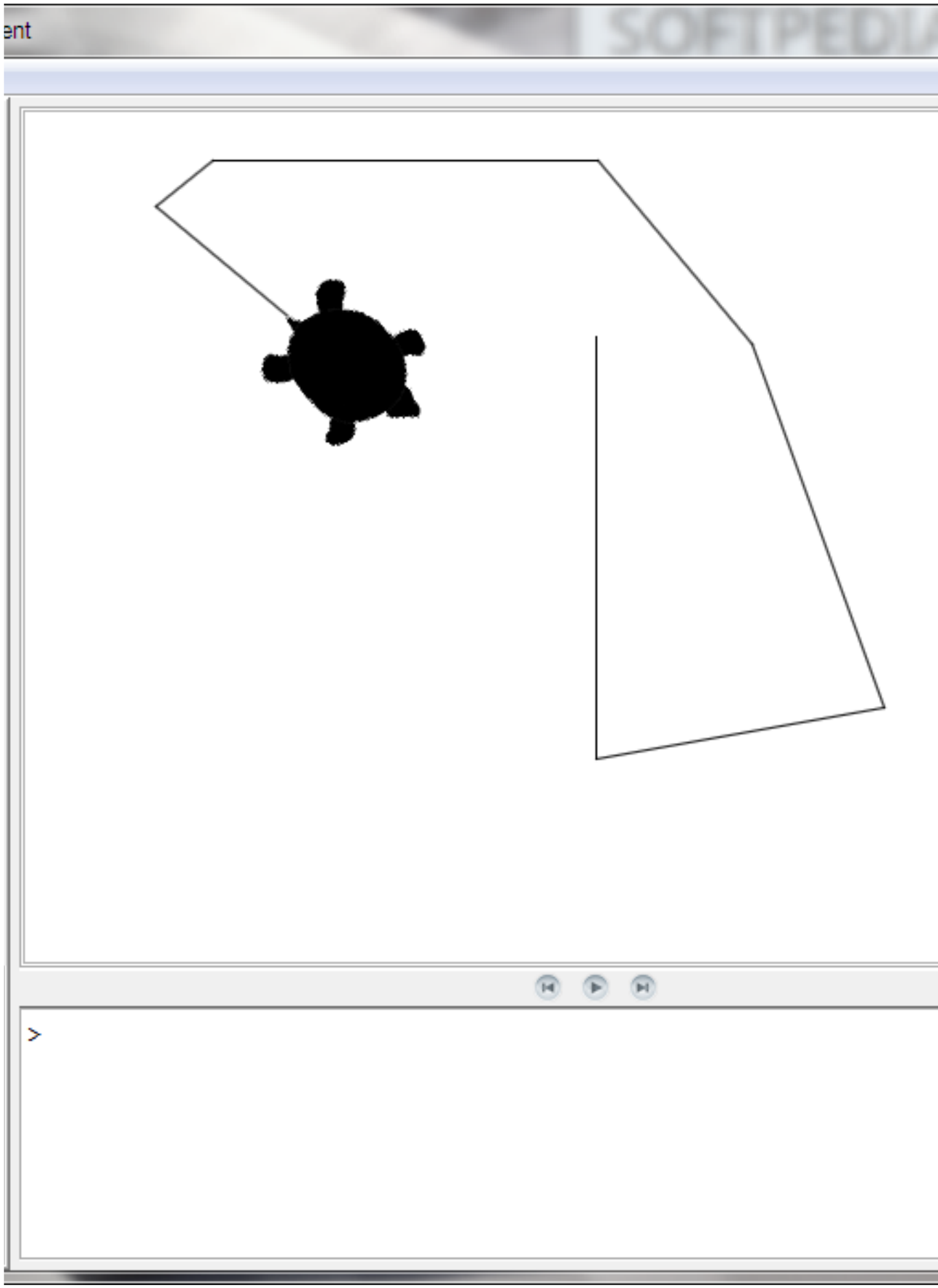
- Trees have a lot of details, and are organic
- Not easy to model using mathematical formulas
- Most trees and bushes have fractal-like behavior
- Can be modeled using simple rules



Trees are fractals

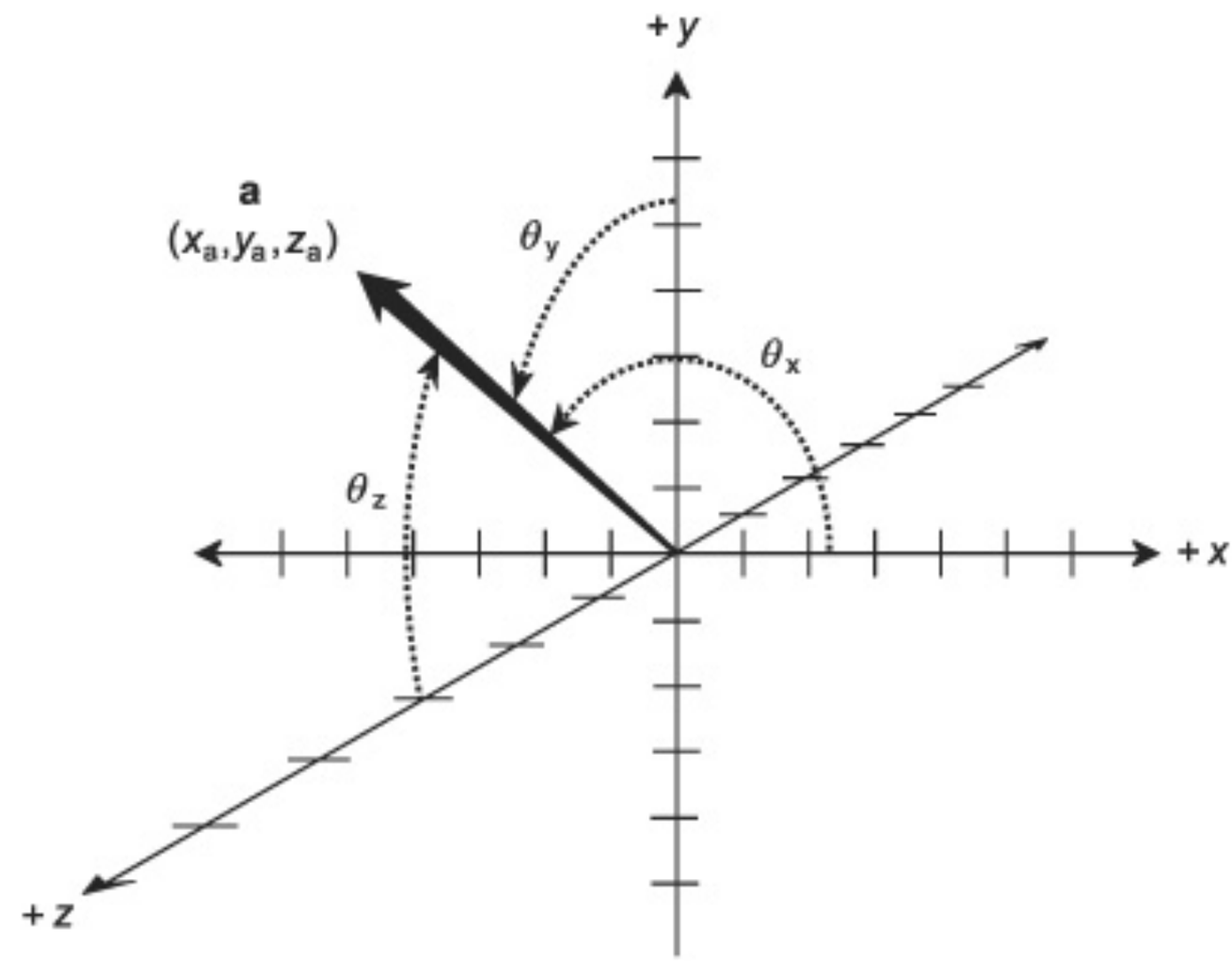
Turtle Graphics

- A simple model used to teach kids computer graphics
- Very effective for making fractals
- Drawing operations are not absolute, but are relative to turtle's position and direction



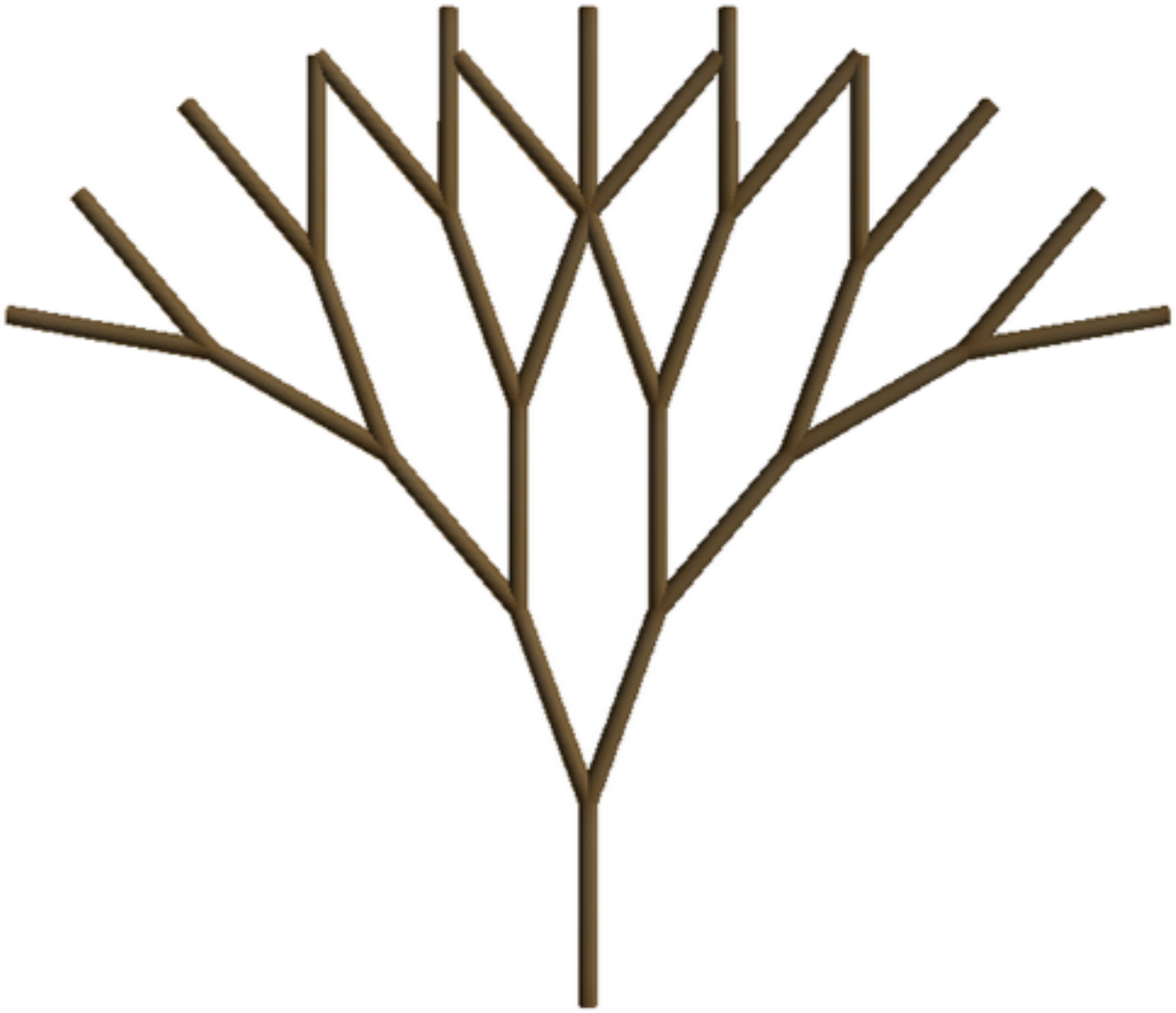
Turtle Graphics (2)

- Turtle starts by looking towards Y axis at position 0.
- Supports a list of moves:
  - Move forward
  - Turn Right/Left
  - Pitch Down/Up
  - Roll Left/Right



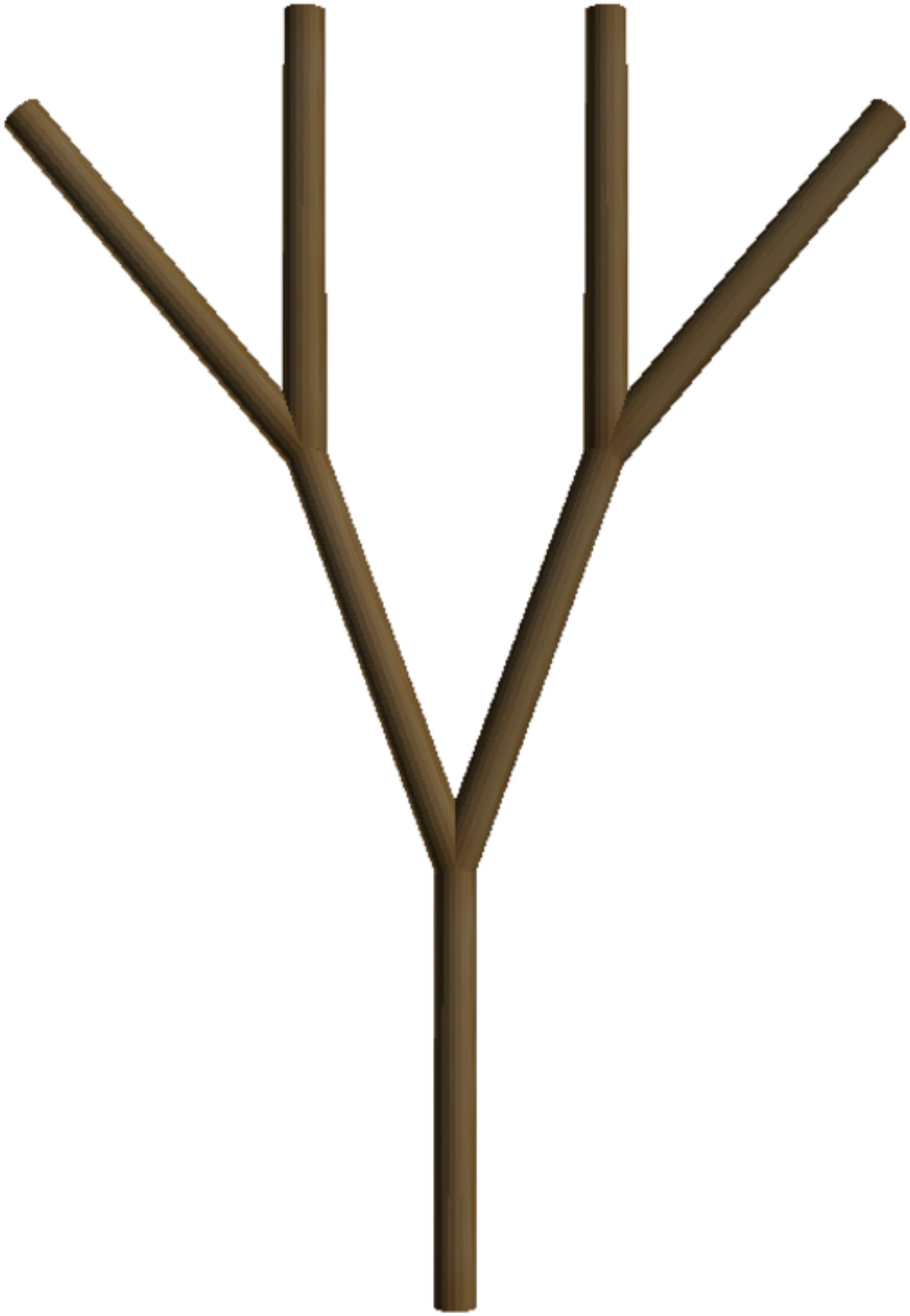
Turtle Graphics (3)

- Turtle can save its state (position + direction + pen) and return to it later.
- Because fractals are recursive structs!
- Consider this rule:
  - Tree= fX
  - X=[+fX][-fX]



L-System

- A string can represent a tree:
- f[+f[+fA][-fA]][-f[+fA][-fA]]
  - f= draw a branch and move forward
  - [= save state
  -] = restore state
  - + = turn right
  - - = turn left

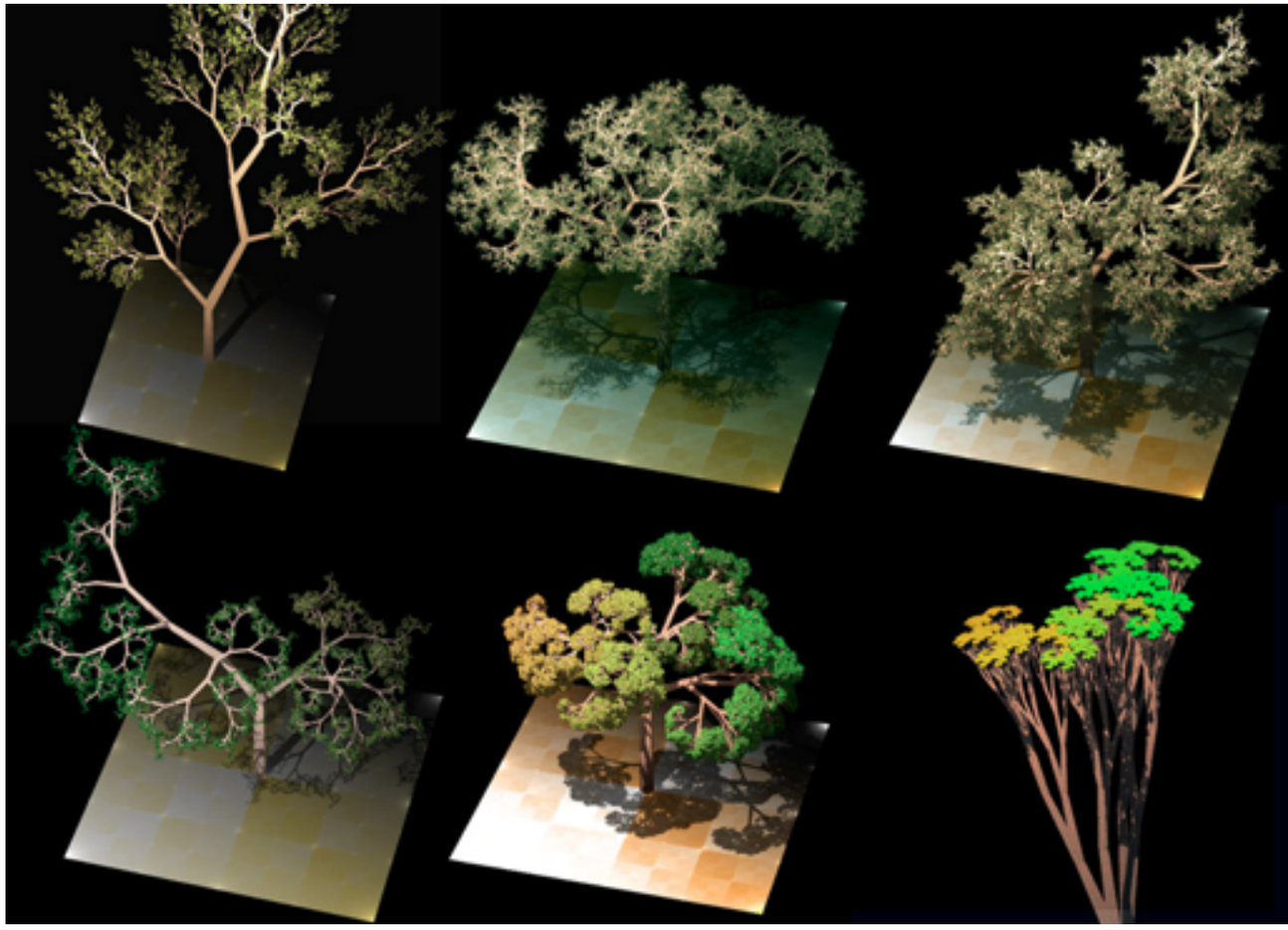


L-System (2)

- Instead of passing around the string (its big for real trees!), lets generate it by a set of rules!

- The set of rules, and the system turning them into a giant tree string, is called L-System

- Named after Aristid Lindenmayer, a Hungarian biologist



L-System (3)

- L-System is similar to a grammar, the only difference is that all rules are applied in each iteration (instead of one)

- Starts with an axiom, populates it with a set of rules in a number of iterations:

```
function populate(string axiom, array rules, int iteration)
{
    if (iteration==0) return axiom;
    foreach (rules as key->value)
        axiom=str_replace(axiom,key,value);
    return populate(axiom, rules, iteration-1);
}
```

L-System (4)

- L-System also needs a parser to consume a tree string, converting tokens to commands passed to a turtle.

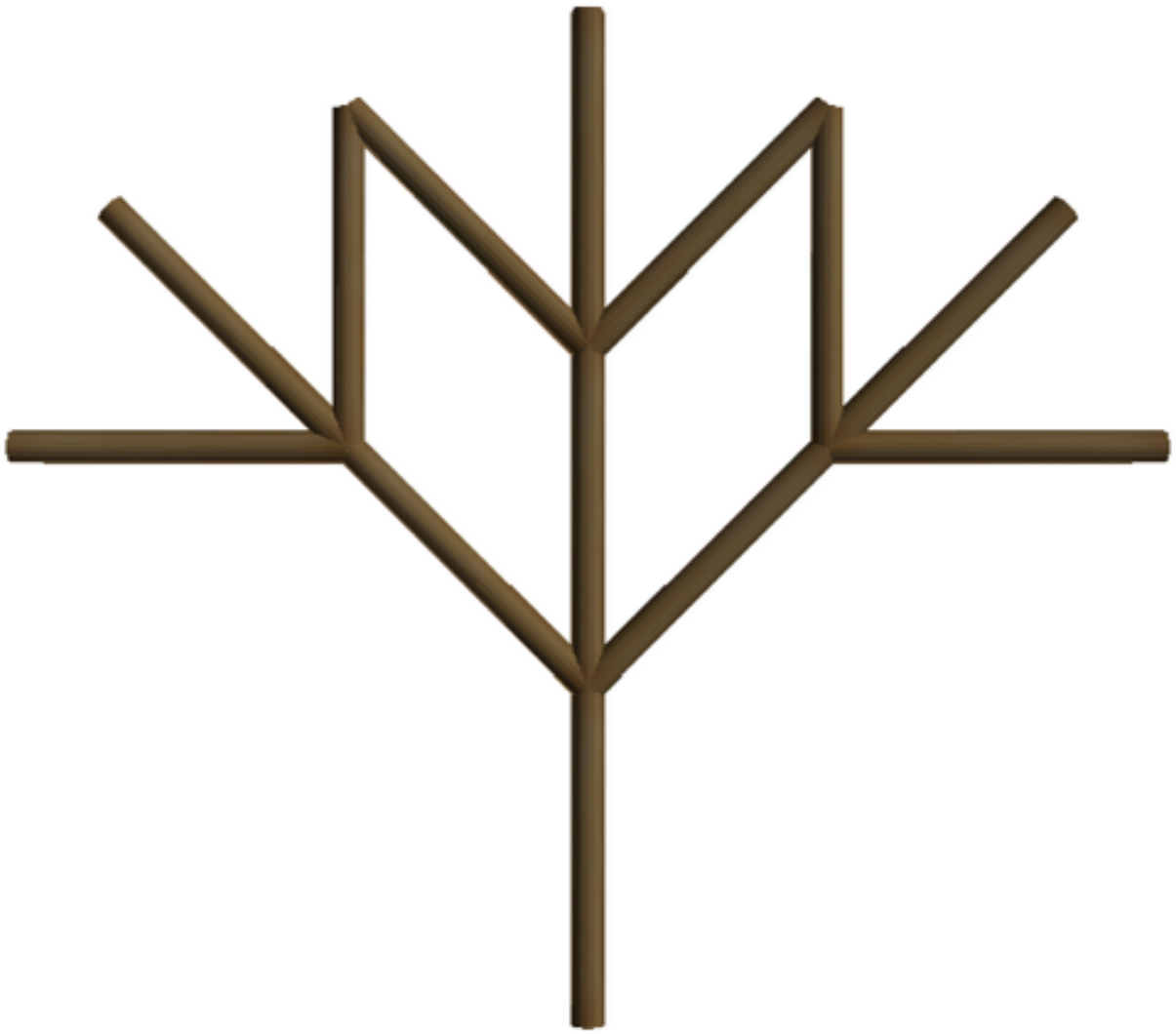
- Rules are typically stored in a *.L file, in following format:

1. number of iterations
2. default rotation angle
3. branch thickness/length ratio
4. axiom
5. rules (one per line)

L-System (5)

- Sample L file:

```
2
45
5
fA
A=[fA][+fA][-fA]
@
```



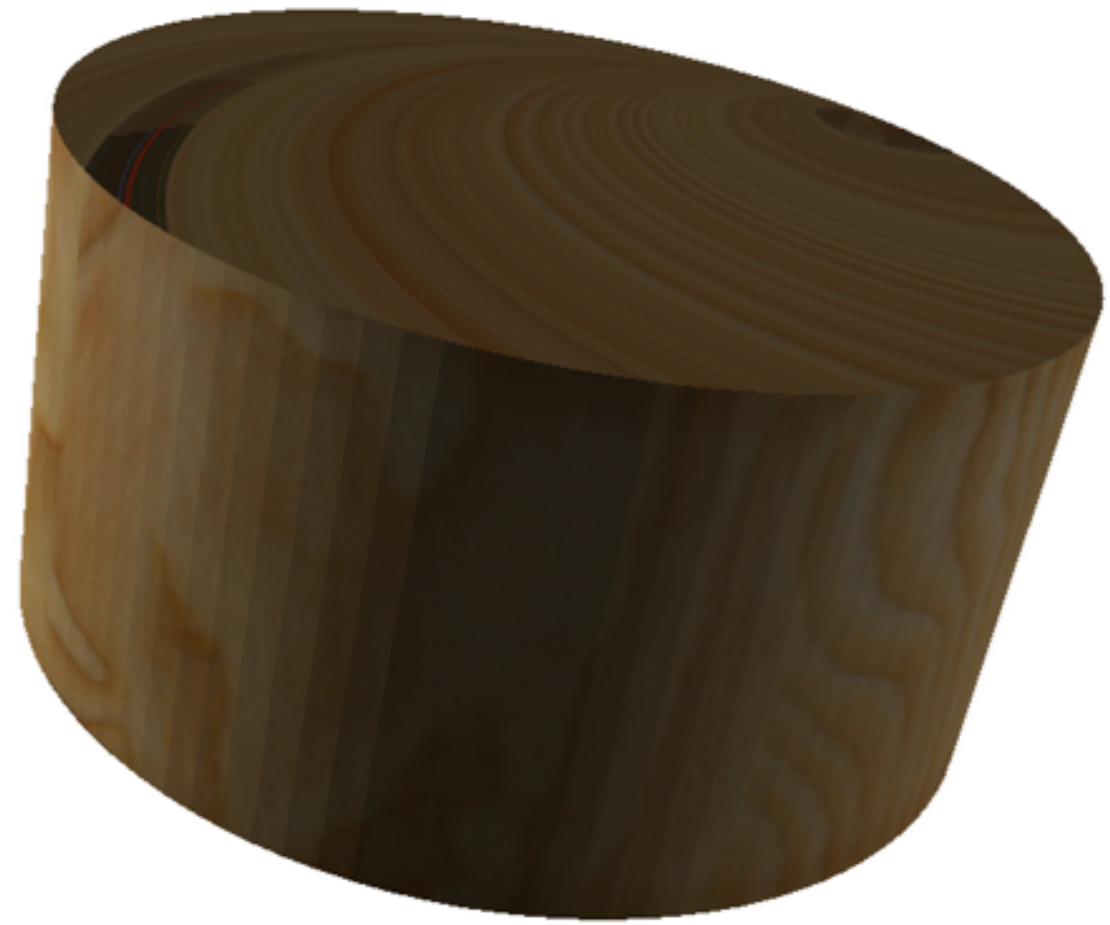
L-System (6)

- More complicated L file:

```
#fractal plant, from wikipedia
6
35
15
X
X=F-[[X]+X]+F[+FX]-X)
F=FF
@
```

Building Blocks

- Trees are made of branches (and leaves)
- Branches are made of wood cylinders
- The cylinders shrink as they progress
- A cylinder is two circles, with linkes between their vertices



Building Blocks (2)

- Trees also have leaves.
- Leaves are small and plentiful, so not much detail is needed
- Made by 9 manual vertices
- Pulled by gravity

L+System

- L-System lacks a lot of things, introduced in L+System!
  - It doesn’t support leaves. (* in L+)
  - It doesn’t support shrinkage
  - It doesn’t support Stochastic Trees!
  - And a lot more

L+System (2)

- Stored in L++ files (backward compatible)
- Operates on OFF+ files (texture coordinates and material)
- > and < decrease and increase branch thickness
- = sets thickness, and % defines shrinkage
- * draws a leaf!

```
switch (command)
{
    case '+':
    turtle.turnLeft(num);
    break; case '-':
    turtle.turnRight(num);
    break;  case '&':
    turtle.pitchDown(num);
    break;  case '^':
    turtle.pitchUp(num);
    break;  case '<':
    turtle.thicken(num);
    break;  case '\\':
    turtle.rollLeft(num);
    break;  case '/':
    turtle.rollRight(num);
    break;  case '>':
    turtle.narrow(num);
    break;  case '%':
    turtle.setReduction(param);
    break;  case '=':
    turtle.setThickness(param);
    break;  case '|':
    turtle.turn180(param);
    break;  case '*':
    turtle.drawLeaf(param);
    break;  case 'F':
    case 'f':
    turtle.draw(param);
    case 'G':
    case 'g':
    turtle.move(param);
    break;  case '[':
        turtle.save();
    break;  case ']':
        turtle.restore();
    break;
```

L+System (3)

- Sample L++ file:

```
5
15
20
fA
A=^f >(30) B\\B\\\\\B
B=[^^fL \\\\\\A L  ]
L=[^(60) [*(.3)] +(50)*(.28)]
L=[^(60)*(.3)]
L=[&(70)*(.3)]
@
```

L+System (4)

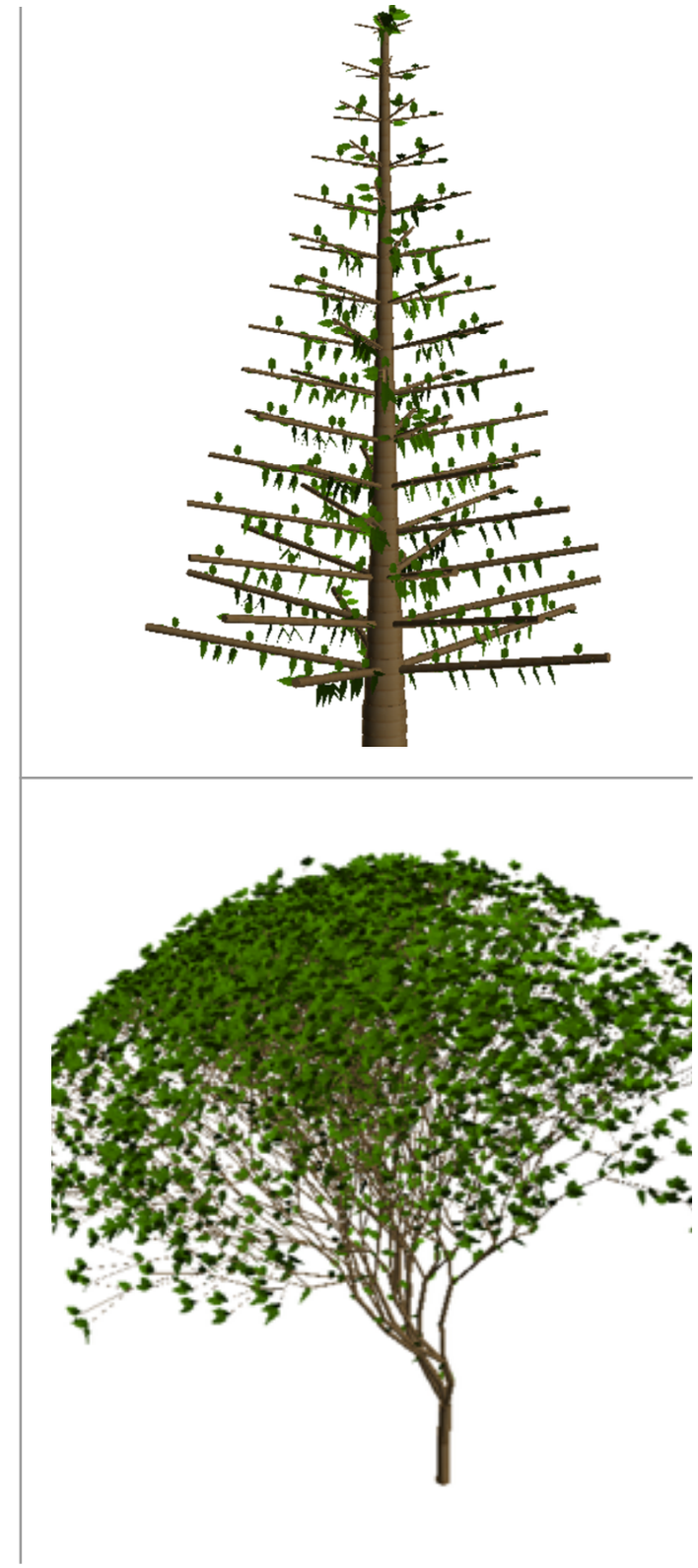
- More complicated L++ file:

```
64
20
10
^(90)Z
Z=^(15)+fB+fB+fBZ
B=[>(20)X]
X=-(90)f(.5)L
X=-(60)&f(.4)[++f(.2)L][--f(.2)L]
X=(-20)&&f(.2)
L=^(80)*(.6)
@
```

Optimizations

- Cylinder and leaf are cached (multiple versions)
- Texture coordinates are static
- Parsing and drawing code is optimized
- Millions of vertices are generated in less than a second!

https://github.com/abiusx/L3D



L+System (by Abbas Naderi)