CSE 167:
Introduction to Computer Graphics
Lecture #15: Procedural Modeling

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2018

Announcements

- Final project is on-line
  - 3 blog post deadlines, first one due next Tuesday
- Discussion topics tomorrow:
  - Final Project
  - Midterm #2
- Midterm #2 next Thursday in class
- Project 4 late grading next Friday at 2pm

Lecture Overview

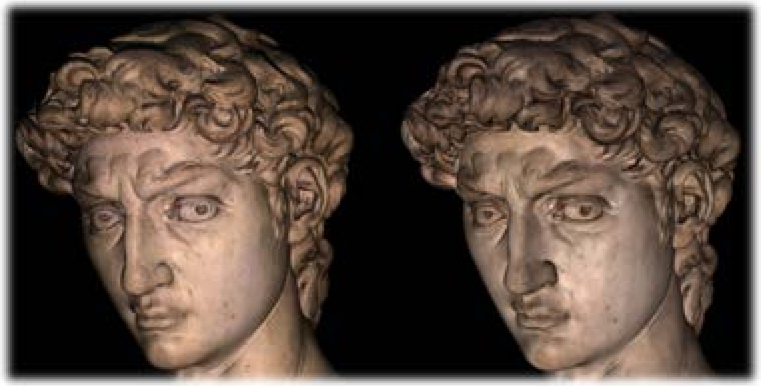
- Procedural Modeling
  - Concepts
  - Algorithms

3D Modeling

- Creating 3D objects/scenes and defining their appearance (texture, etc.)
- So far we created
  - Triangle meshes
  - Bezier patches
- Interactive modeling
  - Place vertices, control points manually
- For realistic scenes, we need extremely complex models containing millions or billions of primitives

Alternatives

- Data-driven modeling
  - Scan model geometry from real world examples
  - Use laser scanners or similar devices
  - Use photographs as textures



Photograph Rendering

[Levoy et al.]

- Procedural modeling
  - Construct 3D models and/or textures algorithmically

Procedural Modeling

- Wide variety of techniques for algorithmic model creation
- Used to create models too complex (or tedious) to build manually
  - Terrain, clouds
  - Plants, ecosystems
  - Buildings, cities

[Deussen et al.]

- Usually defined by a small set of data, or rules, that describes the overall properties of the model
  - Example: tree defined by branching properties and leaf shapes
- Model is constructed by an algorithm
  - Often includes randomness to add variety
  - E.g., a single tree pattern can be used to model an entire forest

Randomness

- Use some sort of randomness to make models more interesting, natural, less uniform
- *Pseudorandom* number generation algorithms
  - Produce a sequence of (apparently) random numbers based on some initial seed value
  - rand() generates random number between 0 and 1
- Pseudorandom sequences are repeatable, as one can always reset the sequence
  - srand(seed) initializes the random number generator
  - If the seed value is changed, a different sequence of numbers will be generated
  - Non-repeatable sequences can be generated with srand((unsigned)time(NULL));

Lecture Overview

- Procedural Modeling
  - Concepts
  - Algorithms

Height Fields

Height Fields

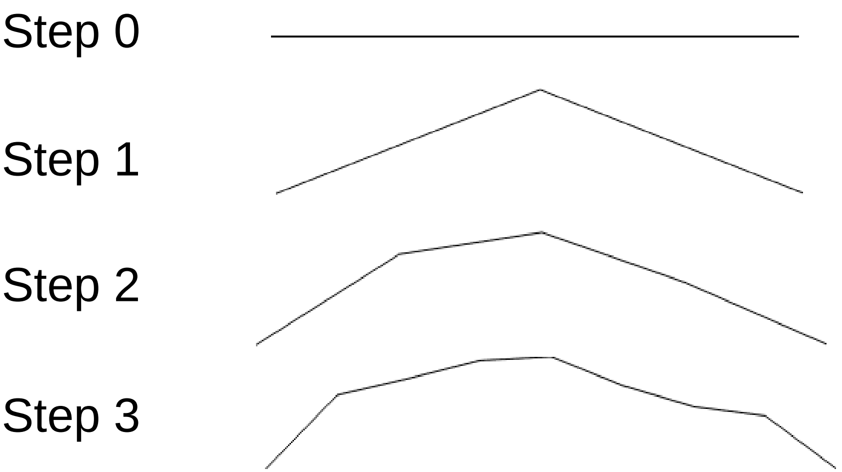
- Landscapes are often constructed as *height fields*
- Regular grid on the ground plane
- Store a height value at each point
- Can store large terrain in memory
  - No need to store all grid coordinates: inherent connectivity
- Shape terrain by operations that modify the height at each grid point
- Can generate height from grey scale values
  - Allows using image processing tools to create terrain height

Midpoint Displacement Algorithm

- Random midpoint displacement algorithm (one-dimensional)

```
Start with single horizontal line segment.
Repeat for sufficiently large number of times
{
   Repeat over each line segment in scene
   {
      Find midpoint of line segment.
      Displace midpoint in Y by random amount.
      Reduce range for random numbers.
   }
}
```

- Similar for triangles, quadrilaterals

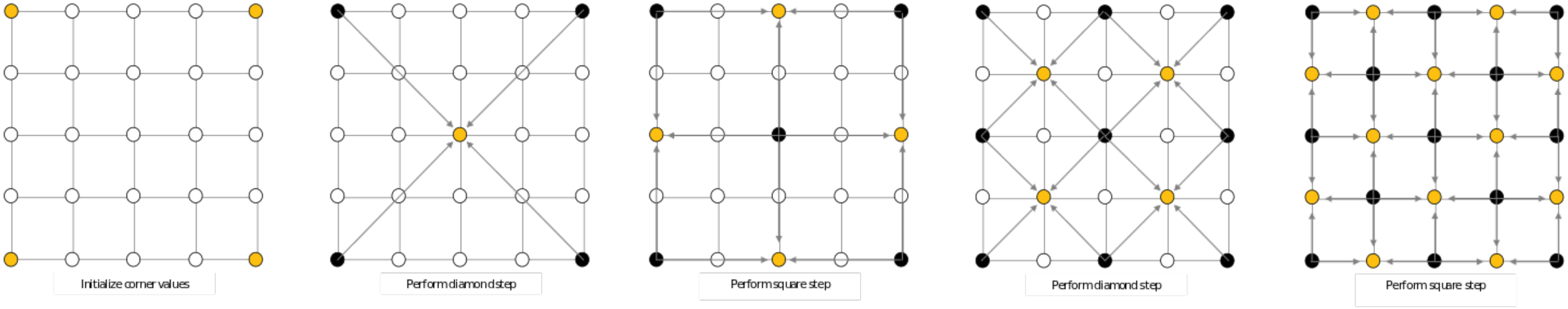


Result: Mountain Range

Source: http://gameprogrammer.com/fractal.html#midpoint

Diamond Square Algorithm

- Begins with a 2D array of size $2^n + 1$
- Four corner points must be set to initial values.
- Perform diamond and square steps alternatingly:
    - The diamond step: for each square in the array, set the midpoint of that square to be the average of the four corner points plus a random value.
    - The square step: for each diamond in the array, set the midpoint of that diamond to be the average of the four corner points plus a random value.
        - Points located on edges of the array will have only three adjacent values set rather than four: take their average.
- At each iteration, the magnitude of the random value should be reduced.



Initialize corner values

Perform diamond step

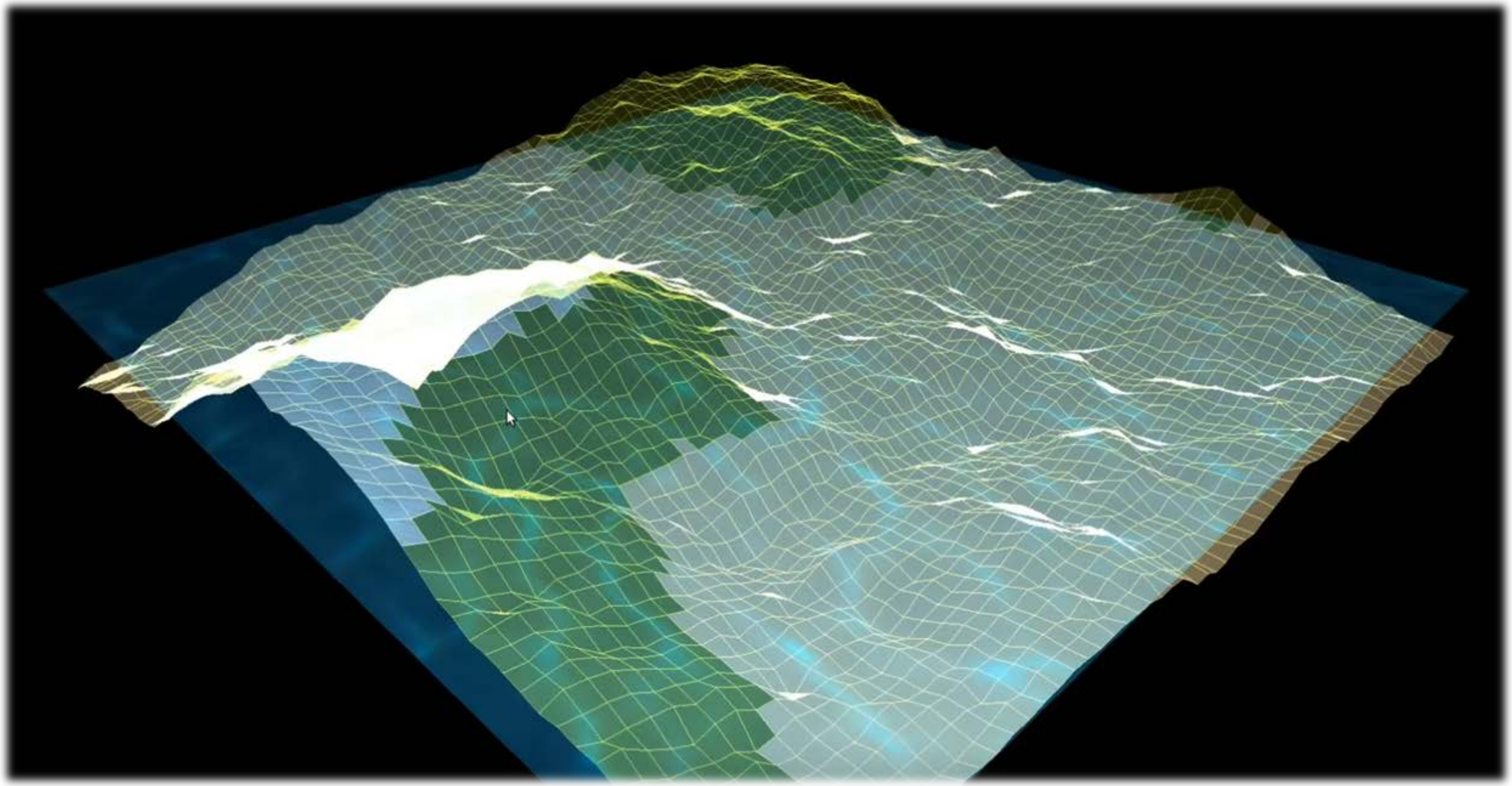
Perform square step

Perform diamond step

Perform square step

Diamond Square Algorithm: Video

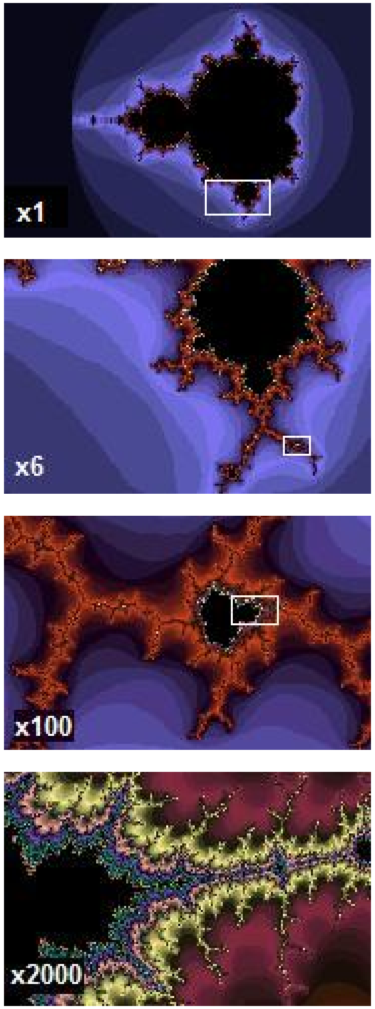
- https://www.youtube.com/watch?v=9HJKrctqIJI



Fractals

Fractals

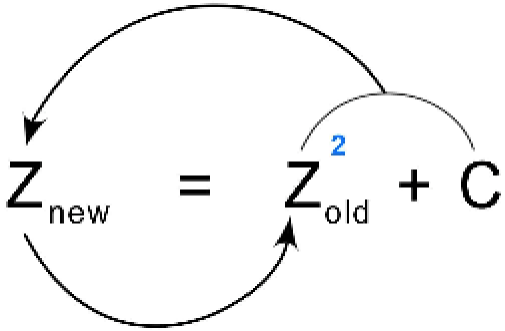
- Fractal:
  Fragmented geometric shape which can be split into parts, each of which is (at least approximately) a smaller size copy of the whole
- Self-similarity



From Wikipedia

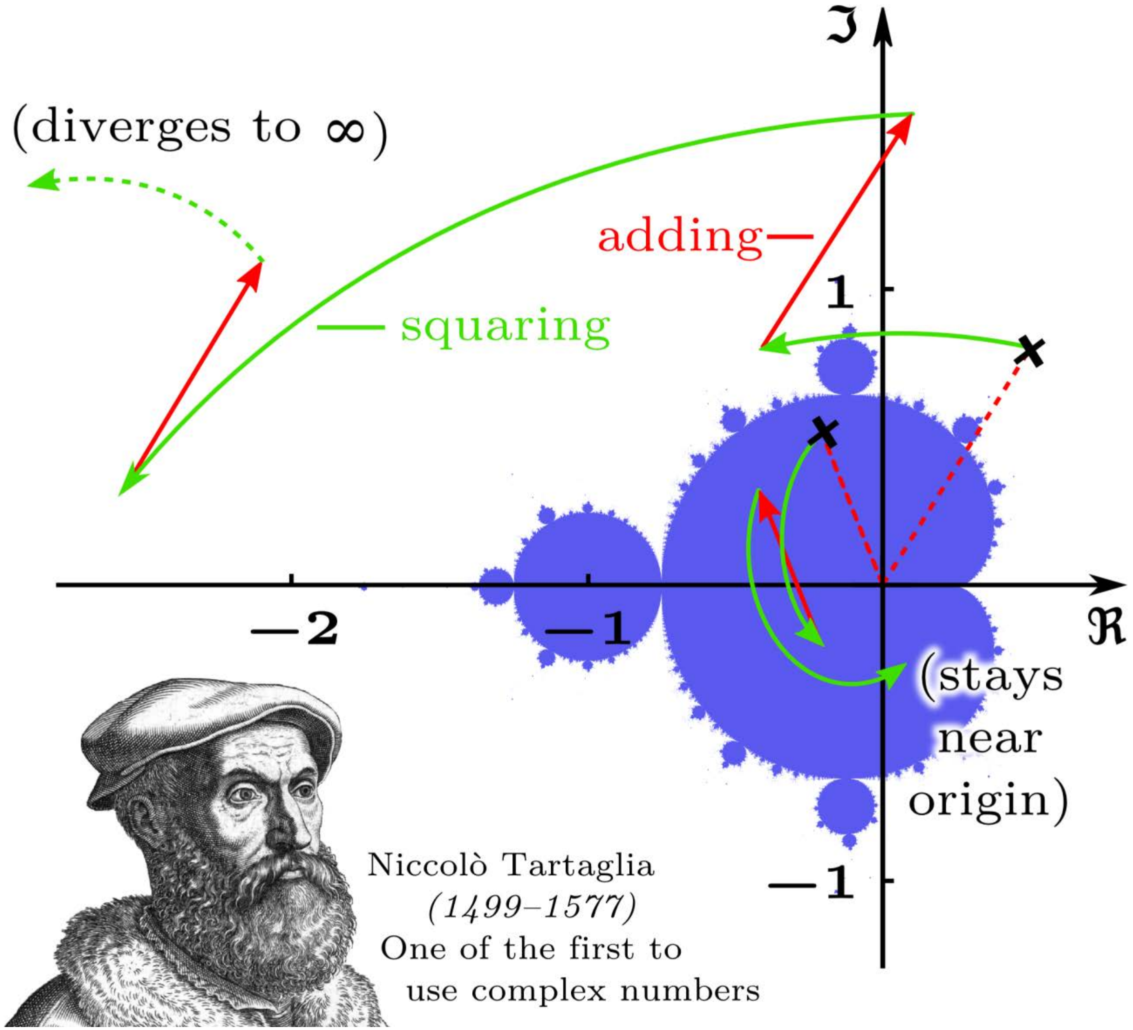
Mandelbrot Set

- Z and C are complex numbers
- Initialize Z with 0+0i
- Pick any C



$$Z_{new} = Z_{old}^{2} + C$$

- If C is diverting to infinity it is not part of the Mandelbrot set
- Otherwise it is



Niccolò Tartaglia
(1499–1577)
One of the first to use complex numbers

Demo: http://www.scale18.com/canvas2.html

Mandelbrot Set: Video

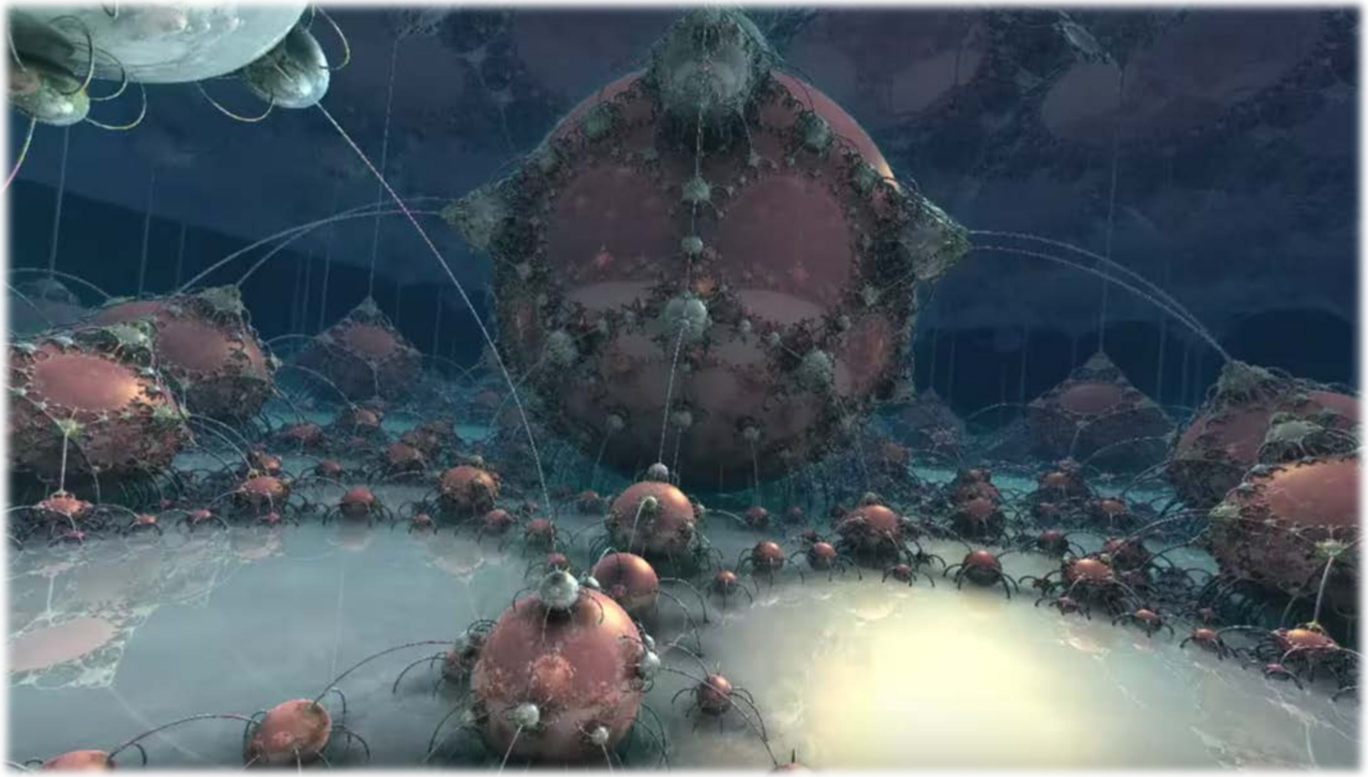
- The Hardest Mandelbrot Zoom in 2017 – New record, 750 000 000 iterations
  - https://www.youtube.com/watch?v=aSg2Db3jF_4

Mandelbox

- A fractal with a boxlike shape found by Tom Lowe in 2010
- Defined in a similar way to the Mandelbrot set
- Can be defined in any number of dimensions, typically drawn in three dimensions for illustrative purposes
- Video
  - http://www.youtube.com/watch?v=0clz6WLfWaY

Real-Time Fractal Rendering

- Trip inside a 3D fractal (Kleinian) GPU realtime rendering
  - https://www.youtube.com/watch?v=XIzScwydxOE



Fractal Landscapes

- Add textures, material properties; use nice rendering algorithm
- Example: Terragen Classic (free software) http://www.planetside.co.uk/terragen/

[http://www.planetside.co.uk/gallery/f/tg09]

L-Systems

L-Systems

- Developed by biologist Aristid Lindenmayer in 1968 to study growth patterns of algae
- Defined by grammar

$$\mathbf{G} = \{V, S, \omega, P\}$$

  - V = alphabet, set of symbols that can be replaced (variables)
  - S = set of symbols that remain fixed (constants)
  - ω = string of symbols defining initial state
  - P = production rules
- Stochastic L-system
  - If there is more than one production rule for a symbol, randomly choose one

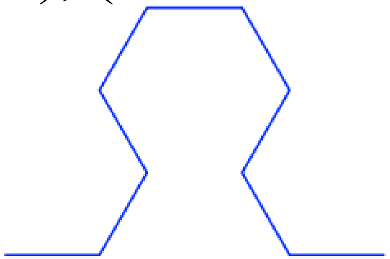
Turtle Interpretation for L-Systems

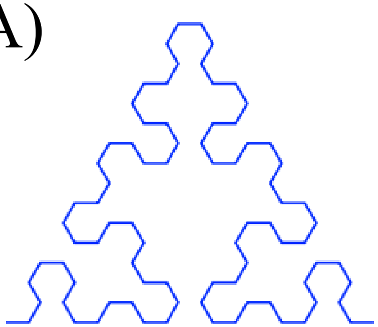
- Origin: functional programming language Logo
  - Dialect of Lisp
  - Designed for education: drove a mechanical turtle as an output device
- Turtle interpretation of strings
  - State of turtle defined by (x, y, α) *for* position and heading
  - Turtle moves by step size *d* and angle increment δ
- Sample Grammar
  - F: move forward a step of length *d*
    New turtle state: (x', y', α)
    $x' = x + d \cos \alpha$
    $y' = y + d \sin \alpha$
    A line segment between points (x, y) and (x', y') is drawn.
  - +: Turn left by angle δ. Next state of turtle is (x, y, $\alpha+\delta$)
    Positive orientation of angles is counterclockwise.
  - −: Turn right by angle δ. Next state of turtle is (x, y, α-δ)

Example: Sierpinski Triangle

- Variables: A, B
  - Draw forward
- Constants: + , -
  - Turn left, right by 60 degrees
- Start: A
- Rules: (A→B-A-B), (B→A+B+A)

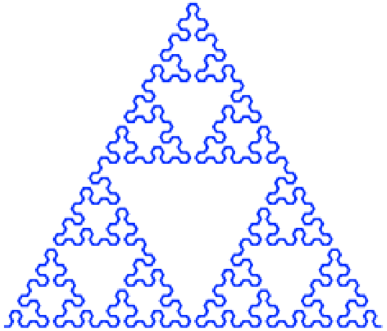
2 iterations

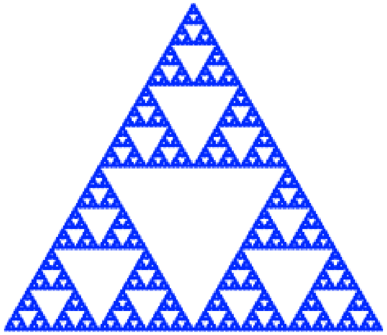




4 iterations

6 iterations

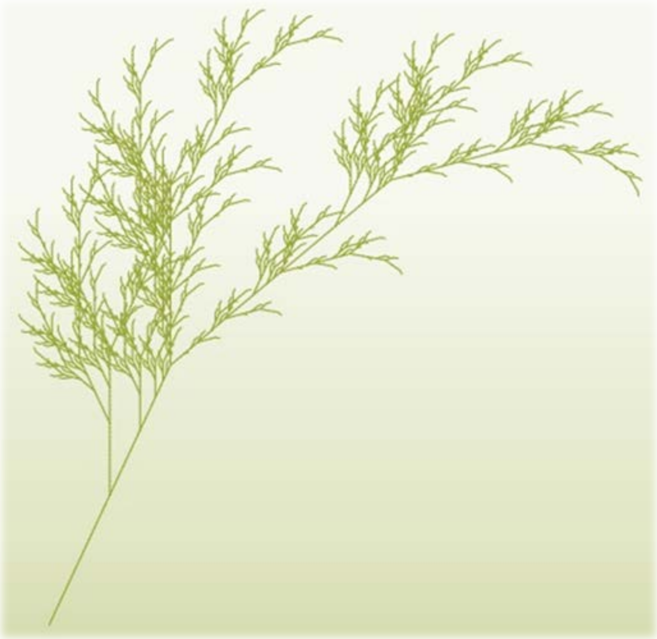




9 iterations

Example: Fern

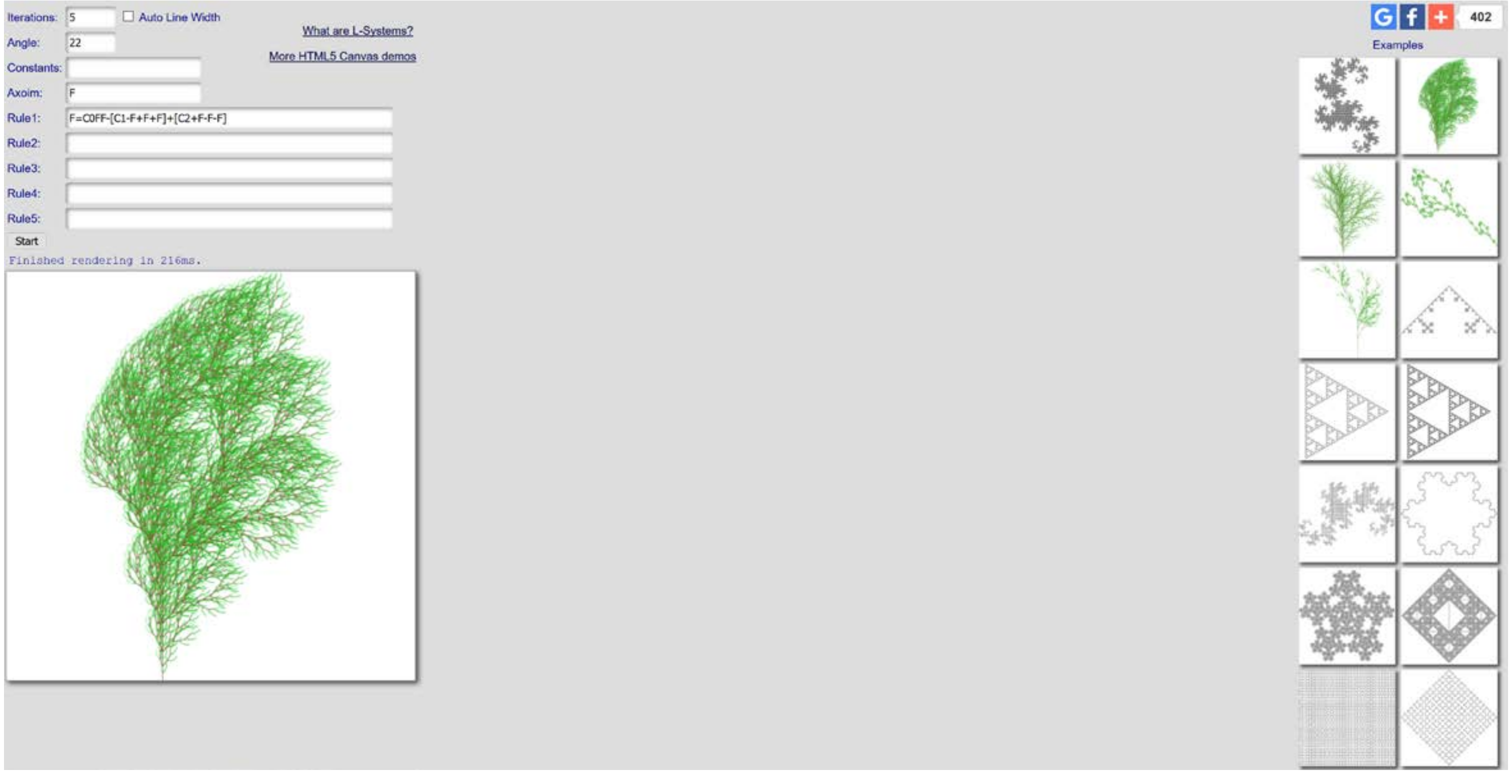
- Variables: X, F
  - X: no drawing operation
  - F: move forward
- Constants: +, −
  - Turn left, right
- Start: X
- Rules:
  (X → F-[[X]+X]+F[+FX]-X),(F → FF)



[Wikipedia]

Demo

- http://www.kevs3d.co.uk/dev/lsystems/



Iterations: 5 ☐ Auto Line Width

Angle: 22

Constants:

Axoim: F

Rule1: F=C0FF-[C1-F+F+F]+[C2+F-F-F]

Rule2:

Rule3:

Rule4:

Rule5:

Start

```
Finished rendering in 216ms.
```

Fractal Trees

- Tutorial for recursive generation of trees in 3D: http://web.comhem.se/solgrop/3dtree.htm
  - Model trunk and branches as cylinders
  - Change color from brown to green at certain level of recursion

Dragon Curve Tree

Some determinstic 3D branching plants.

Source: Allen Pike

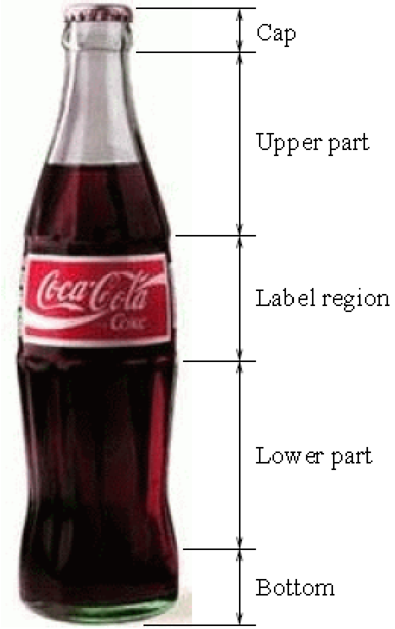
Shape Grammar

Shape Grammar

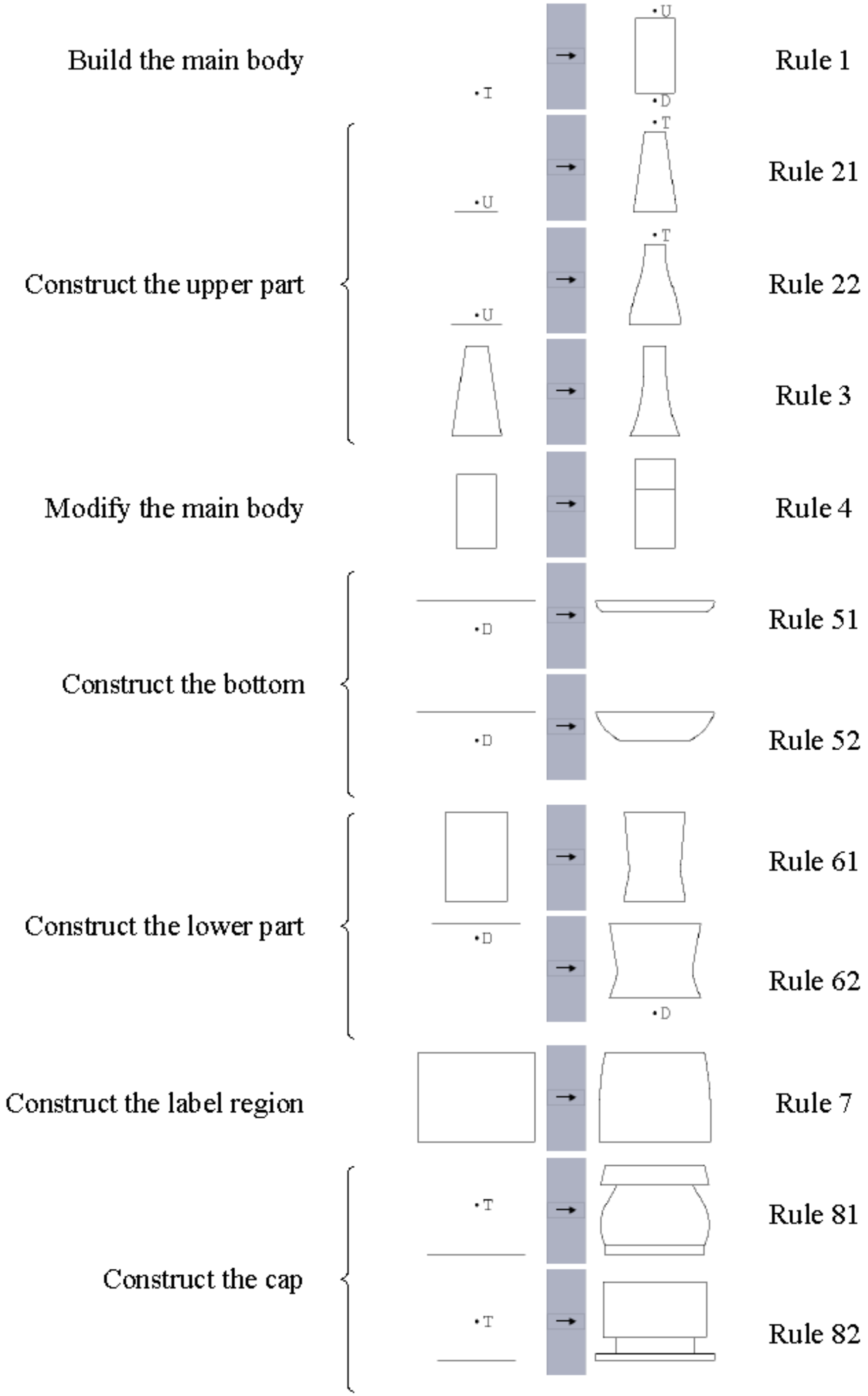
- Shape Rules
  - Defines how an existing shape can be transformed
- Generation Engine
  - Performs the transformations
- Working Area
  - Displays created geometry

Example:
Coca-Cola Bottle

Evolution of Coca-Cola bottles



Division of a Coca-Cola bottle



Build the main body — Rule 1

Construct the upper part — Rule 21, Rule 22, Rule 3

Modify the main body — Rule 4

Construct the bottom — Rule 51, Rule 52

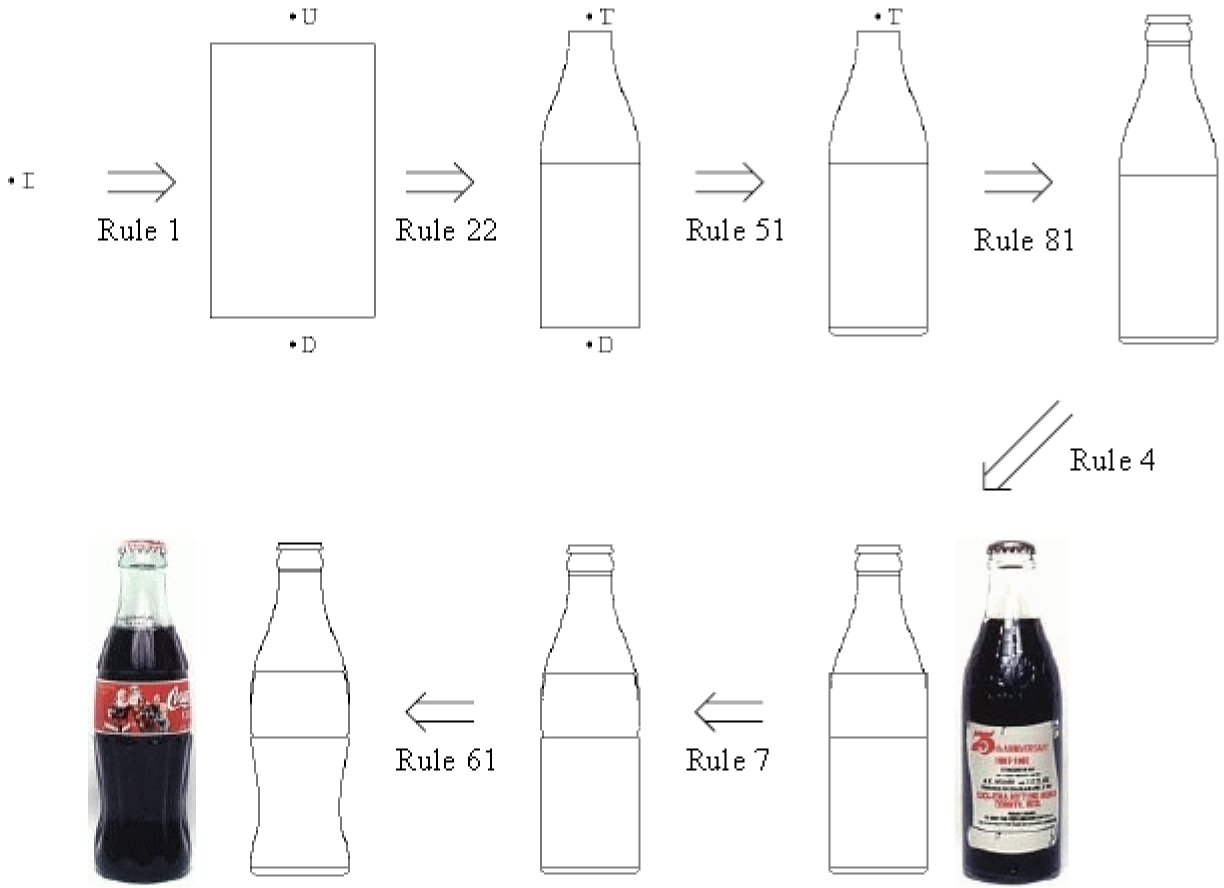
Construct the lower part — Rule 61, Rule 62

Construct the label region — Rule 7

Construct the cap — Rule 81, Rule 82

Shape Computation Example

- Shape computation for two existing Coca-Cola bottles



Source: Chau et al.: “Evaluation of a 3D Shape Grammar Implementation”, *Design Computing and Cognition'04,* pp. 357-376

Demonstration: Procedural Buildings

- Demo fr-041: debris by Farbrausch, 2007
- http://www.youtube.com/watch?v=wqu_IpkOYBg&hd=1
- Single, 177 KB EXE file!
- http://www.farbrausch.de/