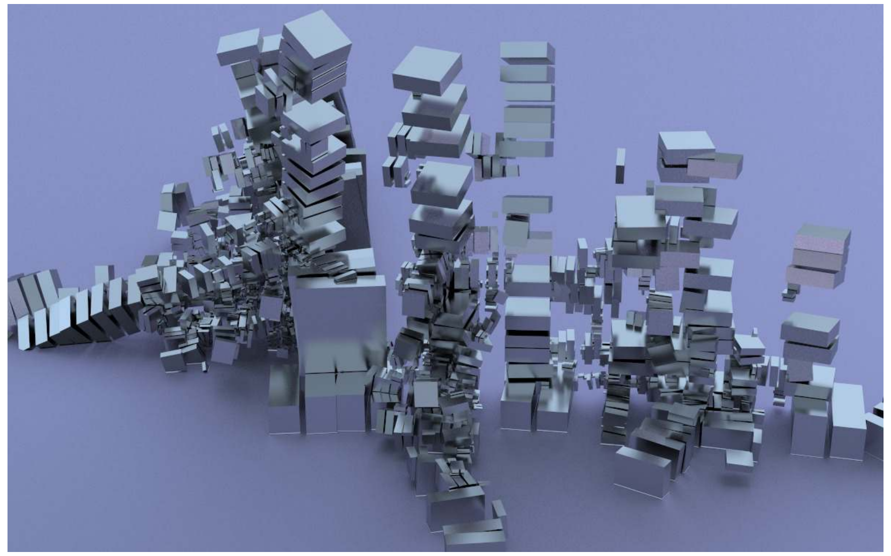
Advanced Computer Graphics
Procedural Modeling



Structure Synth

G. Zachmann
University of Bremen, Germany
cgvr.cs.uni-bremen.de

Global 3D Modeling Pipeline: "Code" vs. "Data"

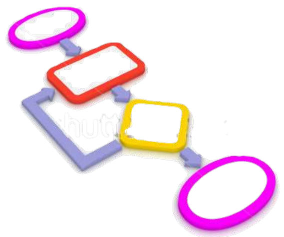
- Acquisition modeling pipeline:

data acquisition → clean-up → build model → consistency check →

- Explicit modeling pipeline:

hand-drawn modeling → consistency check →

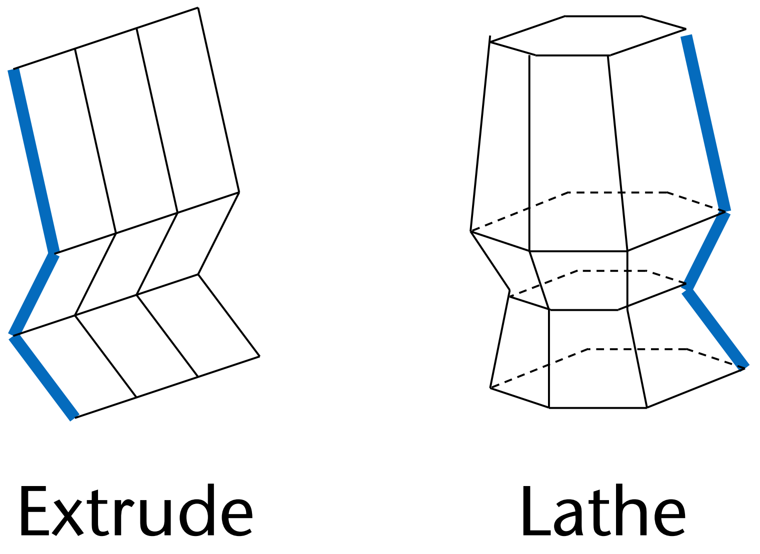
- Procedural modeling approach:

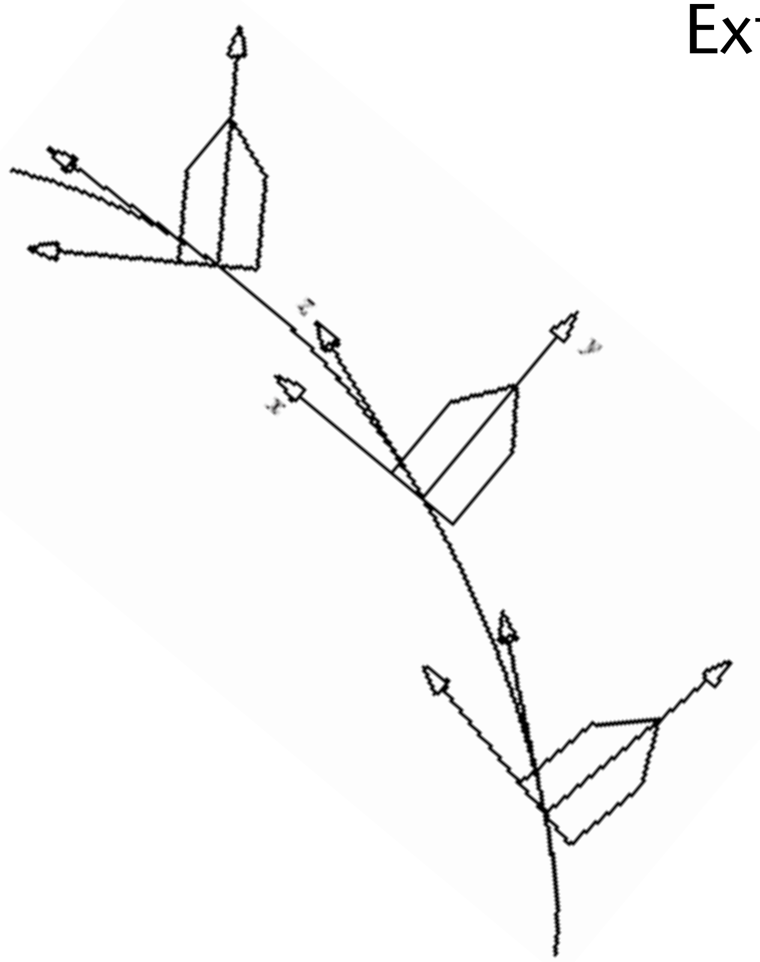


procedural modeling →

Extrusion / Lathing / General Sweep Operation

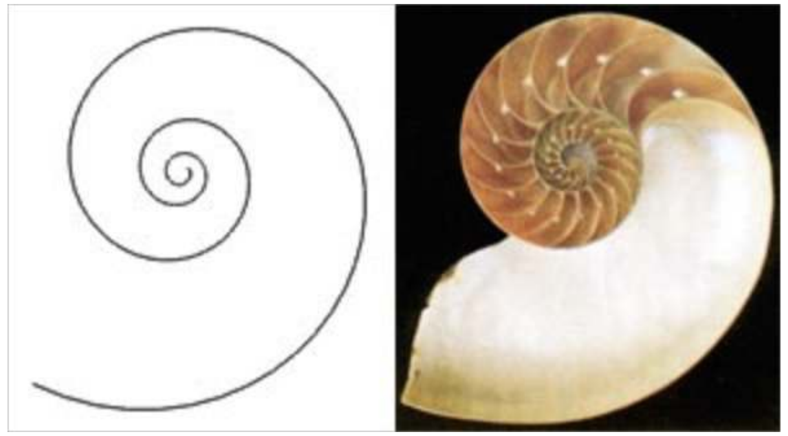
- Many useful shapes can be constructed by extruding or lathing a curve
  - Connect corresponding vertices of several copies of a generating curve
- Generalization: path extrude (sweep)
  - Path given by function **p**(t)
  - Construct the Frenet frame on several points along the path
    - Frenet frame = tangent, normal, binormal
    - Tangent = **p**'(t) ; normal = **p**''(t)
  - Transform generating curve into that frame
    - Transform = 1. Scaling, 2. Rotation(tangent, normal, binormal), 3. Translation





Simple Procedural Model: Seashells

- Growth pattern of seashells is helico-spiral

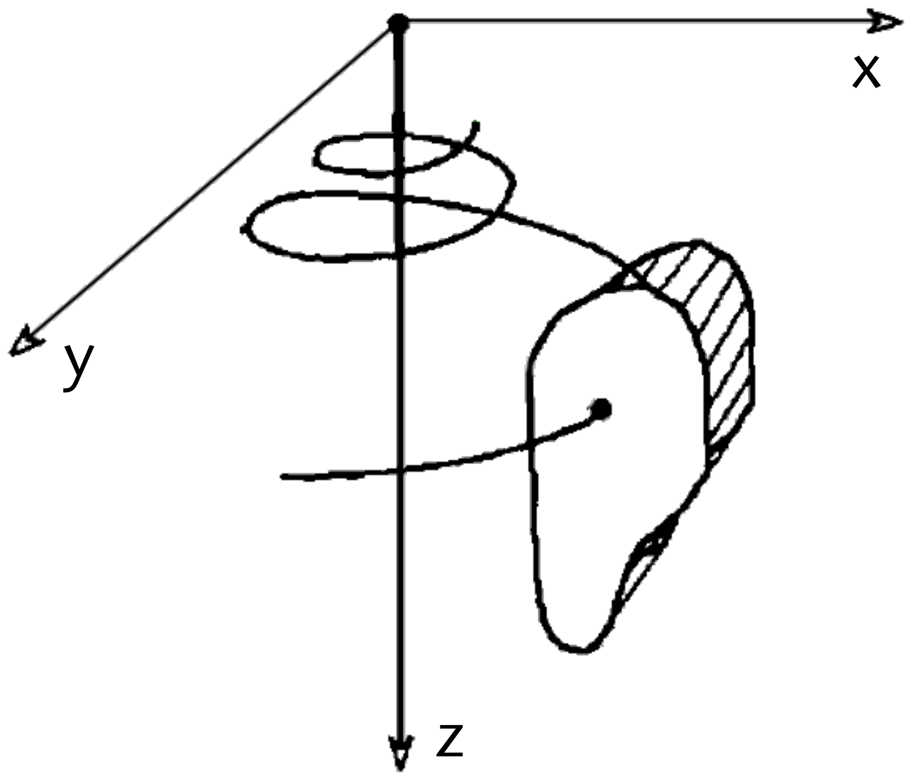


- Structural curve (helico-spiral path):

$$\mathbf{h}(\theta) = \begin{pmatrix} ae^{k_1\theta}\cos\theta \\ ae^{k_1\theta}\sin\theta \\ be^{k_2\theta} \end{pmatrix}$$

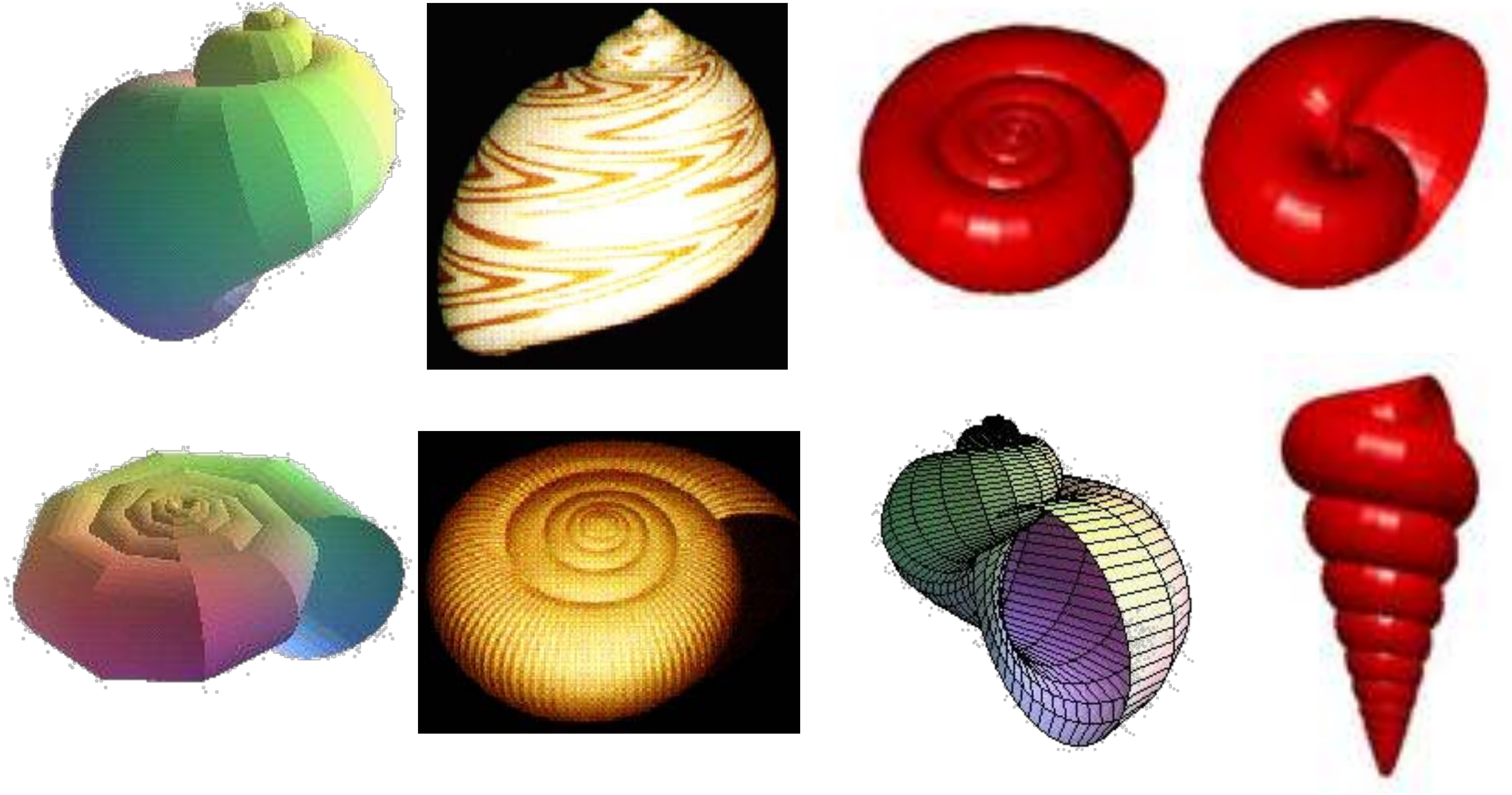
- Generating curve:
  - Modeled by designer (in xz-plane)
  - Sweep along structural curve bytransformation matrix

$$M(\theta) = \mathrm{Transl}(\mathbf{h}(\theta))\,\mathrm{Rot}_z(\theta)\,\mathrm{Scale}(e^{k_s\theta})$$



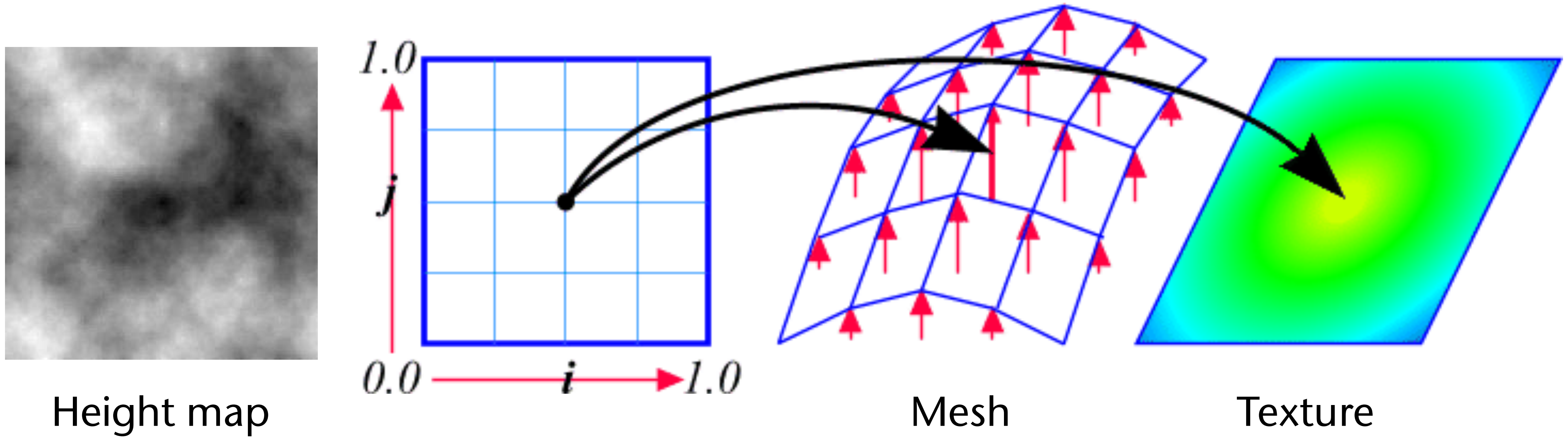
- Increase θ in steps by $\Delta\theta$;
  at each step, connect points on transformed generating curve with corresponding ones from previous step

Examples

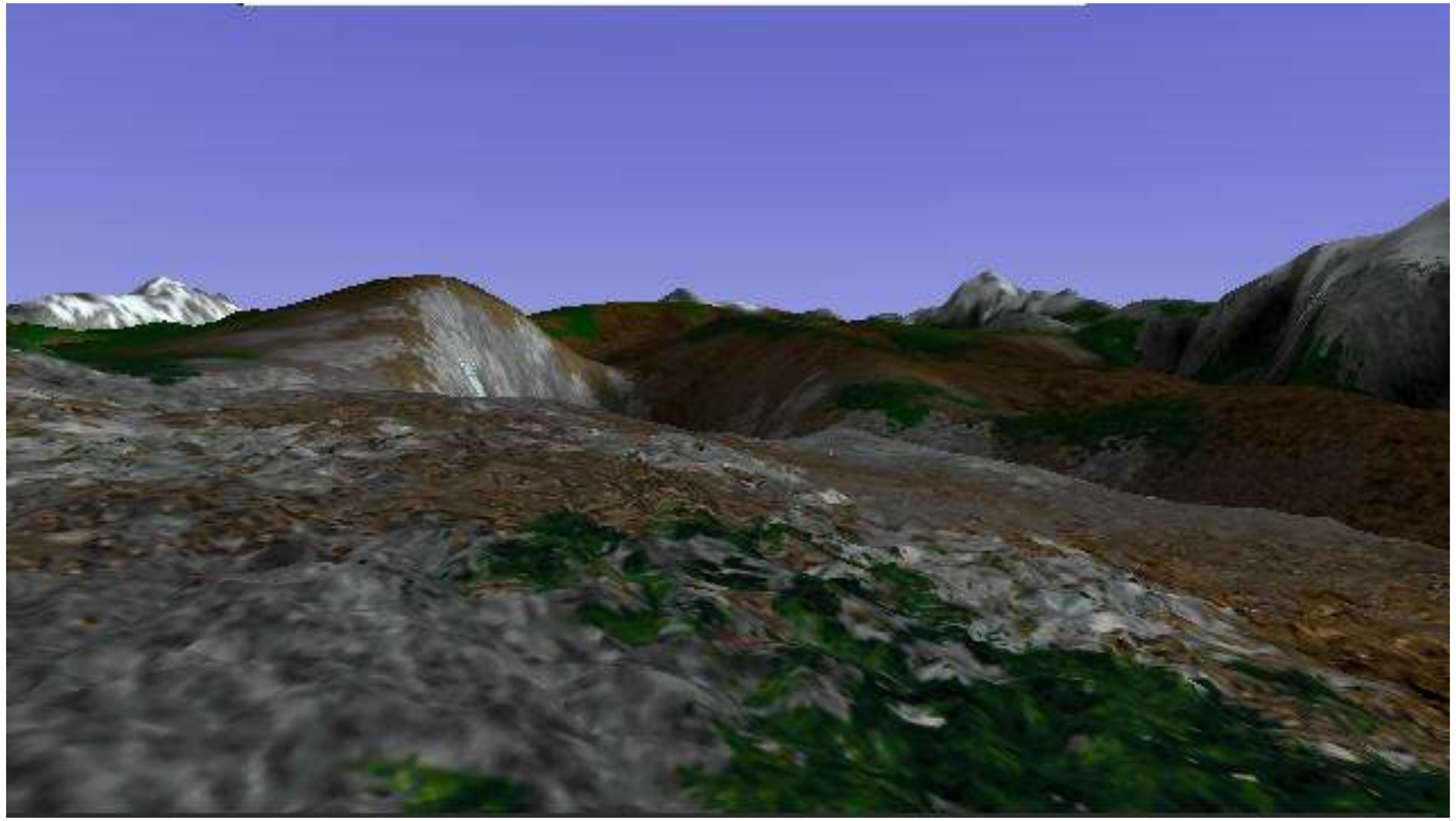
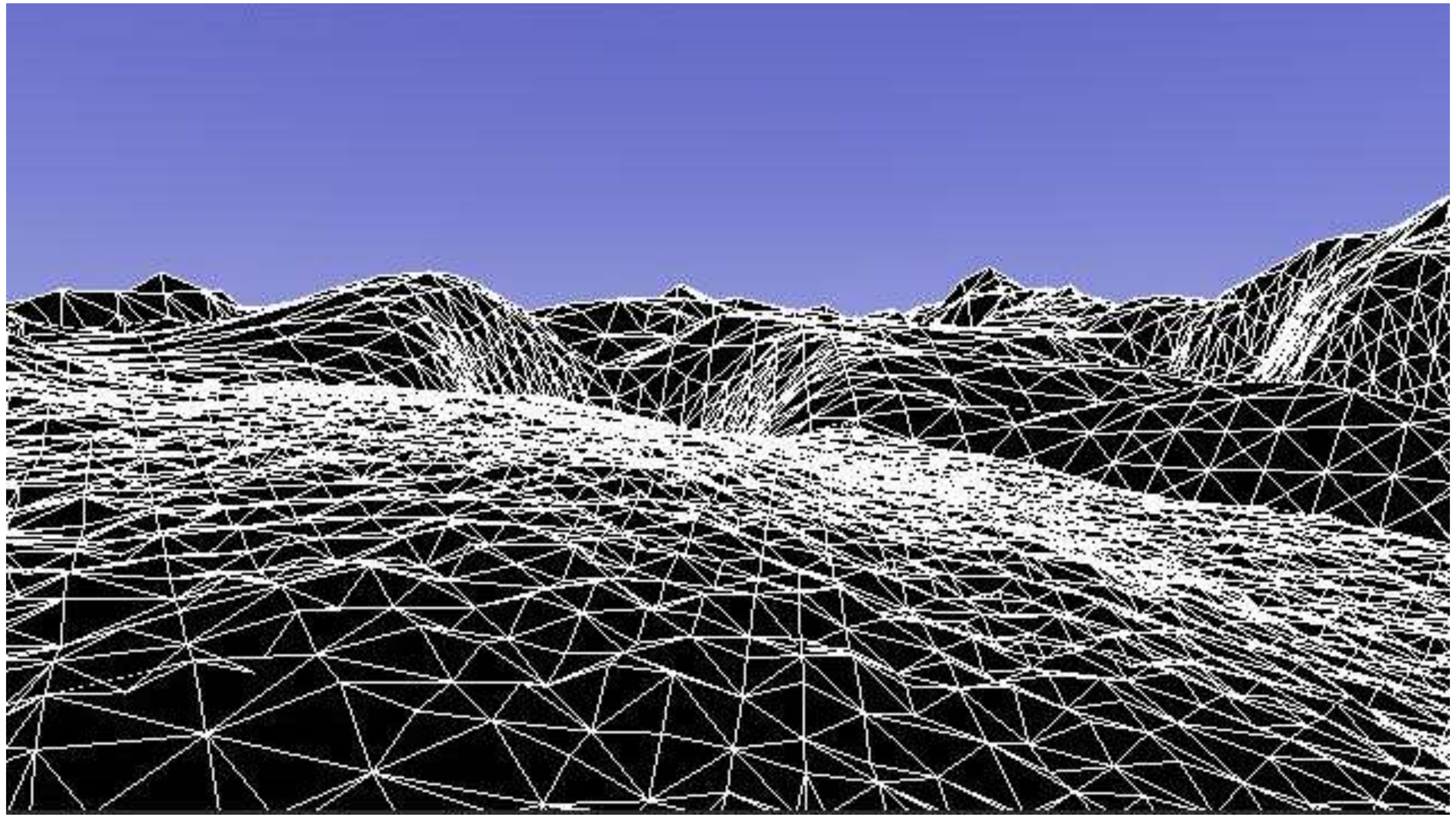


Procedural Terrain Models

1. Generate height map $z = h(x, y)$
2. Convert height map to 3D mesh

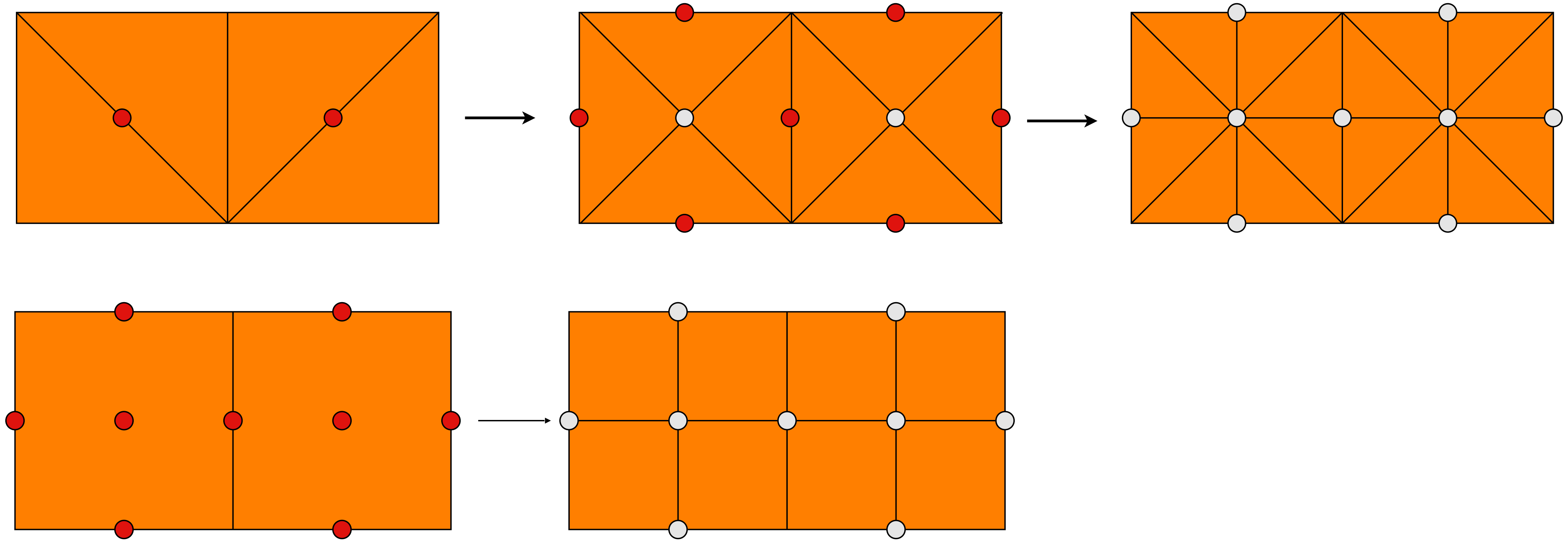


- In the following: just step 1 (step 2 is "straight-forward")
- Food for thought: in reality, step 2 needs view-dependent, adaptive LoD
  - Possible approach: quadtree over height map, traverse until screen-space error is met (cracks at tiles?)

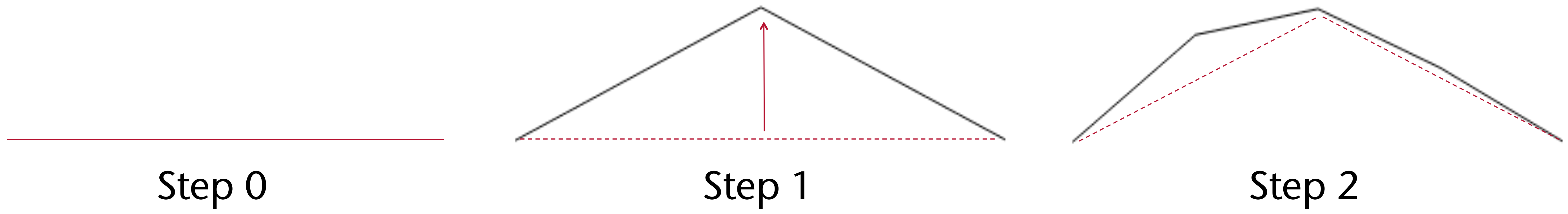
Fractal Terrain Modeling

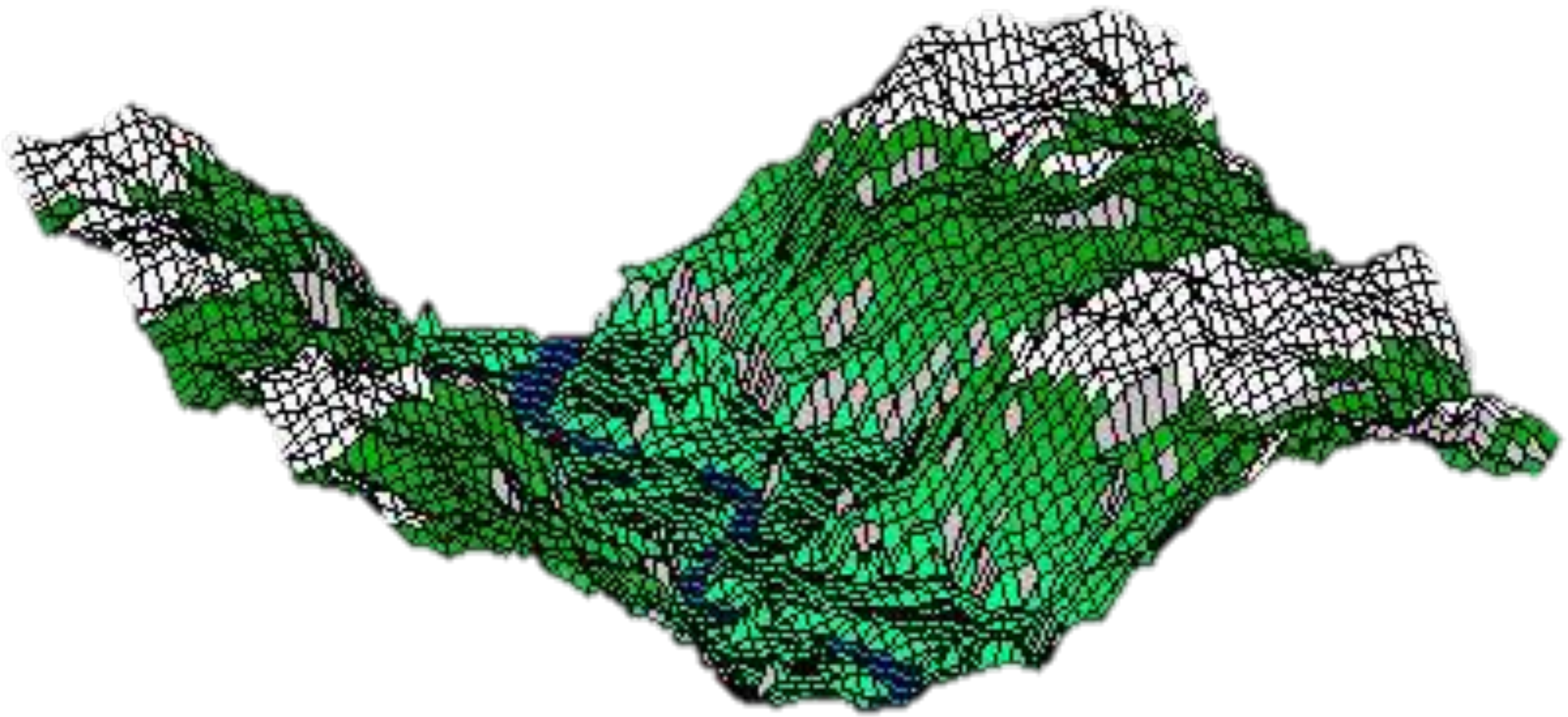
- Start with coarse height map, subdivide successively
- For each new vertex:
  - Compute height, interpolating neighboring (older) vertices (e.g., bilinear)
  - Add random delta to height of new vertices
    - Decrease amplitude with each iteration
- Subdivision methods:



Triangle bintree

1D Example





- Volcanos: flip surface upside down if above maximum value

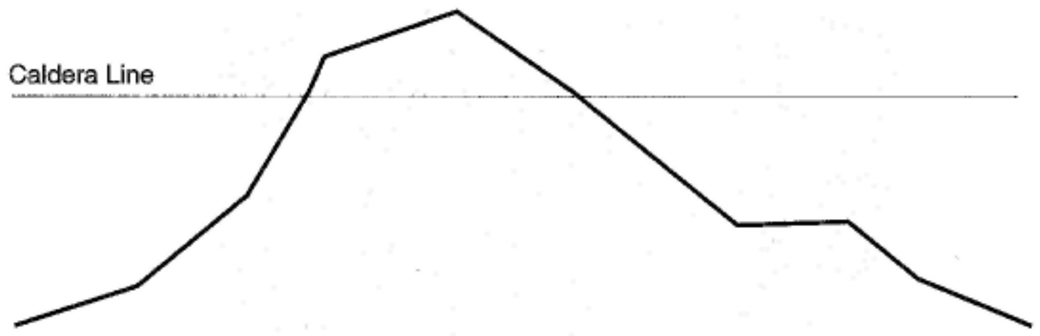


FIGURE 4.19.6 The caldera line.

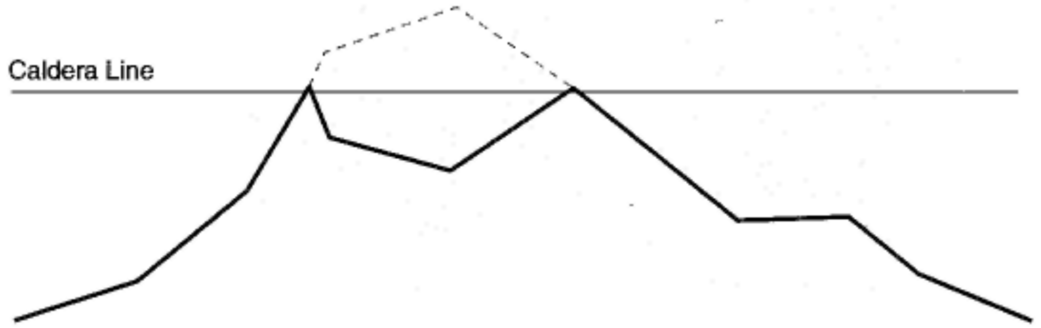
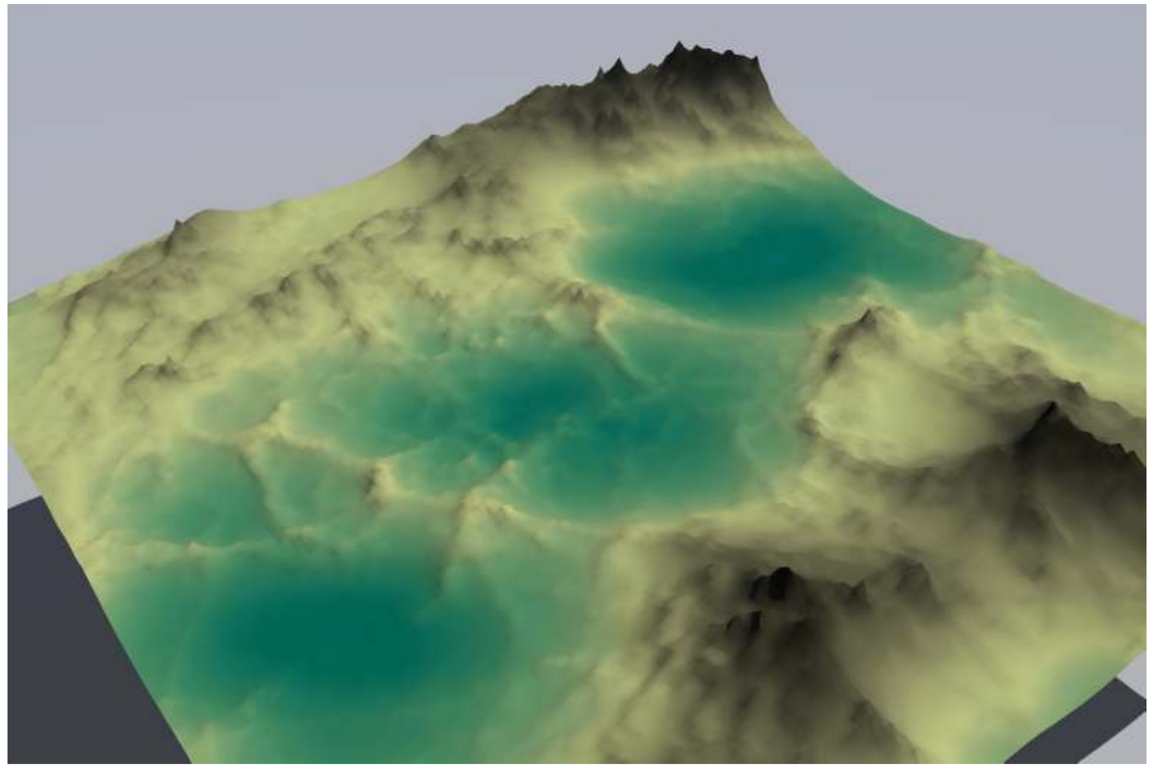
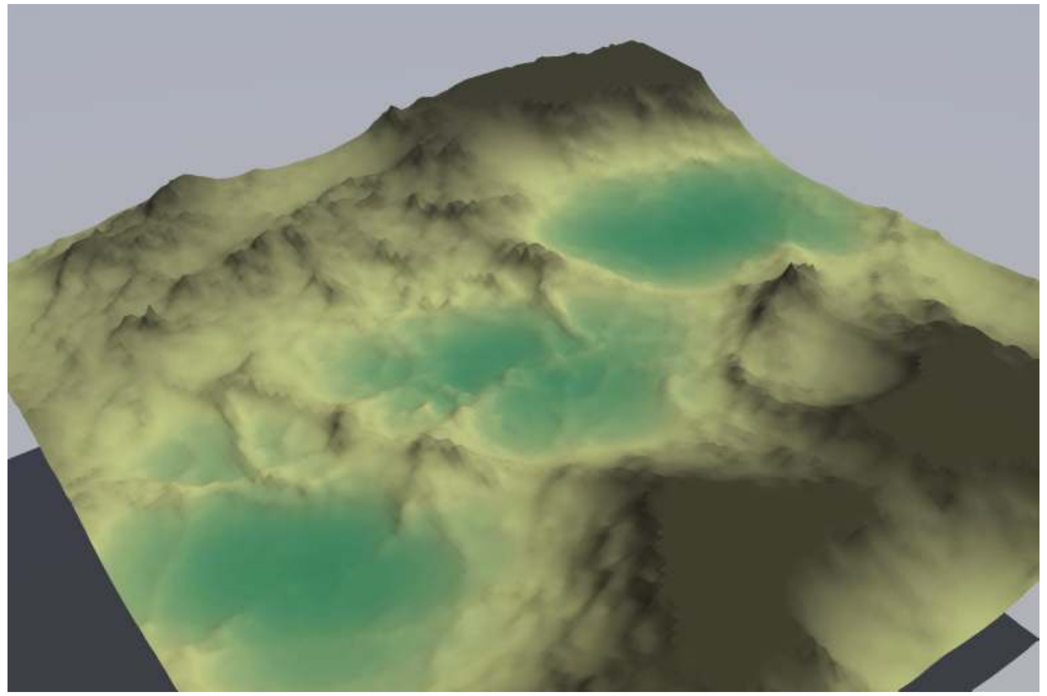


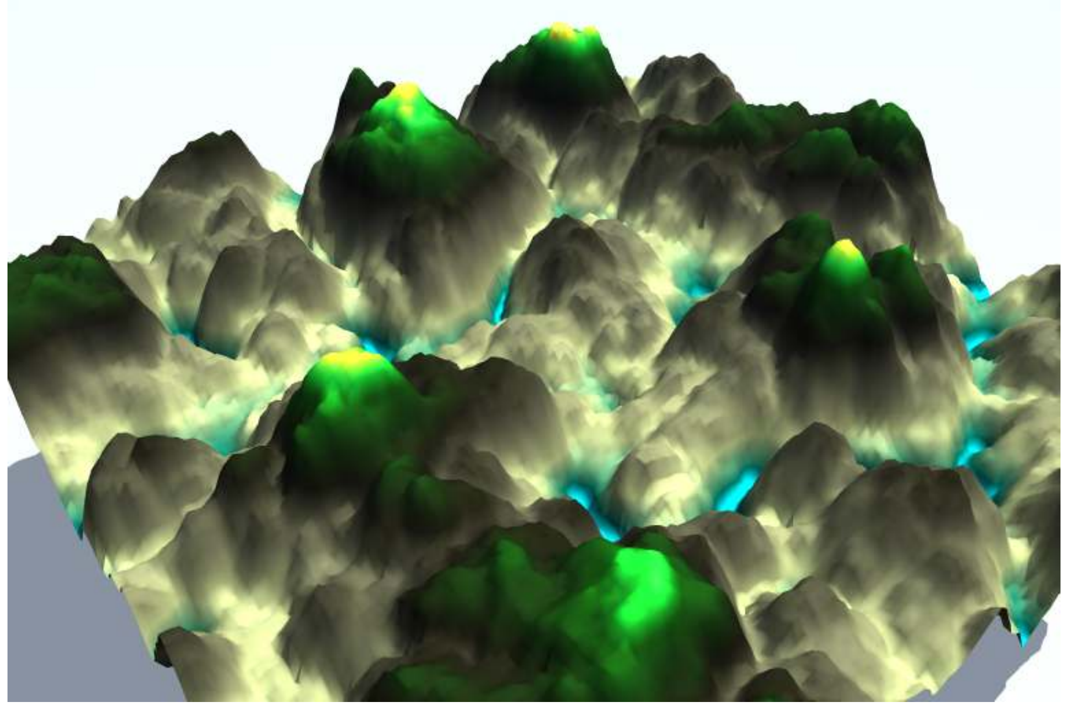
FIGURE 4.19.7 Invert the height field values above the caldera line.

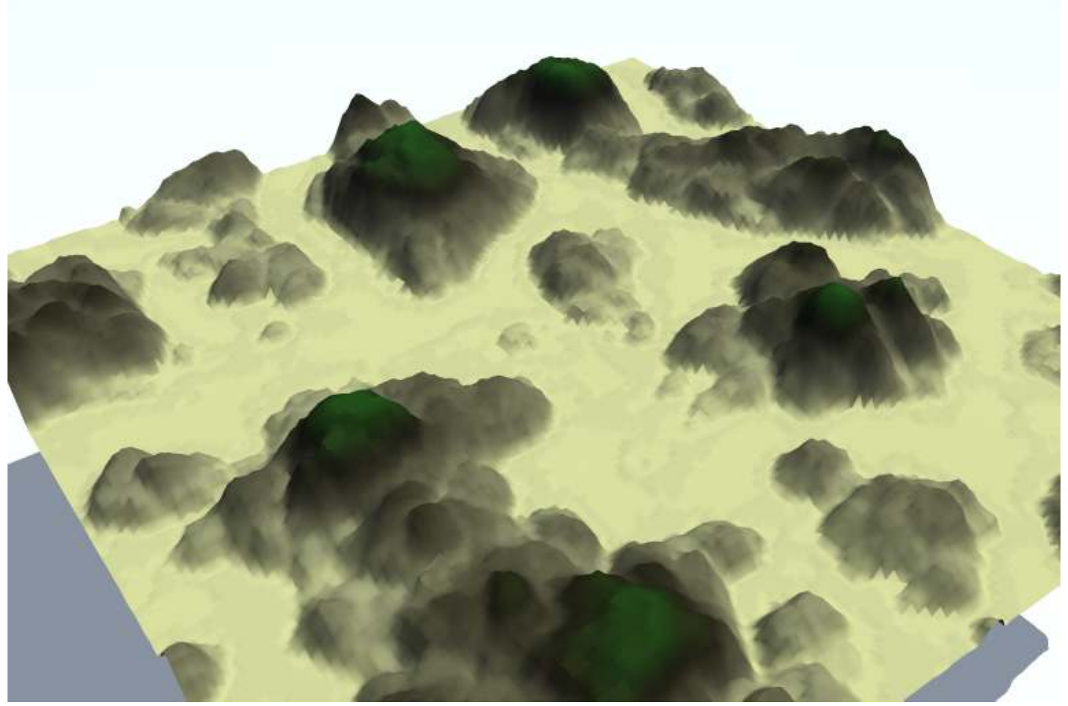
- 1D texture to add vegetation / surface material (e.g., rock, snow)



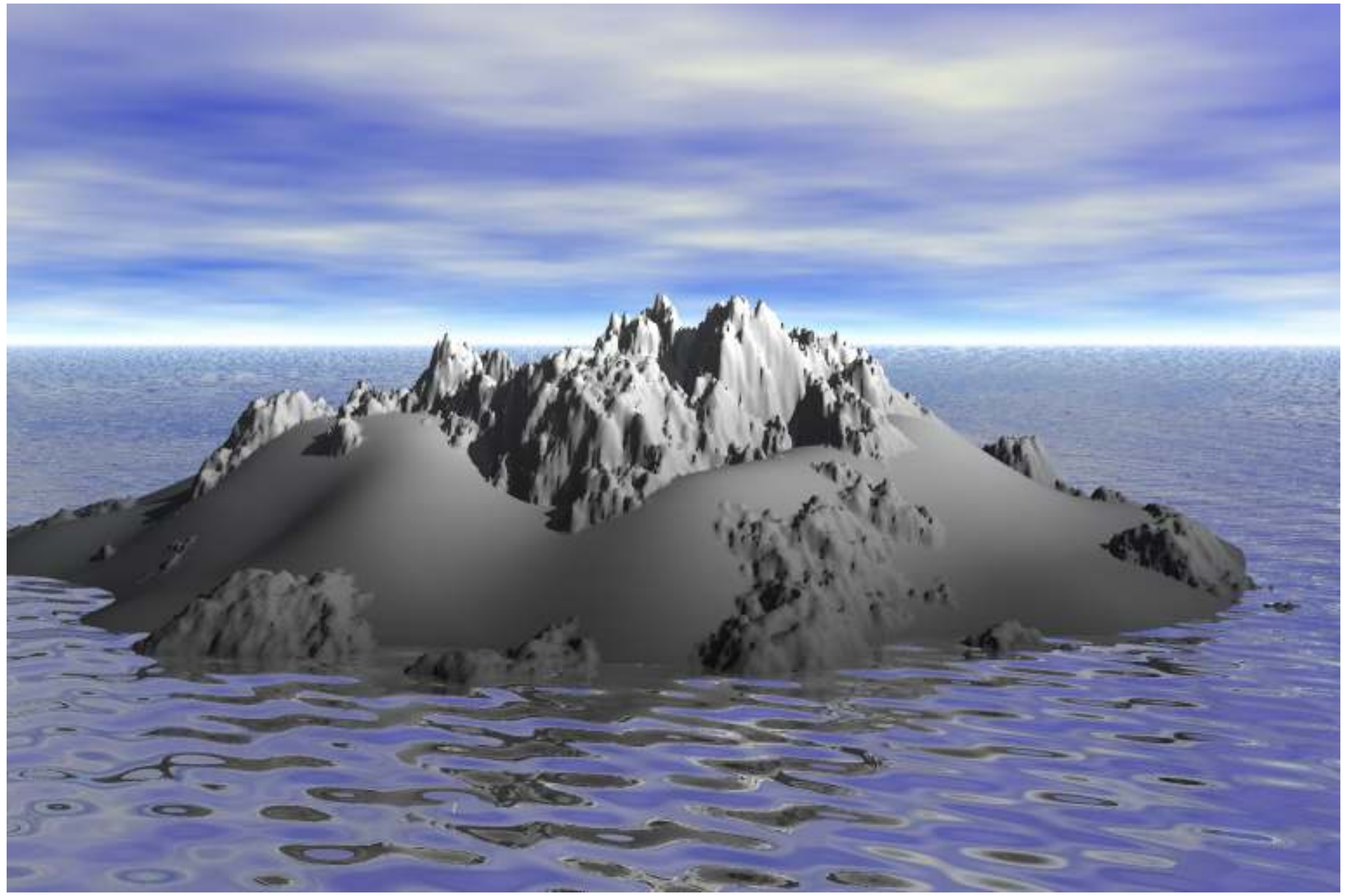
- Plateaus, plains, lakes: clamp height values above maximum, or below minimum





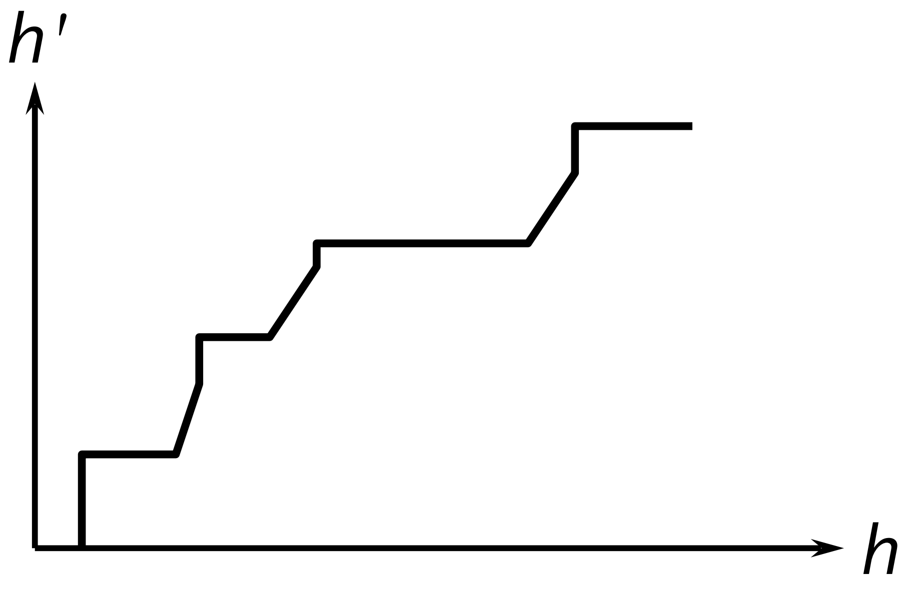


- For more interesting effects:
  - Use low-frequency height map as upper and/or lower envelops
  - Create those low-frequency height map with the same method, but less recursion steps and/or additional smoothing with Gauß kernel

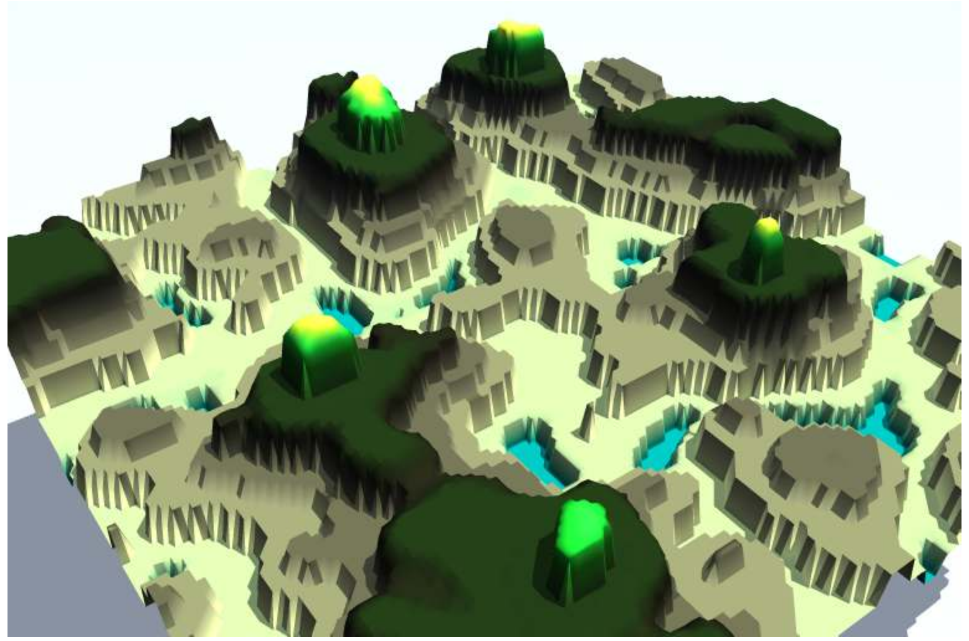


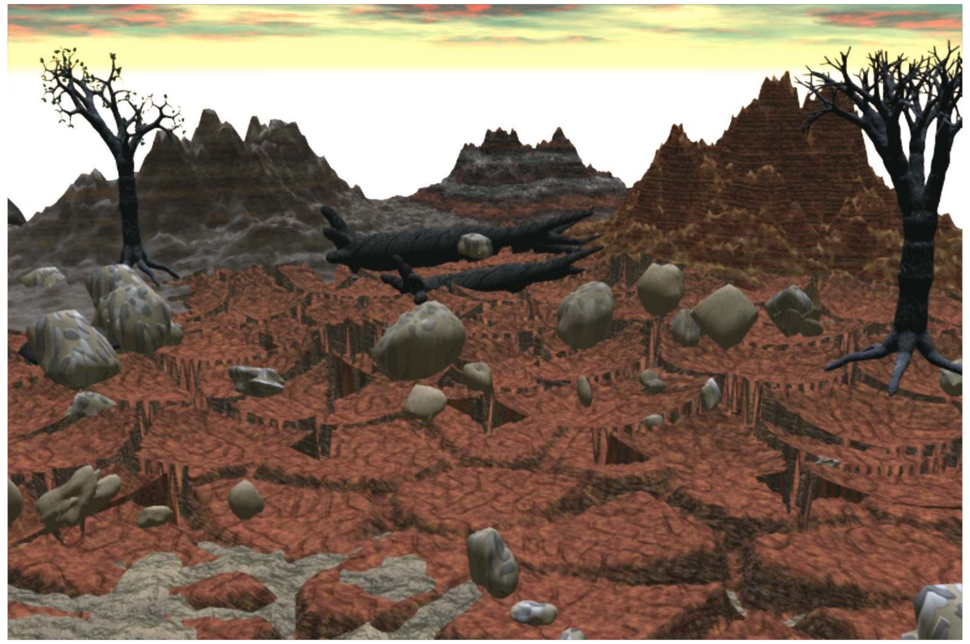
- Ridges: use transfer function that maps

$$h(x, y) \mapsto h'(x, y)$$

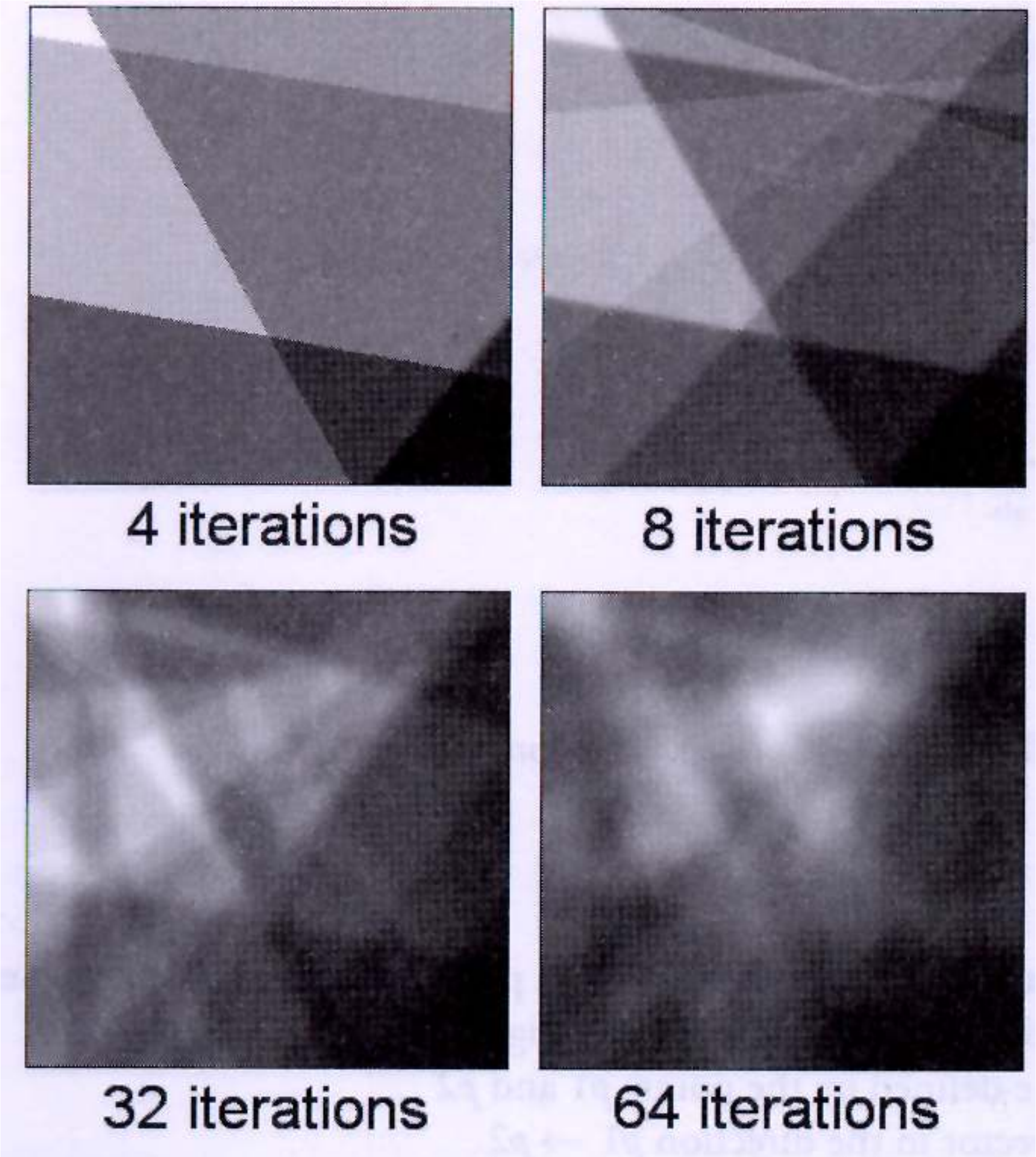


- Cracked terrain:
  - Throw random points onto the plane
  - Construct the Voronoi diagram and use the graph to etch into the terrain

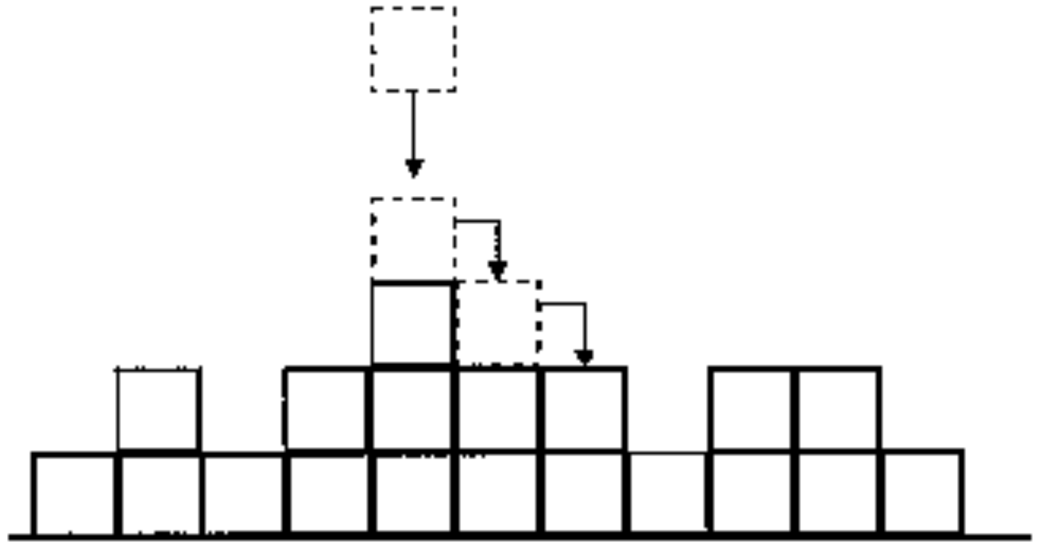




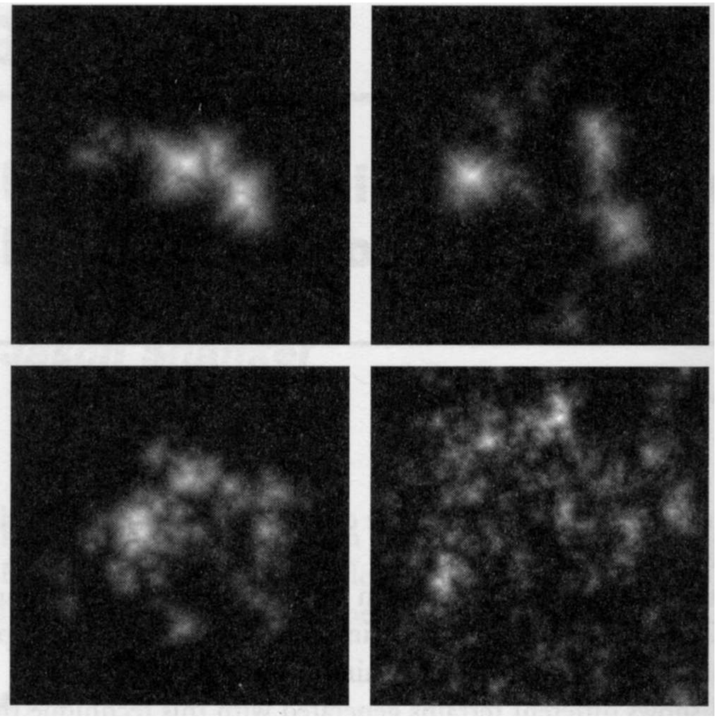
- Add a number of "fault lines":
  - Generate random line
  - Add Δh to all pixels on positive side of fault line
  - Decrease Δh ever k-th step
- Smooth out abrupt fault transitions ("erosion") by convolution with filter kernel (e.g., averaging, or Gaussian)



- Volcanic mountains by particle deposition:
  - Drop particles in areas where you want volcanic mountains
  - Particles flow downhill until they have cause to rest
  - Move drop point to create several peaks



- Particle flow computation:
  - Around given point, consider 3x3 or 5x5 window
  - If local geometry is not "flat enough", move new particle to top of lowest neighbor, and repeat
  - Geometric predicate for "flatness"= check height variance or slope of regression plane
  - Add randomness in various places of algo



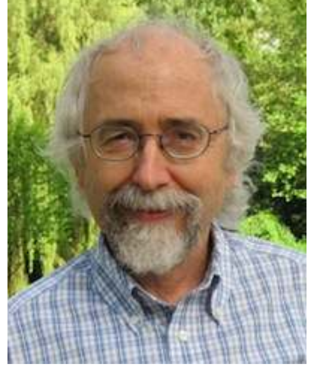
Examples

Image created by Hannes Janetzko, using Terragen2

L-Systems

Jurassic World, 2015

Definition

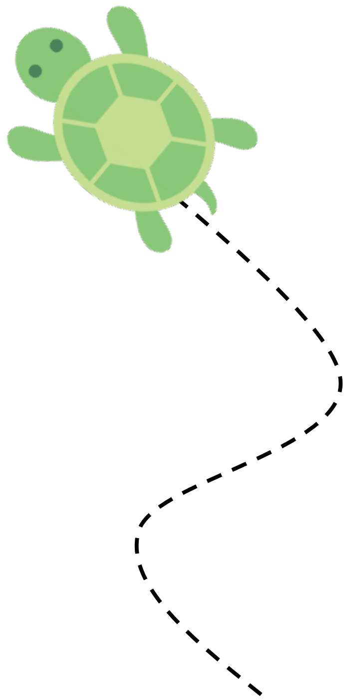


Przemysław Prusinkiewicz

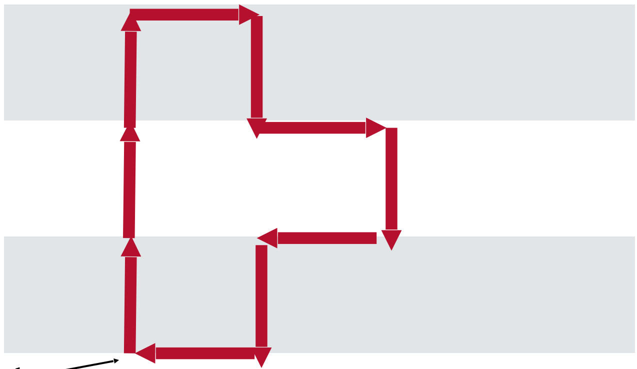
Aristid Lindenmayer

- L-system = term rewriting system
- Simplest variant:
  - D0L-system = deterministic, context-free grammar $G = (V, w, P)$
  - V = alphabet
  - $w \in V^+$ = special start/initiator string
  - P = production rules of the form $p_i\!: \; l \rightarrow r$
    where $l \in V$ and $r \in V^+$
- Note: L-systems have a different "execution model" than grammars!
  - Grammars: apply productions sequentially
  - L-systems: productions are always applied in parallel (in principle)!
    - Intended to capture the simultaneous progress of time in all parts of the growing organism

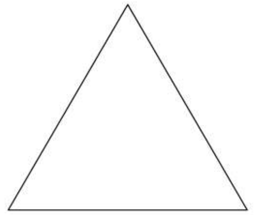
- Giving graphical meaning to strings: turtle graphics
  - Idea: small robot (turtle) with a pen, that reacts to commands
  - Example: w → FFF-F-F+F-F-F+F-F

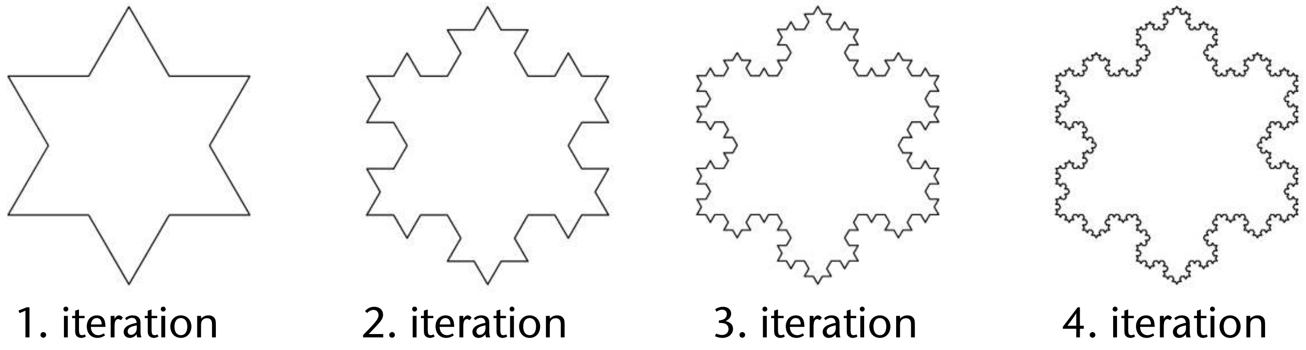


F	Go *d* units forward
+	Turn left about δ degrees
-	Turn right about δ degrees

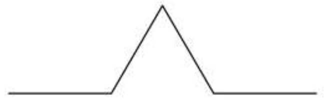


- Further example:
  - Alphabet: $V = \{ F, +, - \}$
  - Initiator: $w =$ F - - F - - F
  - Production rule: $P = \{ F \longrightarrow F + F - - F + F \}$
  - Parameter: $\delta = 60^o$


Initiator



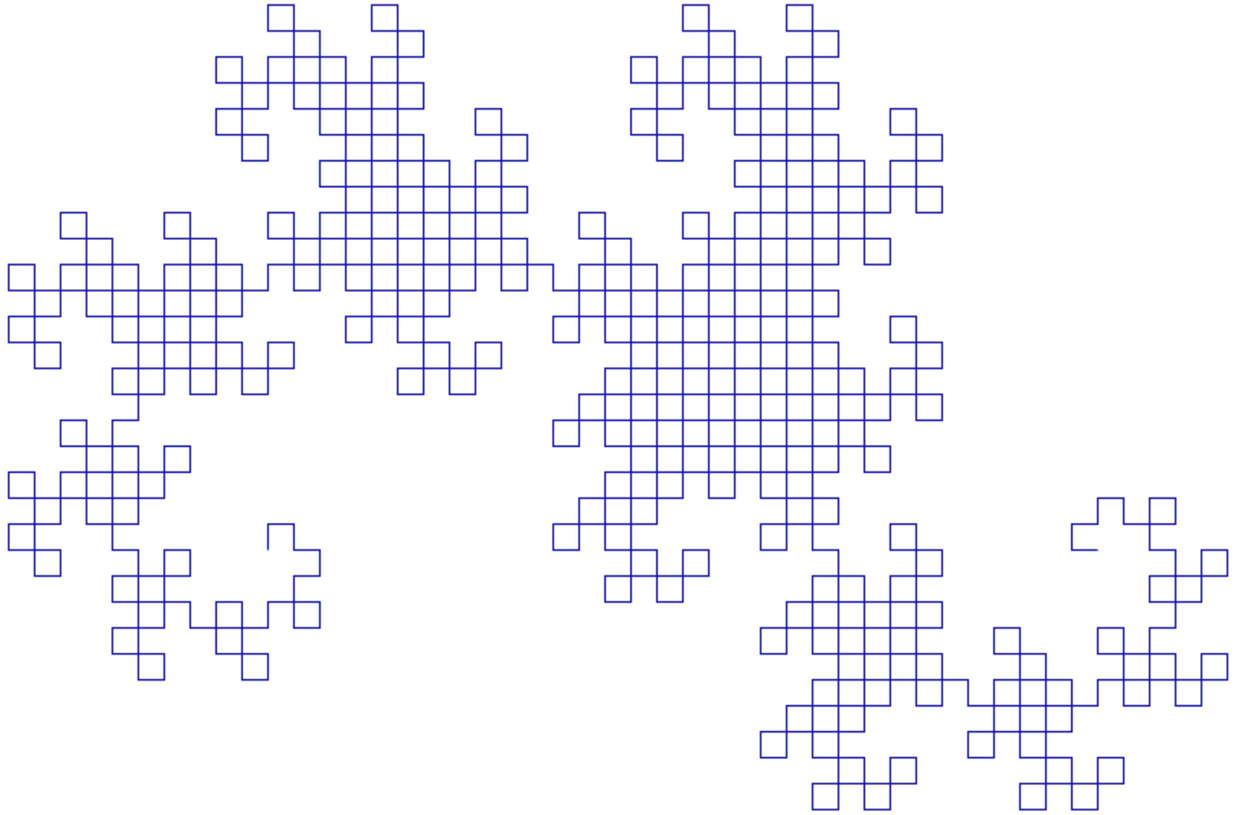
Koch snowflake

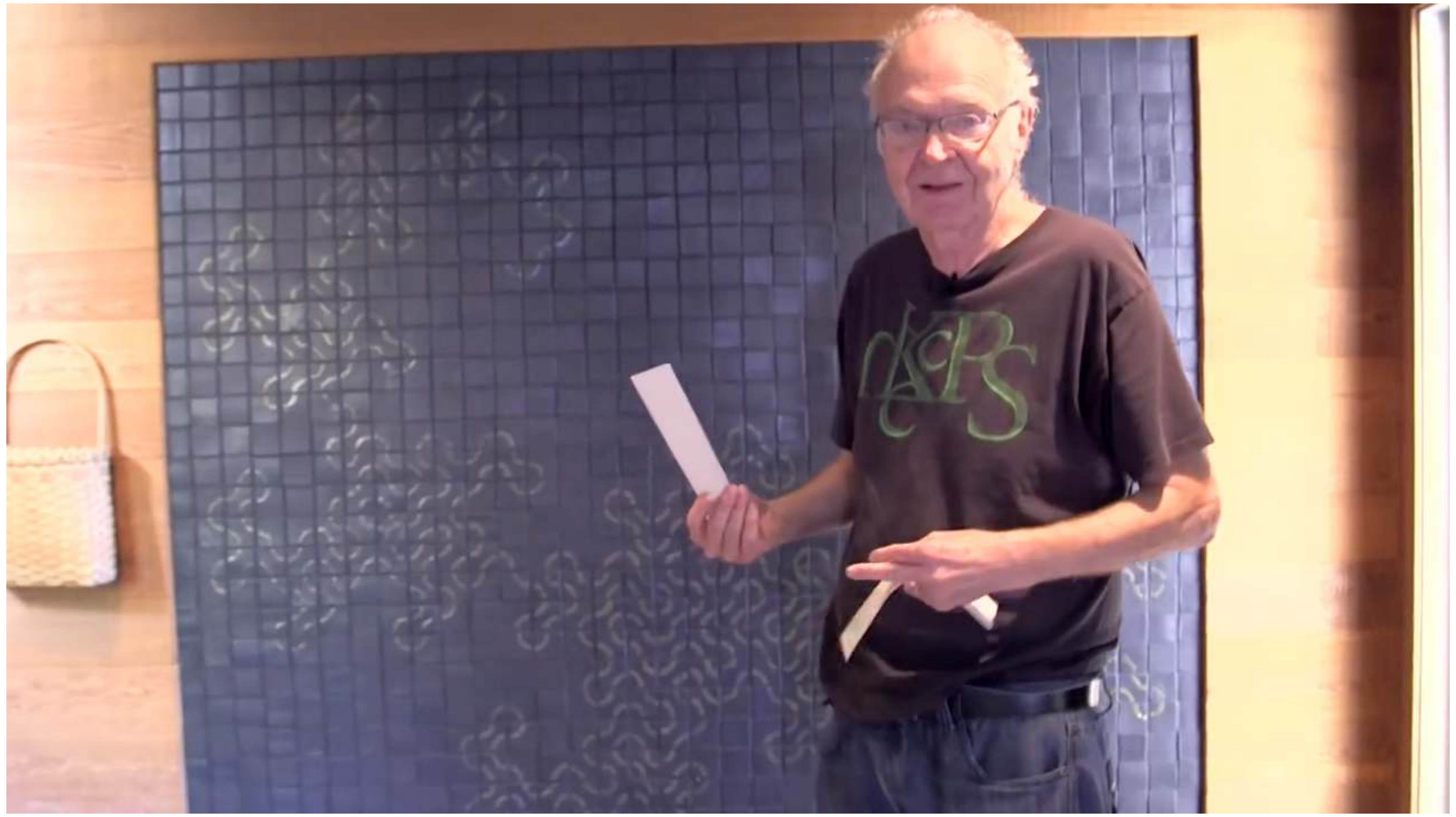

Production rule
(aka generator)

- Dragon curve:
  - Rules:

    X ⟶ X + Y F +

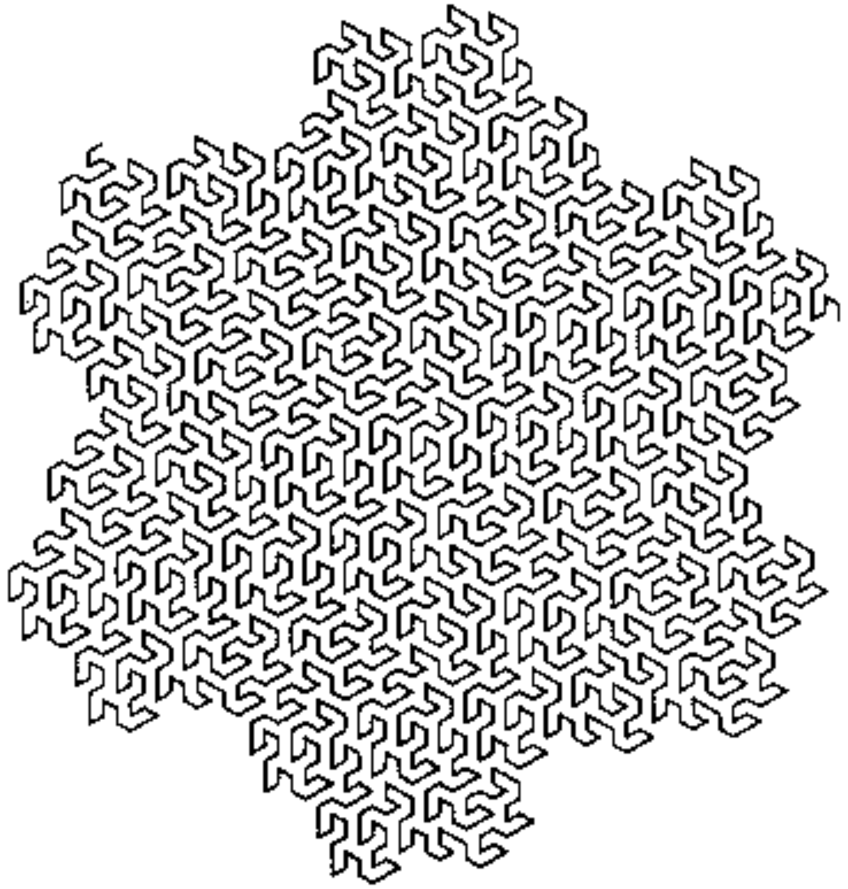
    Y ⟶ - F X - Y
  - Geometric interpretation:
    - F = move forward
    - +/- = turn left/right by 90°
    - X, Y = no geometric interpretation, just for controlling the evolution of the curve





Donald Knuth in his house

- Using L-systems, you can create all kinds of space-filling curves:



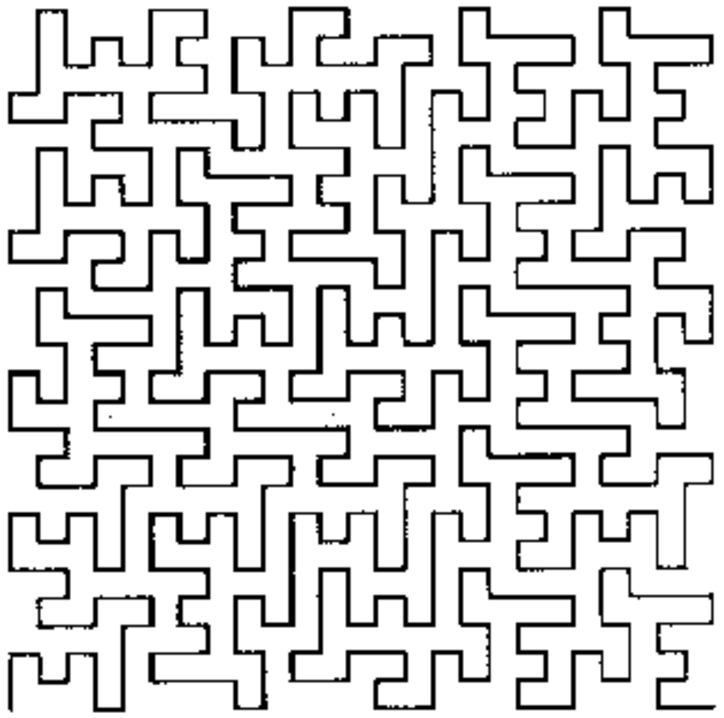
a

$n=4,\ \delta=60°$

F_l

$F_l \rightarrow F_l+F_r++F_r-F_l--F_lF_l-F_r+$

$F_r \rightarrow -F_l+F_rF_r++F_r+F_l--F_l-F_r$



b

$n=2,\ \delta=90°$

$-F_r$

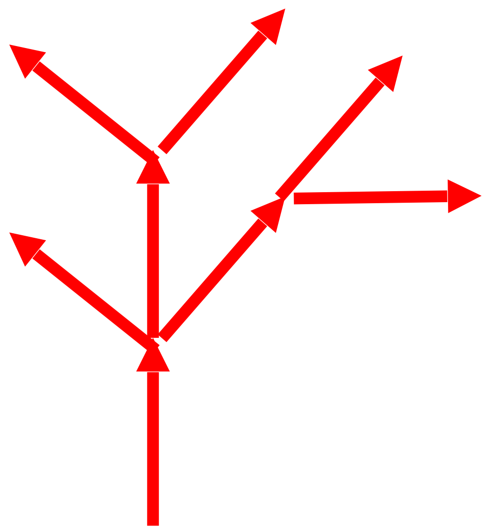
$F_l \rightarrow F_lF_l-F_r-F_r+F_l+F_l-F_r-F_rF_l+$
$\quad F_r+F_lF_lF_r-F_l+F_r+F_lF_l+$
$\quad F_r-F_lF_r-F_r-F_l+F_l+F_rF_r-$

$F_r \rightarrow +F_lF_l-F_r-F_r+F_l+F_lF_r+F_l-$
$\quad F_rF_r-F_l-F_r+F_lF_rF_r-F_l-$
$\quad F_rF_l+F_l+F_r-F_r-F_l+F_l+F_rF_r$

Digression: Real Turtle Graphics

Bracketed L-Systems

- Goal: represent branching structures
- Extension of L-systems: introduce special symbols to maintain a *stack*
  - [= push turtle's current position & heading on *stack* (like OpenGL's pushmatrix()), plus other state info (e.g., segment width) and decrease stepsize *d*
  -] = pop position & heading from stack, set turtle, increase *d*
- Example:



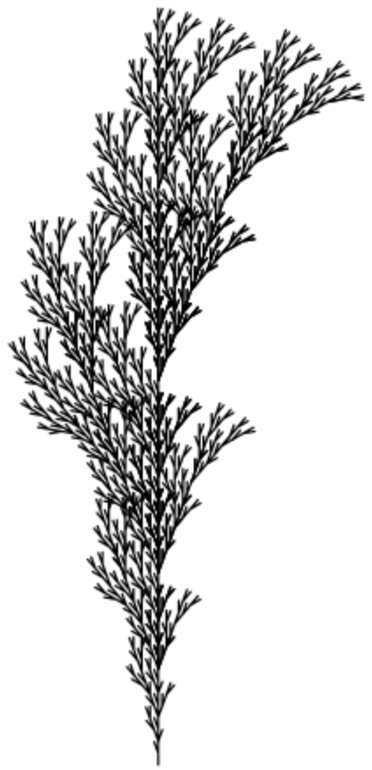
$\delta = 45°$

F[+F][-F[-F]F]F[+F][-F]

- More complex examples:

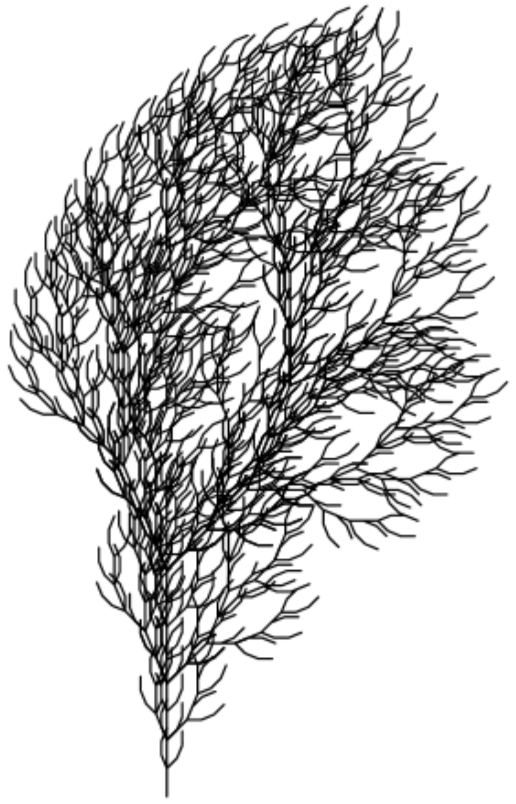
F → F[+F]F[-F]F

$n=5$, $\delta=25.7^{o}$



F → F[+F]F[-F][F]

$n=5$, $\delta=20^{o}$



F → FF-[-F+F+F]+[+F-F-F]

$n=4$, $\delta=22.5^{o}$

X → F-[[X]+X]+F[+FX]-X

F → FF

$\delta=25^{o}$

Bremen

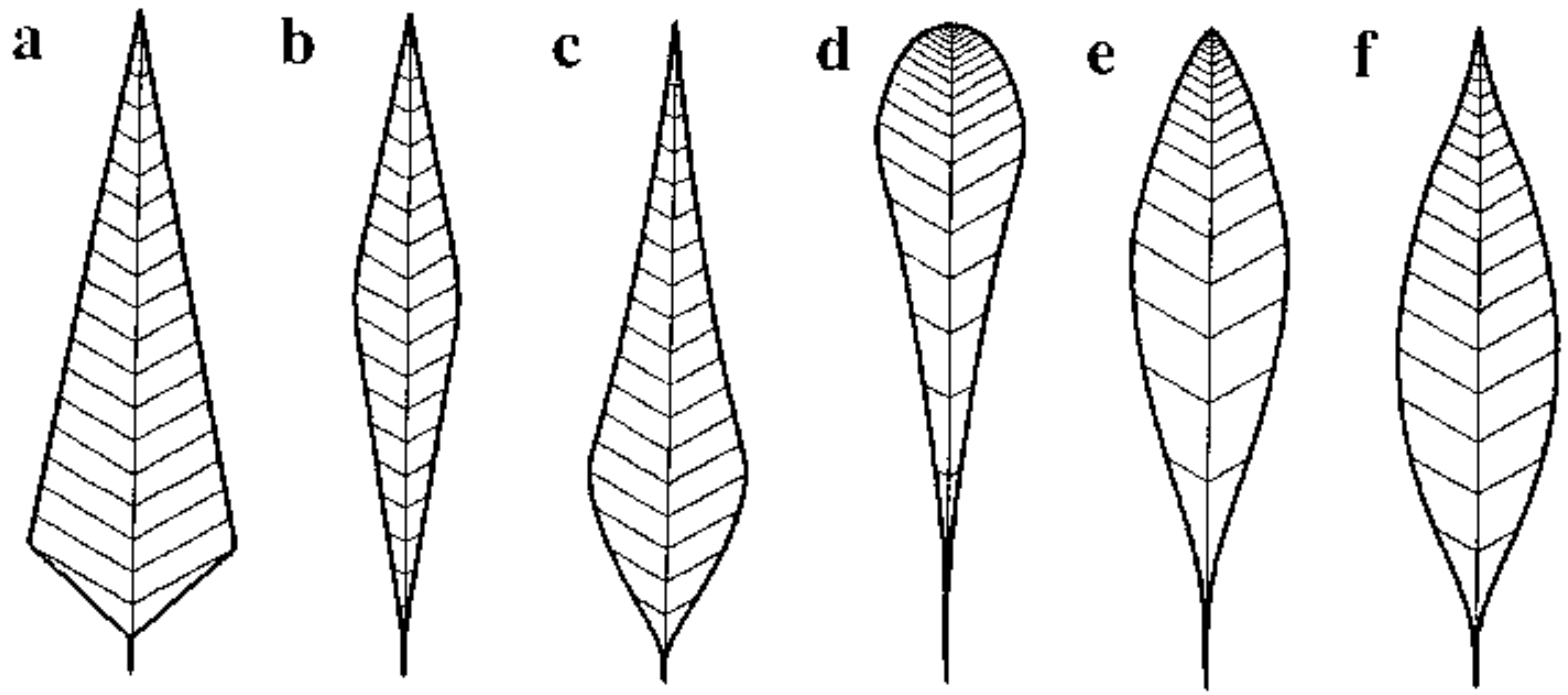
Parametric L-systems

- Idea: since symbols in the string represent geometric/biological features (e.g., a segment of a tree's branch), attribute them with continuous parameters (e.g., segment length)
- Formal definition:
  - Symbols have parameters: $a(x_1, .., x_{ki})$, $a \in V$
  - Parametric production rules have the form p_i: l : *condition over* $(x_1, .., x_{ki}) \rightarrow r$ where $l \in V$ and $r \in V^+$, where l and r are attributed with parameters
- Example: $p_1 \; : \quad \underbrace{A(x,y)}_{\text{Formal parameters}} \; : \; \underbrace{y \leq 3}_{\text{Condition}} \; \rightarrow \; \underbrace{A(2x, x+y)}_{\text{"Output" values}}$
- Example usage: parameter represents length of each segment

$$p_1 : \quad A(s) \; : \; s \leq 3 \; \rightarrow \; A(s+1) \quad \text{segment grows}$$

$$p_2 : \quad A(s) \; : \; s = 4 \; \rightarrow \; A(2)XA(2) \quad \text{segment splits}$$

With parametric L-systems, it is easy to create varying structures



$n=20$, $\delta=60°$

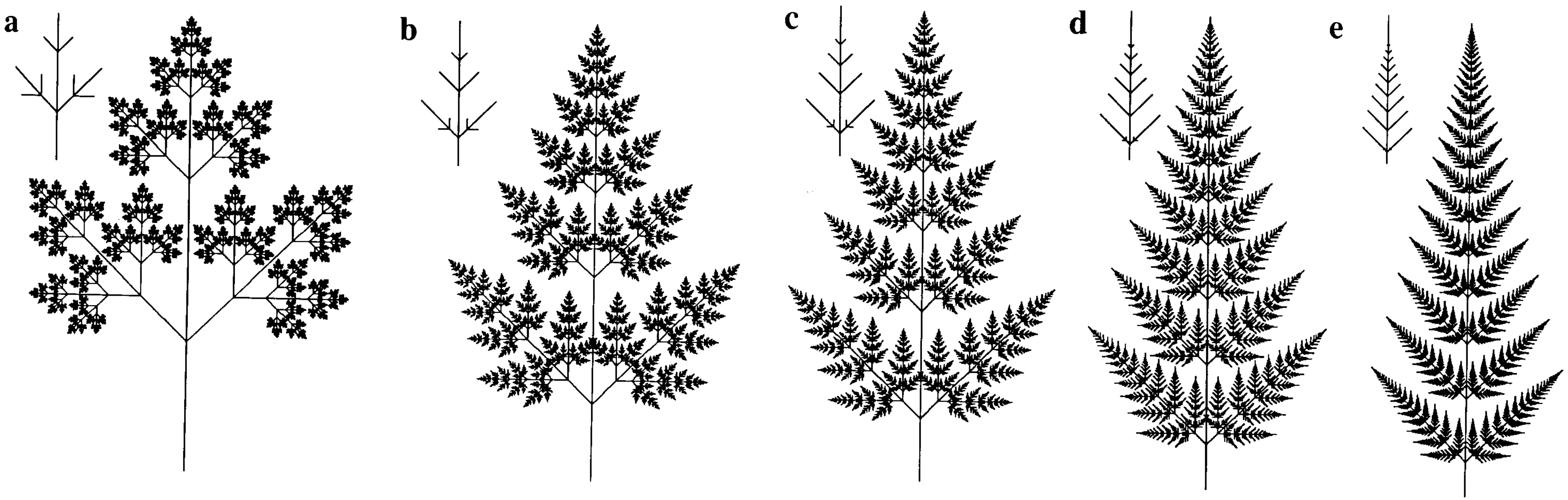
```
#define LA 5    /* initial length - main segment */
#define RA 1    /* growth rate - main segment */
#define LB 1    /* initial length - lateral segment */
#define RB 1    /* growth rate - lateral segment */
#define PD 1    /* growth potential decrement */
```

```
ω  : {.A(0)}
p1 : A(t)    : *    → G(LA,RA)[-B(t).][A(t+1)][+B(t).]
p2 : B(t)    : t>0  → G(LB,RB)B(t-PD)
p3 : G(s,r)  : *    → G(s*r,r)
```

The leaves were generated with a parametric L-system
(only slightly more complex)

$$
\begin{array}{lllll}
\omega & : & \mathtt{A(0)} & & \\
p_1 & : & \mathtt{A(d)} & : \ \mathtt{d > 0} & \rightarrow \mathtt{A(d-1)} \\
p_2 & : & \mathtt{A(d)} & : \ \mathtt{d = 0} & \rightarrow \mathtt{F(1)[+A(D)][-A(D)]F(1)A(0)} \\
p_3 & : & \mathtt{F(a)} & : \ * & \rightarrow \mathtt{F(a*R)}
\end{array}
$$

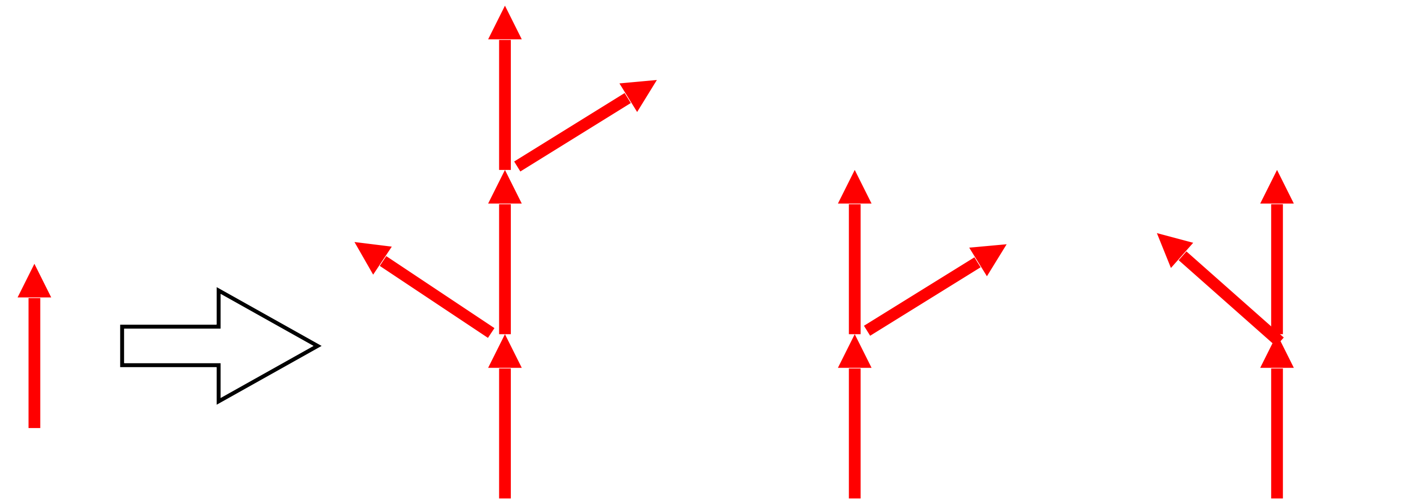
Figure	D	R	Derivation length
a	0	2.00	10
b	1	1.50	16
c	2	1.36	21
d	4	1.23	30
e	7	1.17	45



Stochastic L-Systems

- Goal: individual look for all plants generated by same L-system
- Approach: introduce non-determinism
- Definition: $G = (V, w, P, \pi)$
  - π = probability distribution
  - Production rules : $l \ : \ \text{cond} \xrightarrow{p_i} r$, where p_i = probability that the rule is applied
  - As always: $\sum_{\substack{\text{rules with} \\ \text{same } l}} p_i = 1$
- Example:

$$\omega: \quad F$$
$$p_1: \quad F \xrightarrow{.33} F[+F]F[-F]F$$
$$p_2: \quad F \xrightarrow{.33} F[+F]F$$
$$p_3: \quad F \xrightarrow{.34} F[-F]F$$



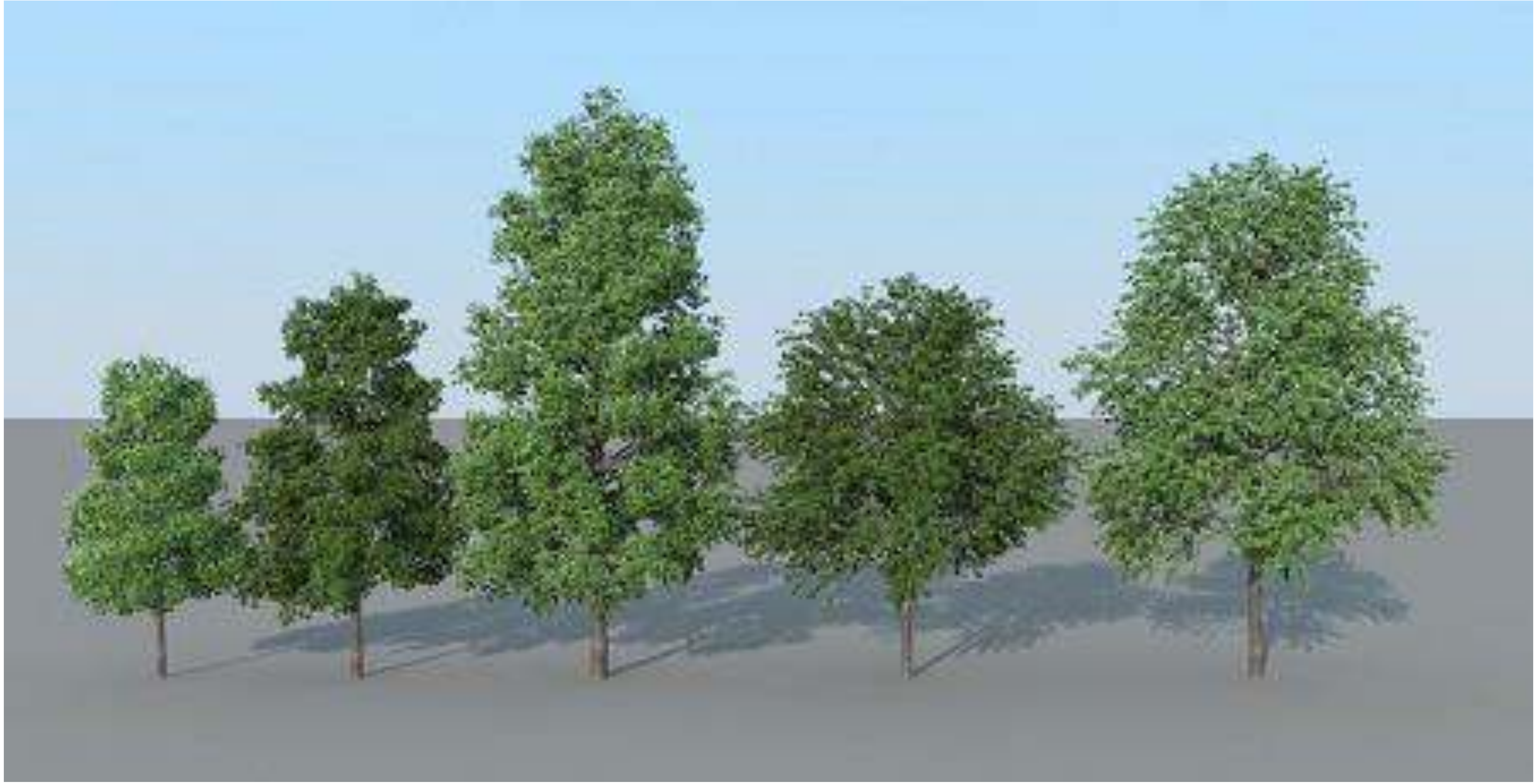
Example: Different plants grown with the L-system of the previous slide

Example of Another Stochastic L-System (all Blue/Red/Yellow Plants have been Grown with the Same L-System)

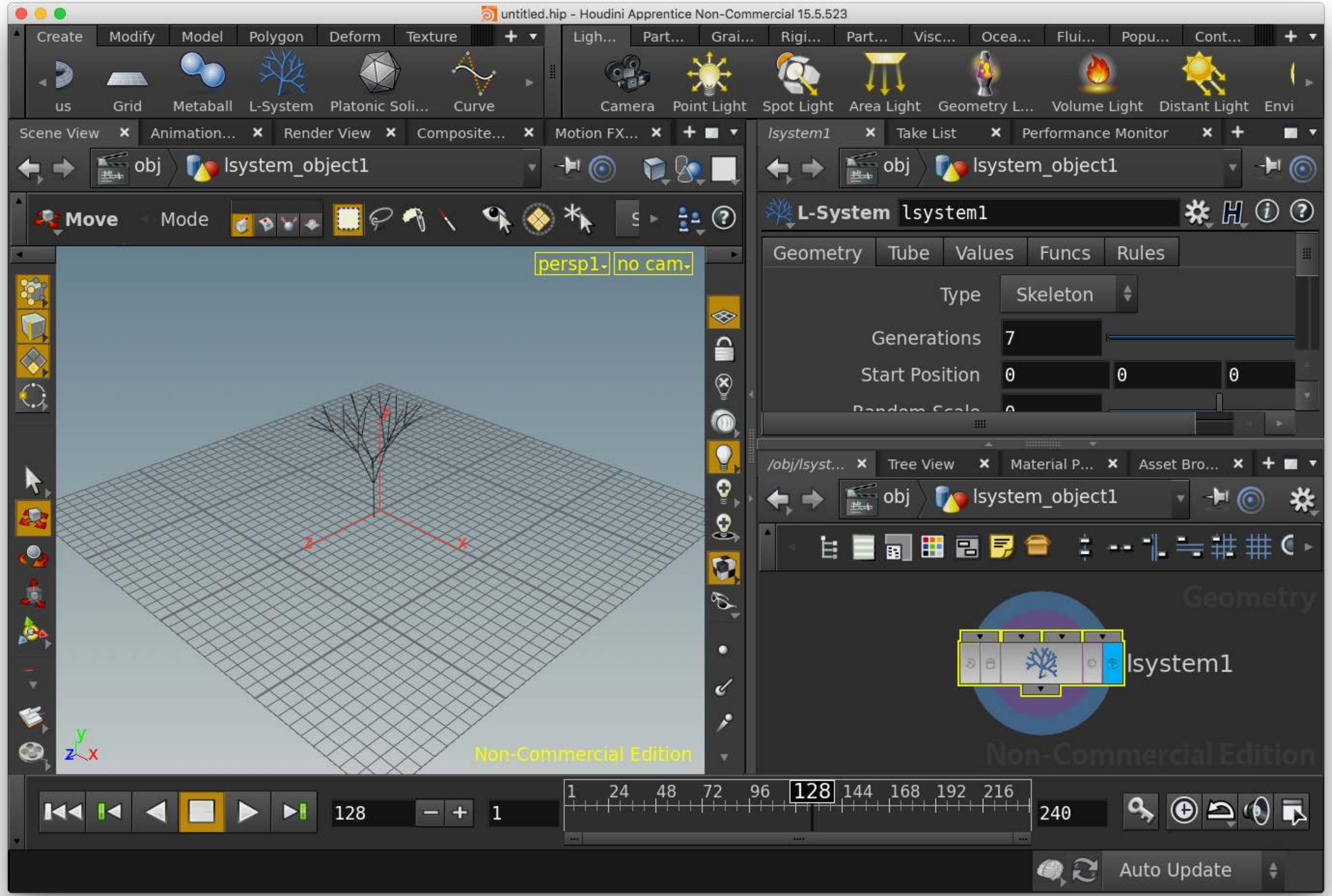
Further Extensions of L-Systems

- Context-sensitive L-systems: productions can be applied only, if neighboring symbols in input string match left-hand side of the rule
- Environmentally-sensitive L-systems: productions can be applied only, if a geometric condition on the left-hand or right-hand side holds
  - E.g., a twig can grow only (F $\longrightarrow$ FF), if its (new) geometry remains within a box
  - Or, a twig can branch off a leaf only, if there is enough light at this point in space

More Examples



Demo: Creation of L-Systems Using Houdini



A Bit of Houdini How-To's

Not Relevant for Exam!

- Step Size Scale:
  - Scaling of branch (`F`) sizes (= step size) with each generation
  - Gets applied when `"` is seen in the RHS of the rule
    - Example: `F -> F"[+F]F[-F]F`
- Angle: angle between stem and branch at **+** and **–** operations
- Parameterization of operators on the right-hand side:
  - Put parameter in `()` after the operator; overrides global parameter (e.g. Angle)
  - Example: `F -> F"(0.95)[+(22)F]F[-(30)F]F`
- Gravity tropism (gravitropism) achieved by operator `T`
  - Examples:
    - `F -> TF[+F]F[-F]F` (straight stem)
    - `F -> T+(2)F[+F]F[-F]F` (bent stem)
  - Set global variable Gravity ≠ 0

- Animating the plant:
  1. Set global variable Gravity to **`sin($F)`** , or similar (**`$F`** = frame counter)
  2. Append Twist node underneath L-System node,
     set **`Bend Mode = Angle`**, set **`Bend = 20*sin($F)`**
  - For fun set additionally **`Twist = 20*sin($F)`**
- Node rewriting (a.k.a. "appending"):
  - Basically same mechanisms, but set up rules such that non-drawing symbols get rewritten
    - Non-drawing symbols = vertices between segments (**`F`** symbol)
  - Example:
    - Initiator: **`FX`**
    - Rule: **`X -> F[+FX][-FX]`**

- Branches with volume (instead of lines):
  - Set **`Geometry/Type = Tube`**
  - **`Tube/Rows`** and **`Tube/Columns`** = tesselation resolution
  - Param. **`Tube/Thickness Scale`** = multiply current thickness with each branching
  - Gets applied with symbol **`!`** in the rules
  - Examples:
    - **`X -> "!F[+FX][-FX]`**
  - Ditto for symbol **`;`** = **`Values/Angle Scale`**
    - Example: **`X -> ";F[!(0.8)+FX][!(0.95)-FX]`**

Make subsequent segments smaller and smaller (Step Size Scale)

Make angle of turns smaller and smaller

Branch off, create one segment, repeat at the end of that segment

Dito

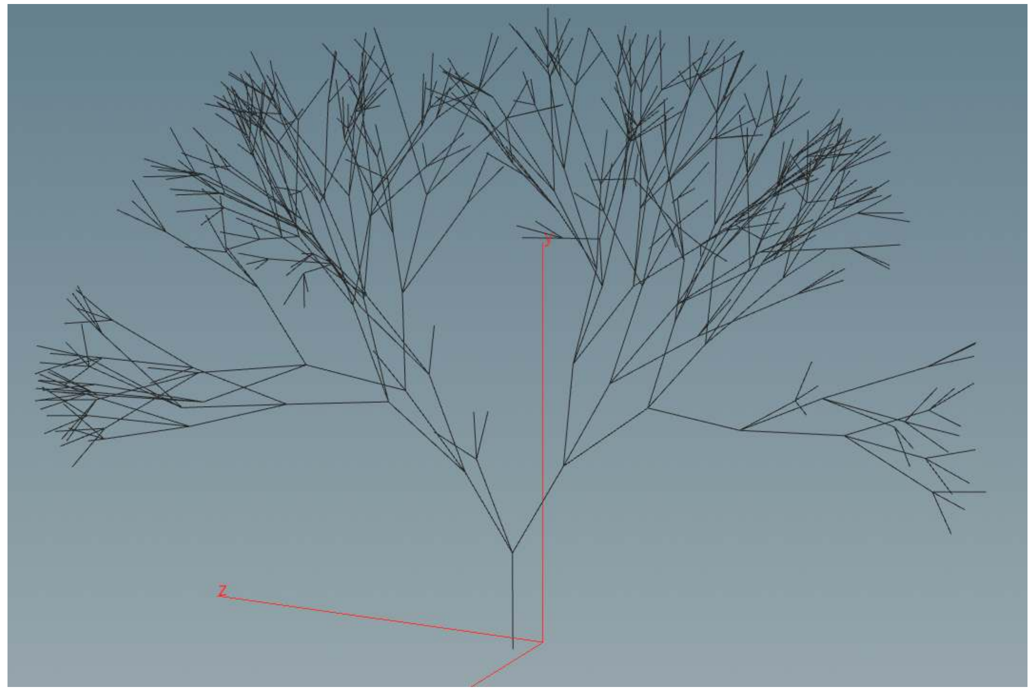
Make tubes smaller by x% of parent branch

- Randomization:
  - Random turn angles: `~(a)`
    makes turtle pitch/roll/turn randomly by up to *a* degrees
  - Example: `X -> ";F[!~(20)FX][~(20)FX]`
  - Example: `X -> ~(20)";F[!+FX][-FX]`
    - Not quite clear why this makes the segments crooked;
      is the random turn angle applied to the small sections of the branch segments?
  - Example:
    `X -> ~(20)";F[!(0.7)~(40)FX][[!(0.8)~(40)FX]!(0.75)~(40)FX]`

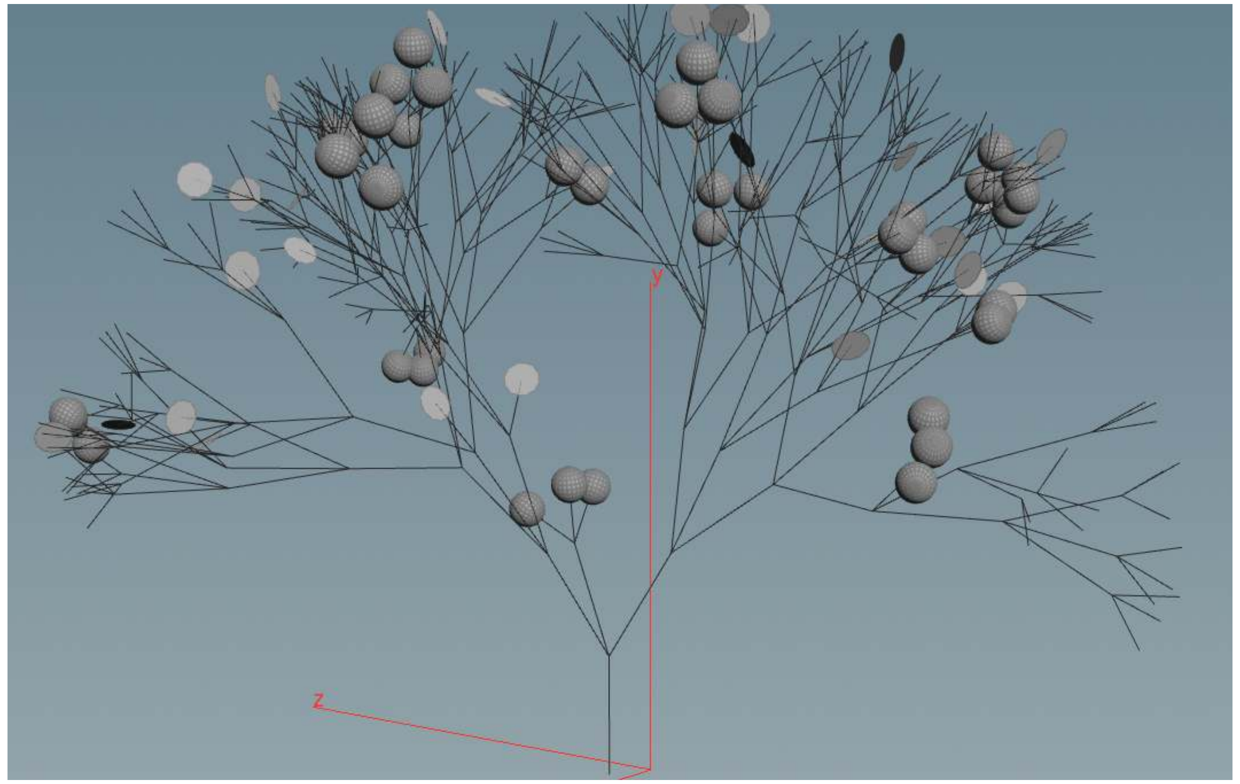
- Stochastic L-systems:
  - Syntax: *symbol* **->** *replacement* **:** *probability*
  - Example:

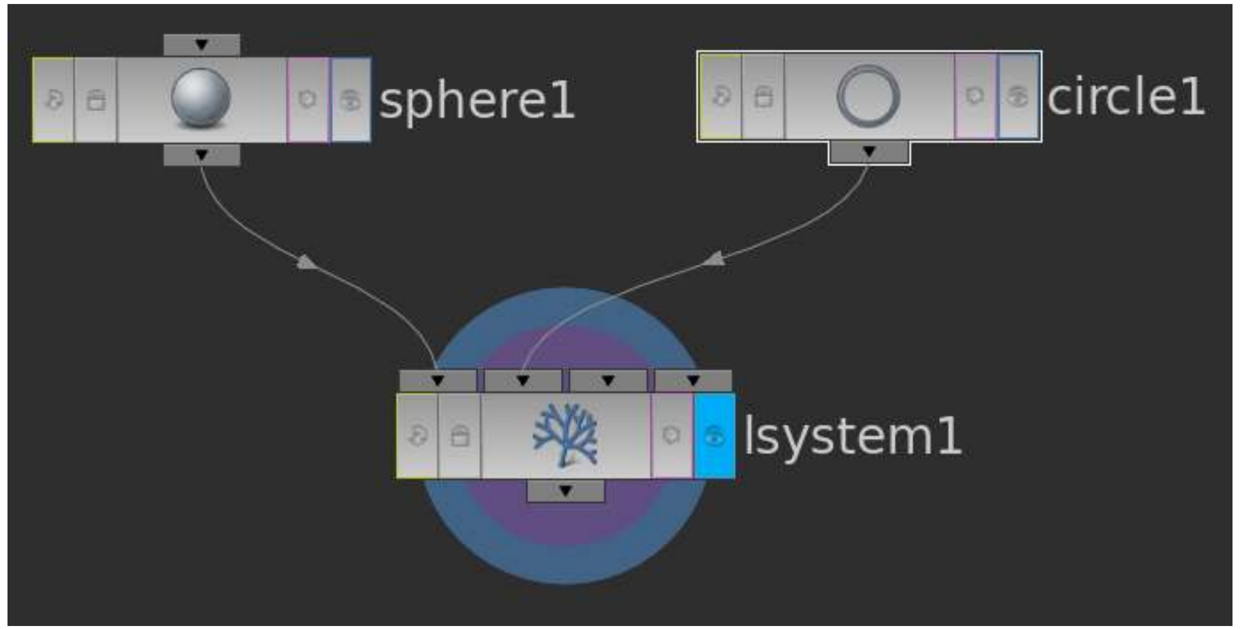
```
Premise:   FX
Rule 1:    X -> F[~(40)FX][~(40)FX][~(40)FX] : 0.7
Rule 2:    X -> F[~(40)FX][~(40)FX][~(40)FJ] : 0.2
Rule 3:    X -> F[~(40)FK][~(40)FK][~(40)FK] : 0.1
```

"K" and "J" nodes:
growth will stop there



- Adding geometry to the tree:
  - "J" and "K" symbols correspond to the "J"/"K" inputs of the L-system node
  - With every symbol J and K in the fully expanded string, one instance of the geometry is attached, when a geometry node is connected to the respective input of the L-system node





- Use conditionals to put leaves on *all* branch endpoints:
  - Syntax: *symbol* : *condition* -> *replacement*
  - Example:

```
FX
X:  t<6   -> F[~(40)FX][~(40)FX][~(40)FX]  :  0.6
X:  t<6   -> F[~(40)FX][~(40)FX][~(40)FJ]  :  0.2
X         -> F[~(40)FK][~(40)FK][~(40)FK]  :  0.2
X:  t>=6  -> F[~(40)FK][~(40)FK][~(40)FK]  :  0.6
```

Can get applied only during the first 6 iterations

Growth stops here anyways

After 6 iterations, only this rule can be applied, and this produces only "K"-nodes

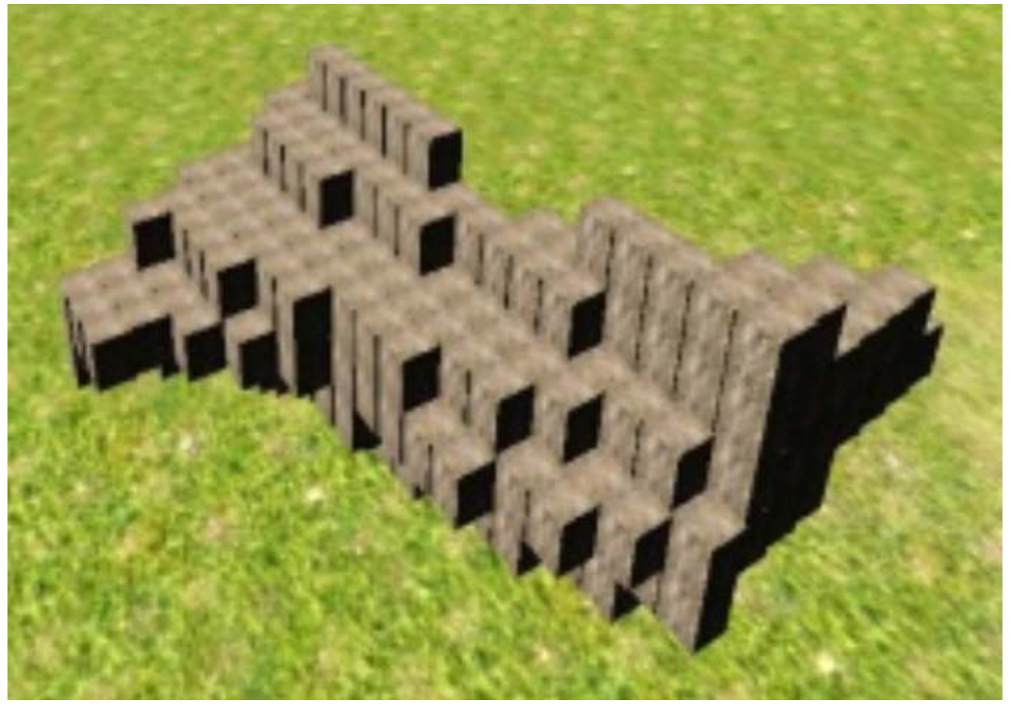
Probability

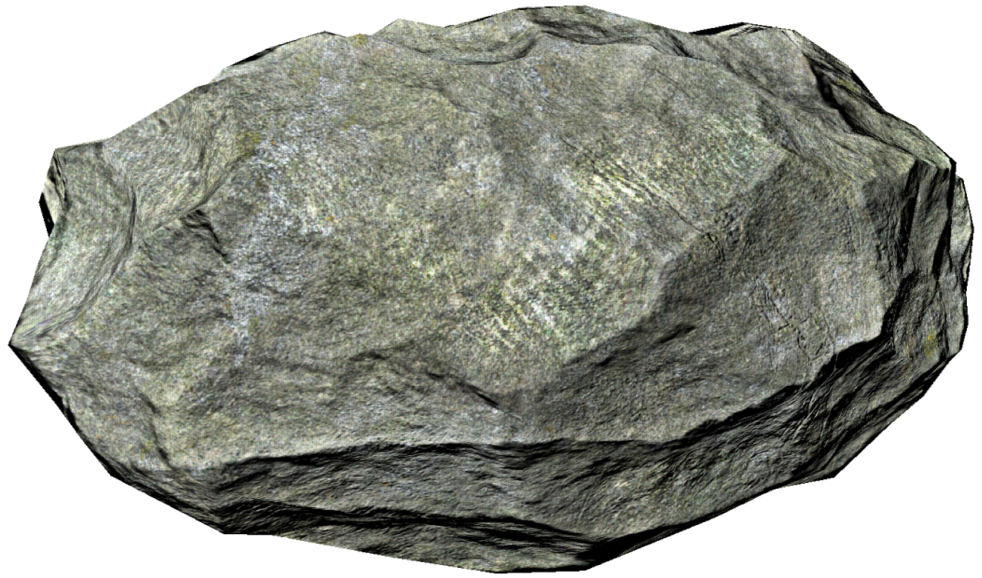
Iteration/generation count (pre-defined variable)

Demo: animated growth in Houdini

Procedural Modeling of Rocks

- Use 3D L-system to generate "Minecraft"-like rocks
  $\longrightarrow$ skeleton of the rock
- Generate randomly a number of rule sets
  - Each represents a "family of rocks"
- Use genetic algorithm to generate and modify the L-systems in a large population of L-systems
  - Optimization goal (= fitness function): overall shape should fit user-specified parameters, e.g., aspect ratio of bbox, "roughness of surface", ...
- Post-processing afterwards on the fittest rock:
  - Compaction = try to eliminate interior voids
  - Erosion = remove outer "spike-like" cubes (protrusions)
- Generate mesh

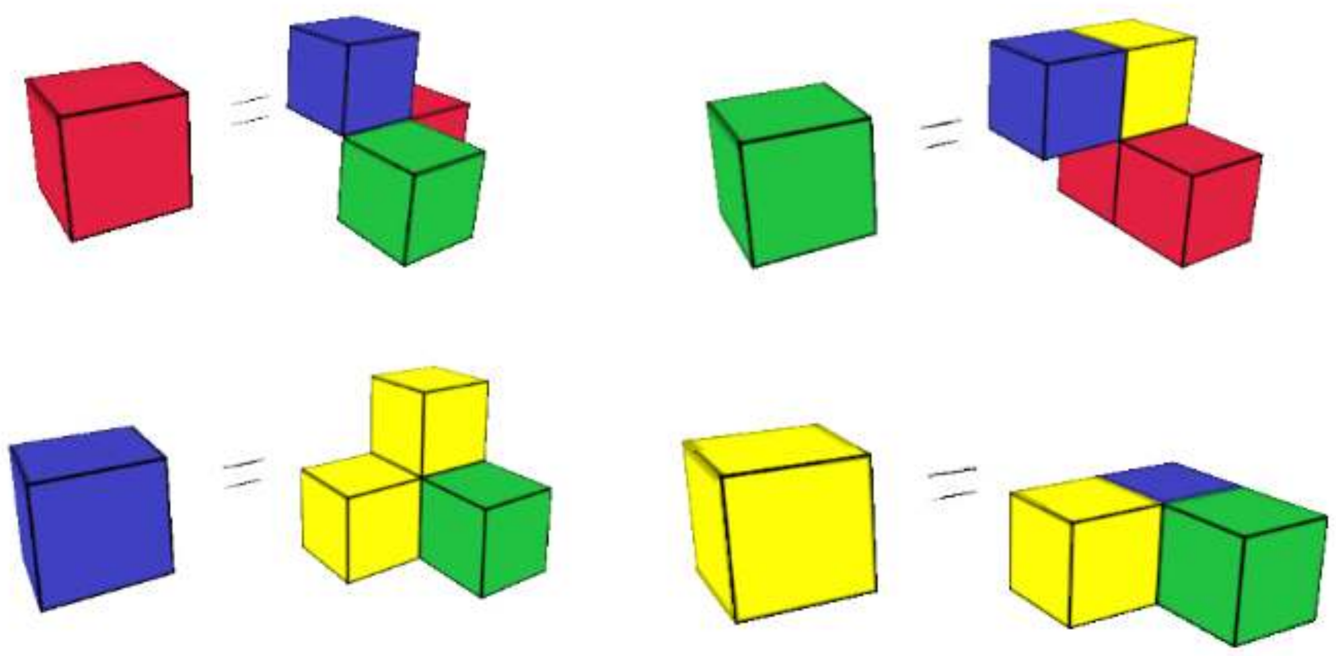


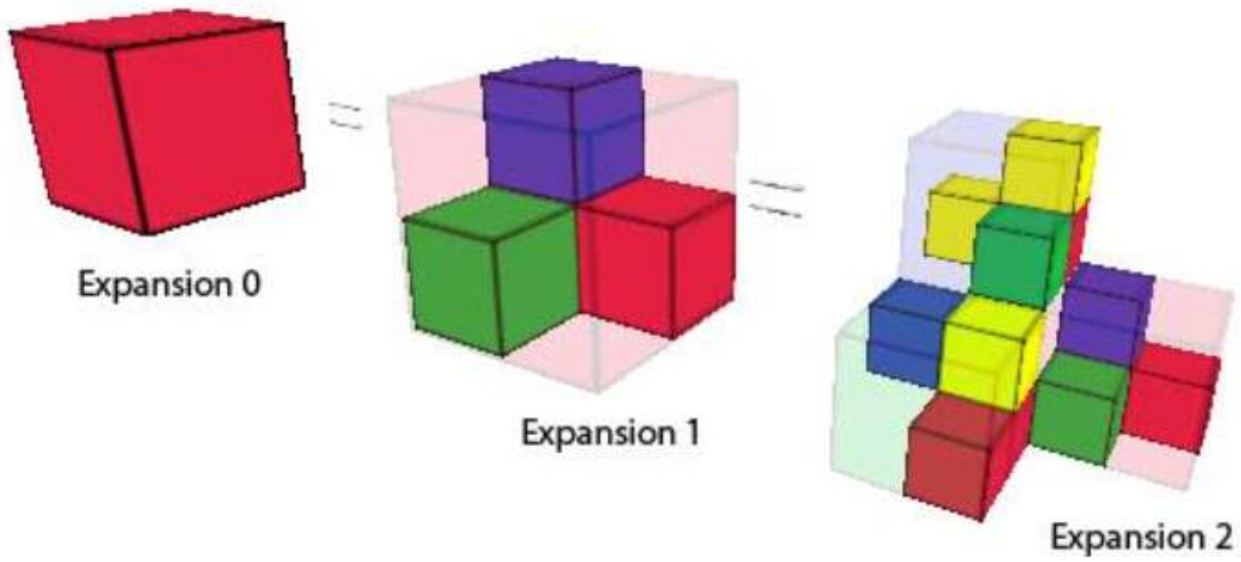


["SpeedRock" – Dart et al., 2011]

The Cube-Based 3D L-system

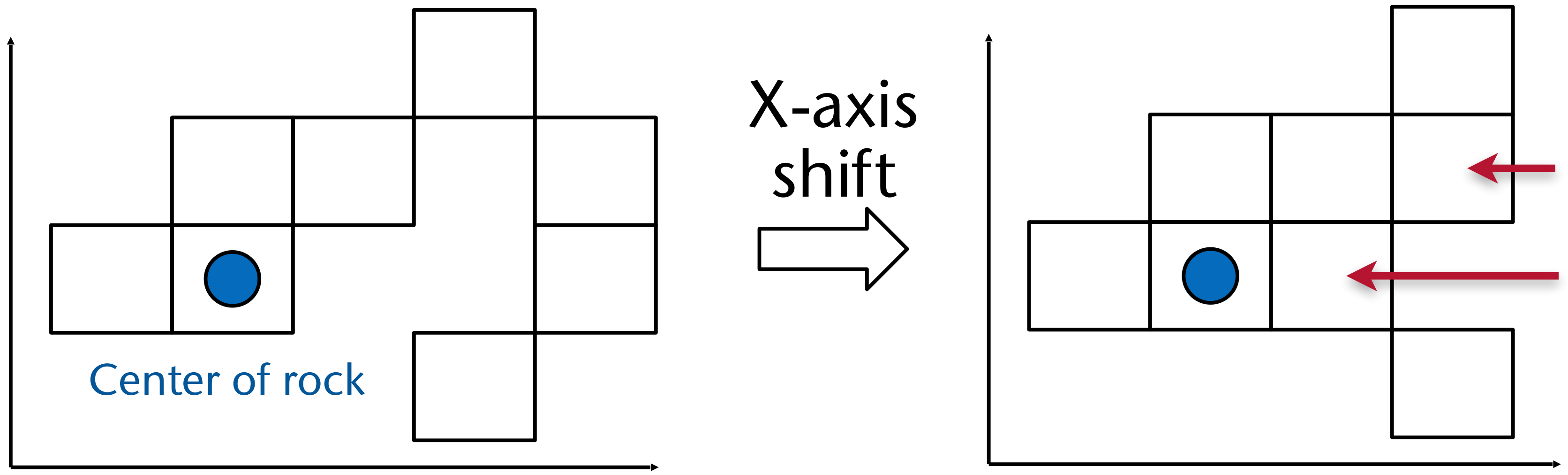
- Work directly on cubes (no turtle)
- Rules have the form / meaning:
  - Partition cube into 2x2x2 sub-cubes
  - Replace big cube by filling some of the sub-cubes
  - Cubes have colors (= IDs)
- Such rule sets can be generated randomly very easily
- Example expansion



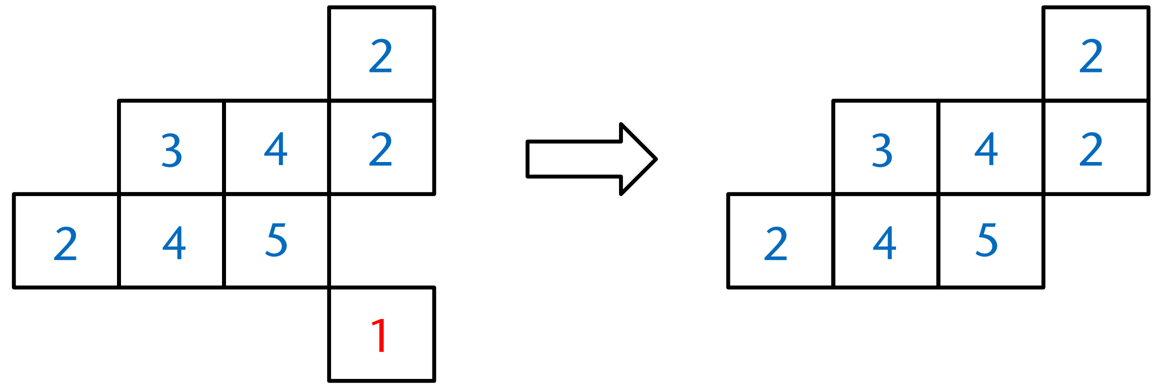


The Genetic Algorithm

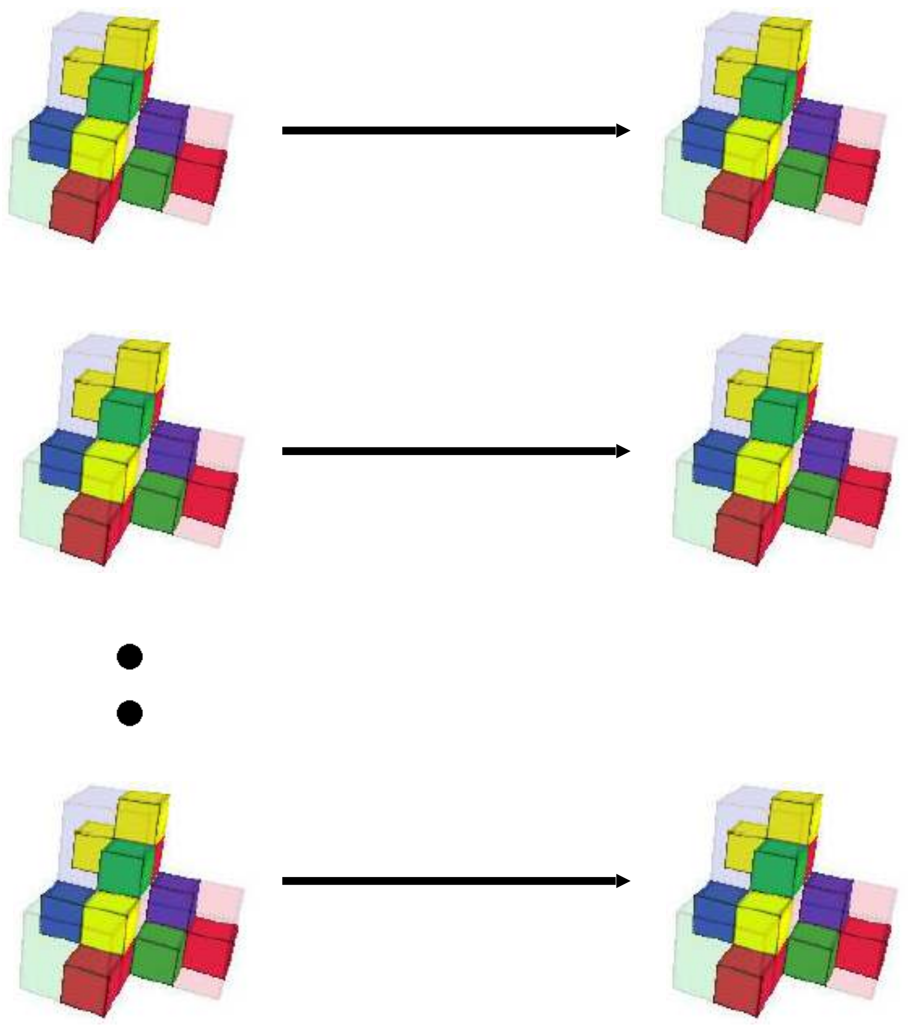
1. Populate pool of L-systems by random generation of rule sets
2. Expand each L-system up to n expansions $\longrightarrow$ "cuboid"-shaped structures
3. Compact these structures
   - Shift cubes along a coord axis towards center, if neighboring cells are free
   - Take turns along each axis



4. Erode prototype structures:
   - For each cube, if number of neighbors < user-specified threshold (e.g., 2), delete the cube



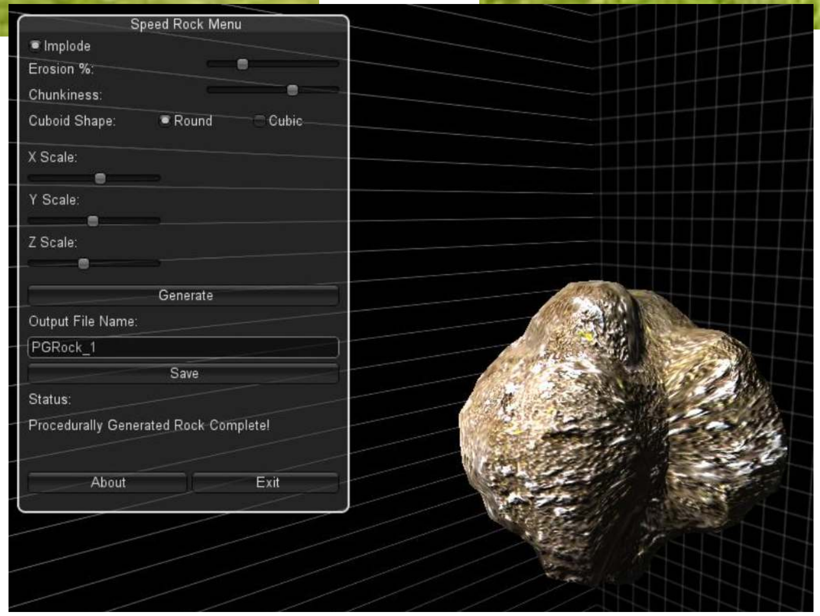
5. Evolve population (= selection and mutation)
   - For all such structures, compute their fitness according to user-specified shape fitness criteria (e.g., aspect ratio)
   - Delete *x*% of unfittest rule sets (= rocks)
   - Mutate some of the remaining rule sets
     - Randomly set new value of a random sub-cube
   - Splice some rule sets to create new L-systems, i.e., mix rule sets



- Fitness criteria (parameters for user):
  - Mass (= number of occupied cubes in final expansion)
  - Aspect ratio of bbox
  - Percentage of cubes deleted in erosion step
  - Total number of sub-cubes that had to be shifted in the compaction step
- Repeat evolution steps, until no improvement in overall fitness of the whole population occurs any more
- Pick *n* fittest rock structures
- Generate mesh for each:
  - Triangulate outer cube sides
  - Add noise to vertex coordinates
  - Do a few iterations of mesh smoothing

Examples

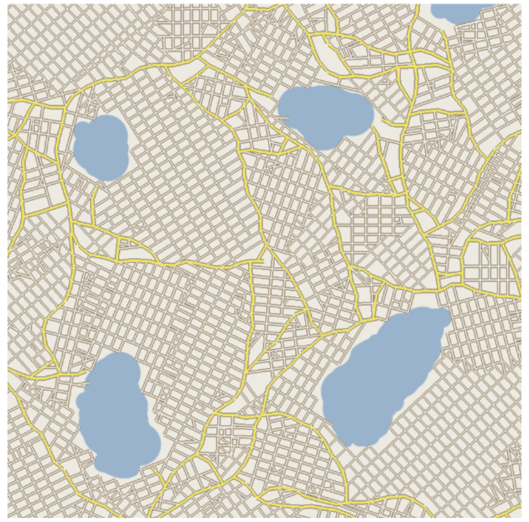


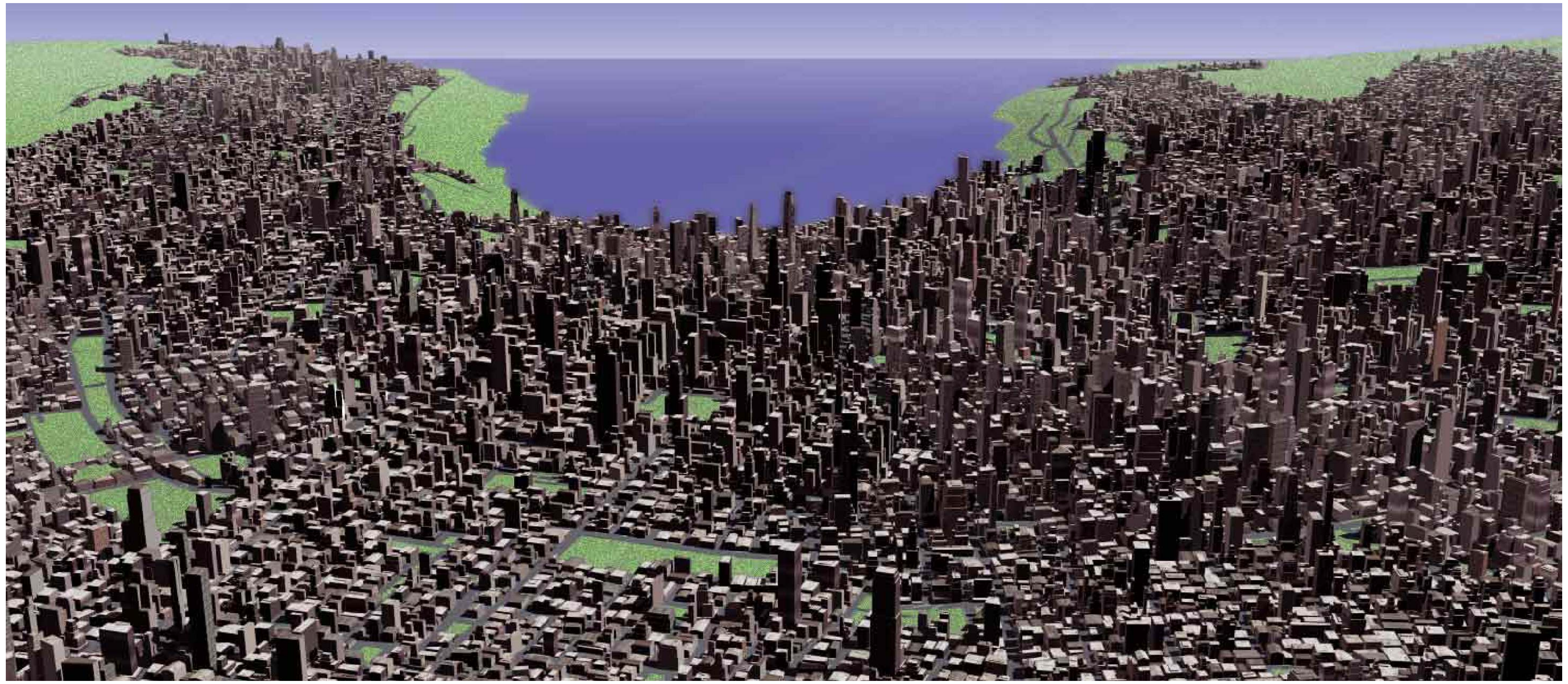
"SpeedRock", Dart et al. 2008 / 2011

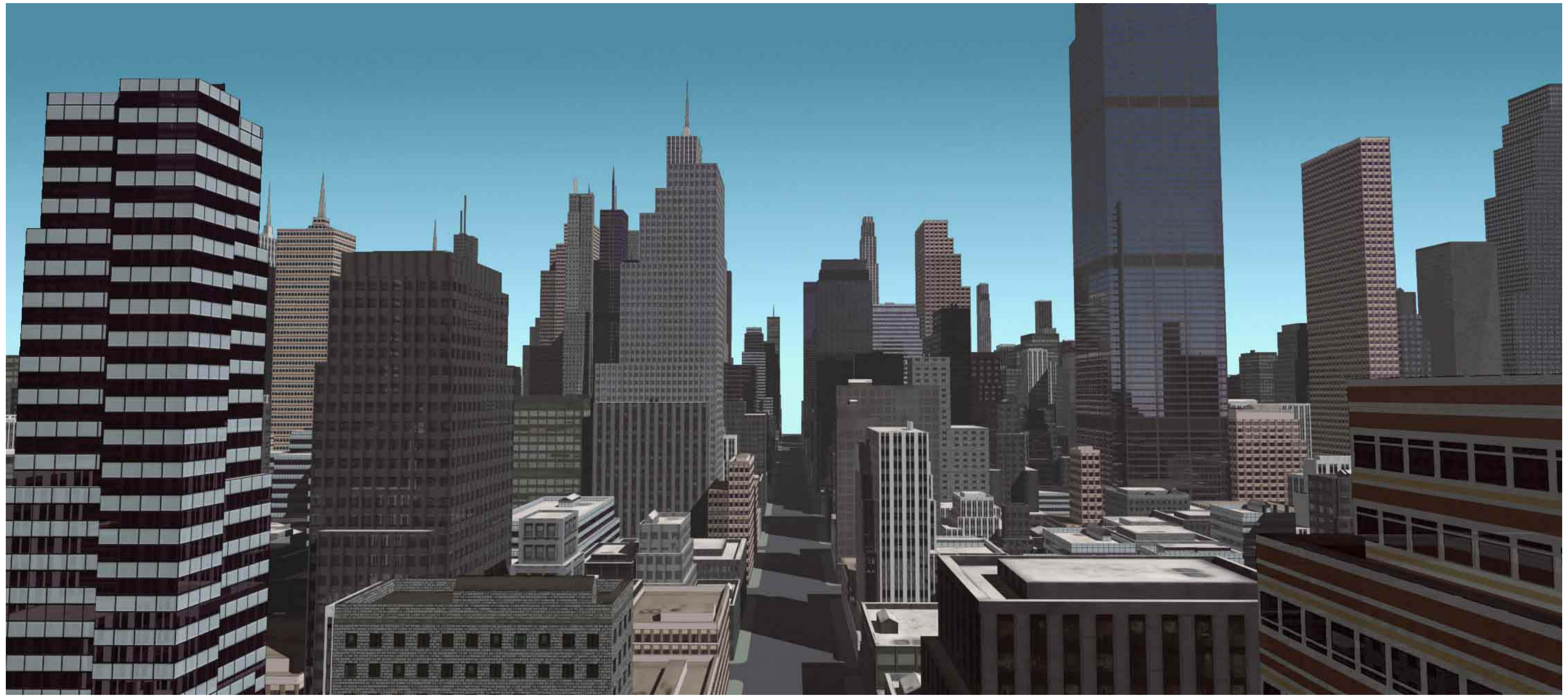
Procedural Modeling of Cities

- Input is geographic map and population density map
- Algorithm divided into four main stages:
  - Roadmap creation (extended L-systems)
  - Division into lots/quarters/neighborhoods (subdivision)
  - Building generation (stochastic, parametric L-systems)
  - Geometry (parser)

Petrasch 2008, TU Dresden





Generative Art Using L-Systems