Added Realism, Continued

Fractals, Continued

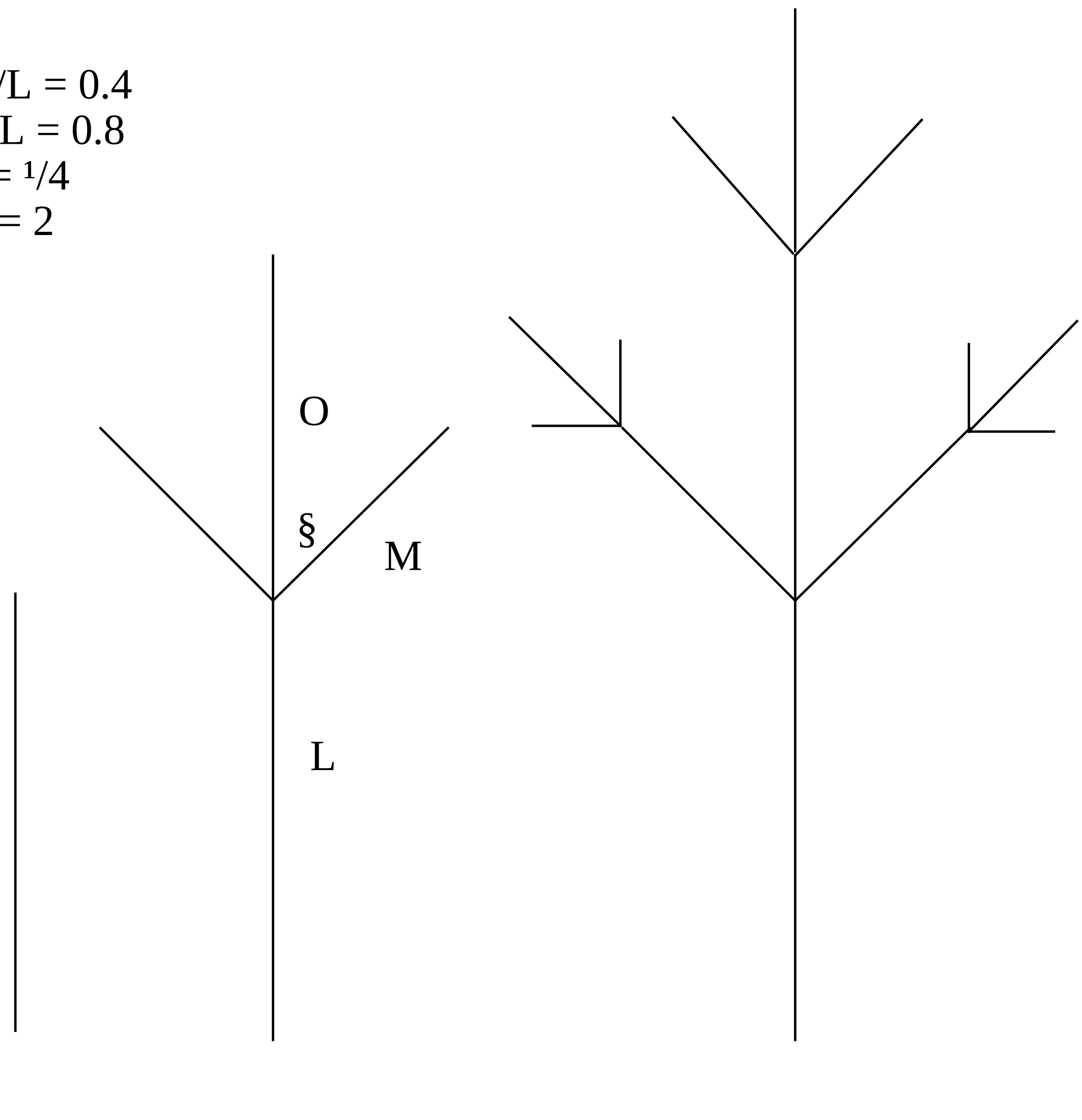
Fractal Trees (Peter Oppenheimer)

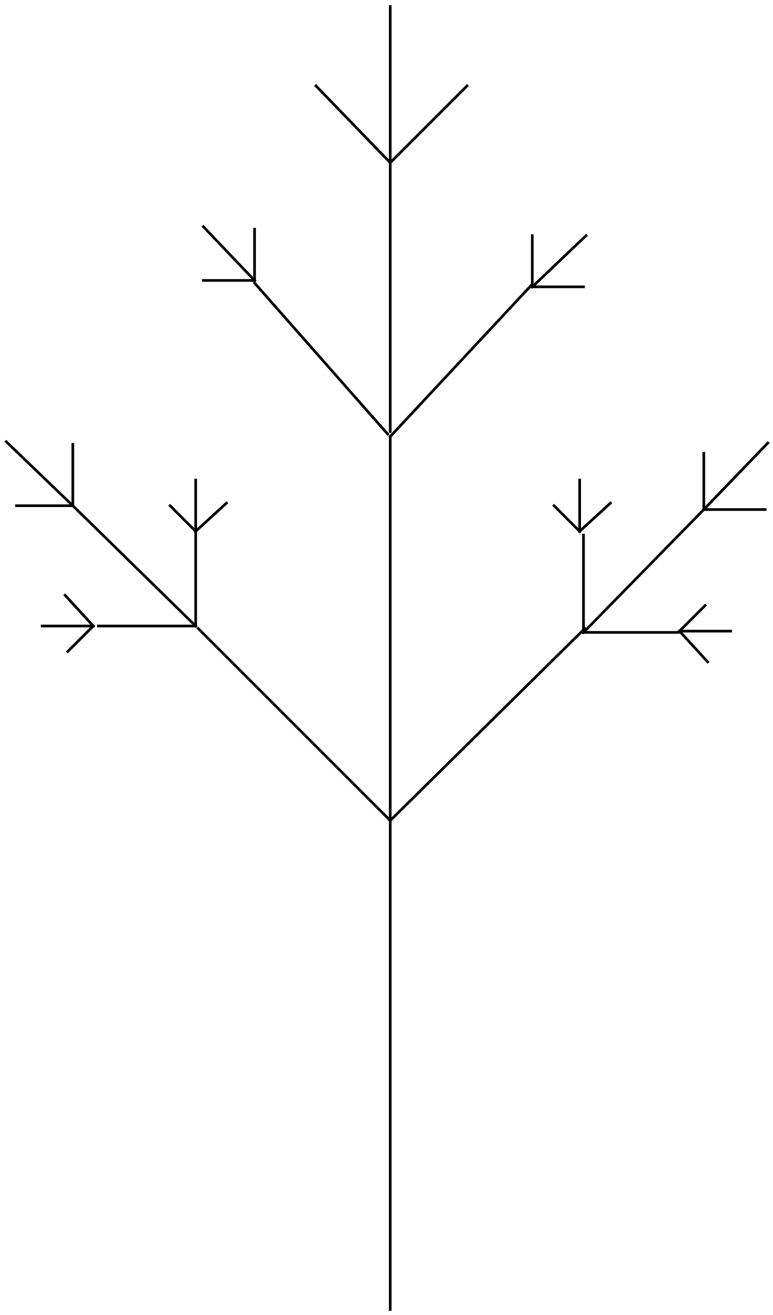
```
tree :=
{
        Draw Branch Segment
        if (too small)
                Draw Leaf
        else
        {
                # Continue to Branch
                {
                        Transform Stem
                        "tree"
                }
                repeat N times
                {
                        Transform Branch
                        "tree"
                }
        }
}
```

Parameters:

- § angle between main stem and the branches
- M/L size ratio of branch to old stem
- O/L size ratio of new stem to old stem
- T rate at which stem tapers
- H amount of helical twist in the branches
- N number of branches per stem segment

M/L = 0.4
O/L = 0.8
§ = ¹/4
N = 2





If parameters don't change at each step

- strictly self similar
  - eg. fern

If parameters change randomly

- statistically self similar
  - eg. juniper - gnarled tree

Stem shape

- we did lines
- cylinders, spiral, helix, squiggle
  - (see figure)

Rendering the fractal

- thick antialiased lines
- texture bump mapping
- texture mapping - bark
- bark by analytical means
  - sawtooth waves modulated by brownian fractal motion
    - (see figure)

Particle Systems

Want to model clouds, smoke, fire, water, etc.

Problems:
- not smooth well defined shapes
- shape changes over time

Solution:
- particle systems
  - (see figures)

How particle systems differ from other representations

1) not set of primitive surface elements
   but rather a cloud of primitive particles

2) not static
   but dynamic - particles change form and move
   new particles born, old particles die

3) not deterministic
   but stochastic shape appearance, etc.

Advantages:

1) particles are very simple-- i.e. point versus polygon
   therefore process more, faster
   therefore can easily motion blur

2) model definition is procedural
   therefore easy to program
   therefore can adjust level of detail easily

3) dynamic
   therefore shape change is possible

To compute each frame of motion sequence

1) new particles generated

2) new particles assigned attributes

3) old particles past life-time killed

4) remaining particles transformed

5) image of living particles rendered

Very general since is procedural

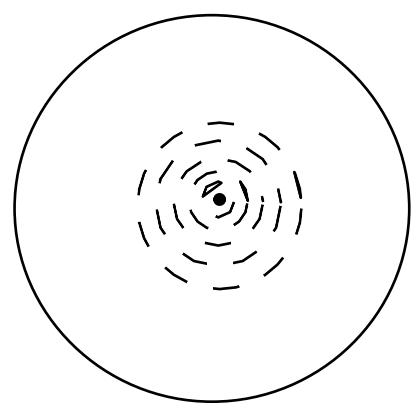
eg step 4 could be solution to partial differential equations
or statistical mechanics or set of rules

Examples for each step

1) a) keep mean number of particles and variance constant
b) number of particles is function of screen size
(don't compute more if too small to see)
2) attributes include: initial position, initial velocity, size, color,
transparency, shape, lifetime
may specify overall shape and position of "cloud" of particles
(maybe just inside of sphere - start in middle and move out)
3) a) extinguish if past lifetime
b) extinguish if intensity is too low
c) extinguish if particles move outside of shape
4) particle dynamics
motion, color, size, transparency may change
5) a) render as points composed with rest of scene
b) render as point light sources
good for fire, explosions, bad for clouds

Prototypical example

- Genesis scene in Start Trek - Wrath of Khan
  - Wall of fire
  - (see figure)
- Two hierarchy particle system
- Top level particles start on sphere
  - First at impact point
  - Then in expanding rings
    - time to generate is function of distance from center
- Motion of particles
  - perpendicular to sphere's surface with an initial velocity
  - gravity pulls particles back to surface
- Average color and rate of color change inherited from parent
  - particle but varies stochastically



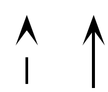
Grass

static - show entire growth in one frame

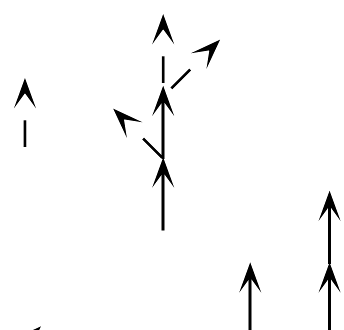
(see figure)

Graftals

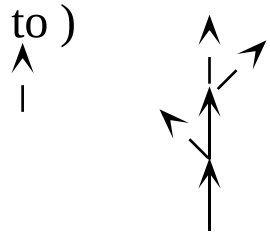
- similar to fractals
- based on formal language techniques: L-systems
- like fractals - "the closer you get - the more it looks the same"
    - but not strictly or statistically self-similar because can't compute D (fractal dimension)
- L-systems
    - Lindenkayer systems
    - parallel rewriting grammars
        - apply productions in parallel
    - example:
        - alphabet { 0, 1, [,], (,)}



left, right



- axiom 0
- production rules
  - 0 to 1[0]1(0)0
  - 1 to 11
  - [to [,] to]
  - (to (,) to)



- so first level 0
- second level 1[0]1(0)0
- third level 11[1[0]1(0)0]11(1[0]1(0)0)1[0]1(0)0

Get data structure

- How render?
  - 0 and 1's as stems - antialiased lines
  -] and) as leafs - antialiased disks

Not a fractal

- is a subfractal
- problem is the 1's in the trunk generation

Graftals

- family of objects generated by parallel graph grammars
- includes many fractals
  - example:
    - could have graphtal representation of Koch curve