CSE 167:
Introduction to Computer Graphics
Lecture #16: Procedural Modeling

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2014

Announcements

- Final Project due December 18th at 3pm
  - First Blog entry due tomorrow at midnight
  - Bezier patch moved to easy and only if not done in HA6
- TA Evaluations
- CAPE
- Siggraph student volunteer program
  - http://s2014.siggraph.org/volunteers/student-volunteers
- Midterm #2 returned today
  - Scores are on Ted
  - Please return after class if score change requested

UCSD

Midterm

Statistics	Midterm 1	Midterm 2
Average	65.9%	67.8%
Median	66.1%	67.5%
Max	97%	94%
Min	11%	21%
# Participants	118	99

UCSD

Lecture Overview

- Procedural Modeling
  - Concepts
  - Algorithms

UCSD

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, need extremely complex models containing millions or billions of primitives
- Modeling everything manually is extremely tedious

UCSD

Alternatives

- **Data-driven modeling**
  - Scan model geometry from real world examples
  - Use laser scanners or similar devices
  - Use photographs as textures
  - Archives of 3D models
    - http://www-graphics.stanford.edu/data/3Dscanrep/
    - Reader for PLY point file format: http://w3.impa.br/~diego/software/rply/

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

Photograph   Rendering
[Levoy et al.]

UCSD

Procedural Modeling

[Deussen et al.]

- 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
- Usually defined by a small set of data, or rules, that describes the overall properties of the model
  - 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

UCSD

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
- Pseudorandom sequences are repeatable, as one can always reset the sequence
  - E.g., if a tree is built using pseudorandom numbers, then the entire tree can be rebuilt by resetting the seed value
  - If the seed value is changed, a different sequence of numbers will be generated, resulting in a (slightly) different tree

UCSD

Recursion

▶ Repeatedly apply the same operation (set of operations) to an object

▶ Generate self-similar objects: fractals

   ▶ Objects which look similar when viewed at different scales

▶ For example, the shape of a coastline may appear as a jagged line on a map

   ▶ As we zoom in, we see that there is more and more detail at finer scales

   ▶ We always see a jagged line no matter how close we look at the coastline

UCSD

Lecture Overview

▶ Procedural Modeling

   ▶ Concepts

   ▶ Algorithms

UCSD

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

UCSD

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.
    }
}
```
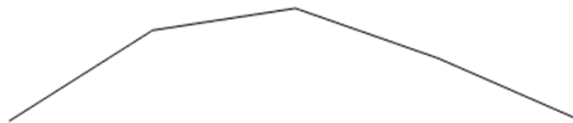
▸ Similar for triangles, quadrilaterals
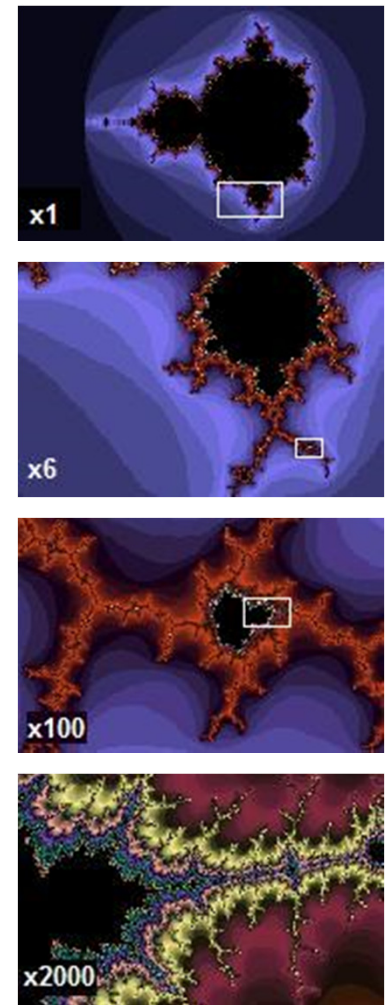
Step 0

Step 1
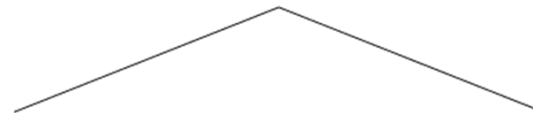
Step 2

Step 3

Result: Mountain Range

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

UCSD

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

- Demo: Mandelbrot Set
  http://www.scale18.com/canvas2.html

From Wikipedia

UCSD

Video

- ## 3D Mandelbrot Zoom

  - http://www.youtube.com/watch?v=0clz6WLfWaY

UCSD

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]

UCSD

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

UCSD

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 $(x, y, \alpha-\delta)$
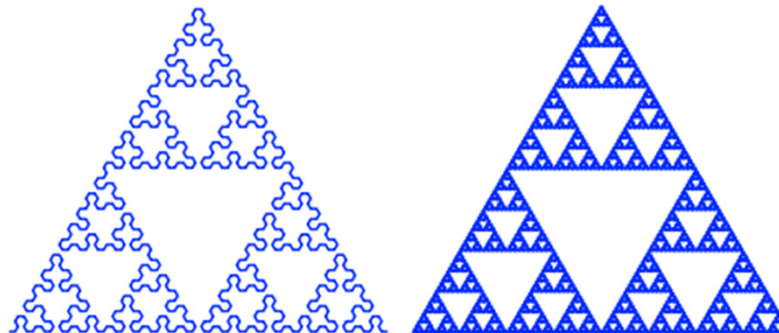
UCSD

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
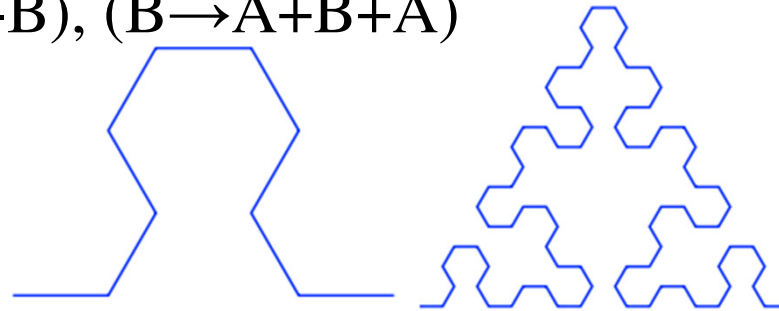
4 iterations

6 iterations

9 iterations

UCSD

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]

UCSD

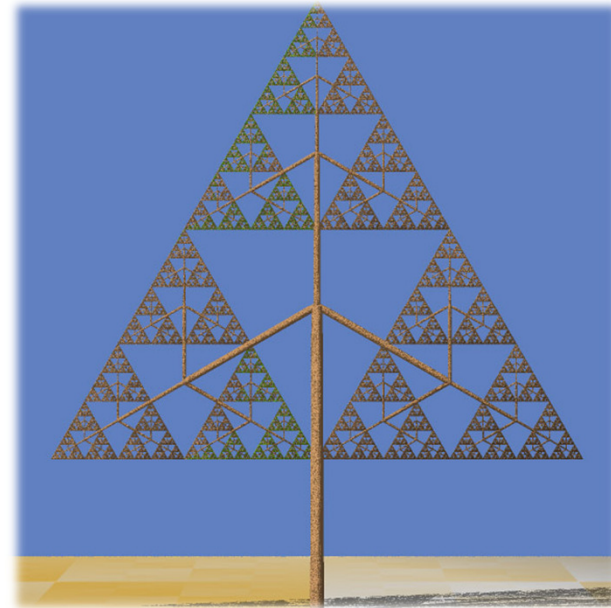# Fractal Trees

- ## 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

Sierpinski Tree
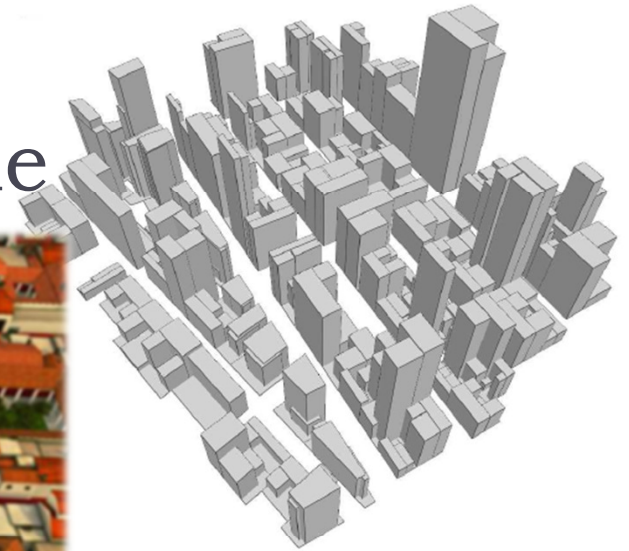
UCSD

Algorithmic Beauty of Plants

▸ Book "The Algorithmic Beauty of Plants" by Przemyslaw Prusinkiewicz and Aristid Lindenmayer, 2004

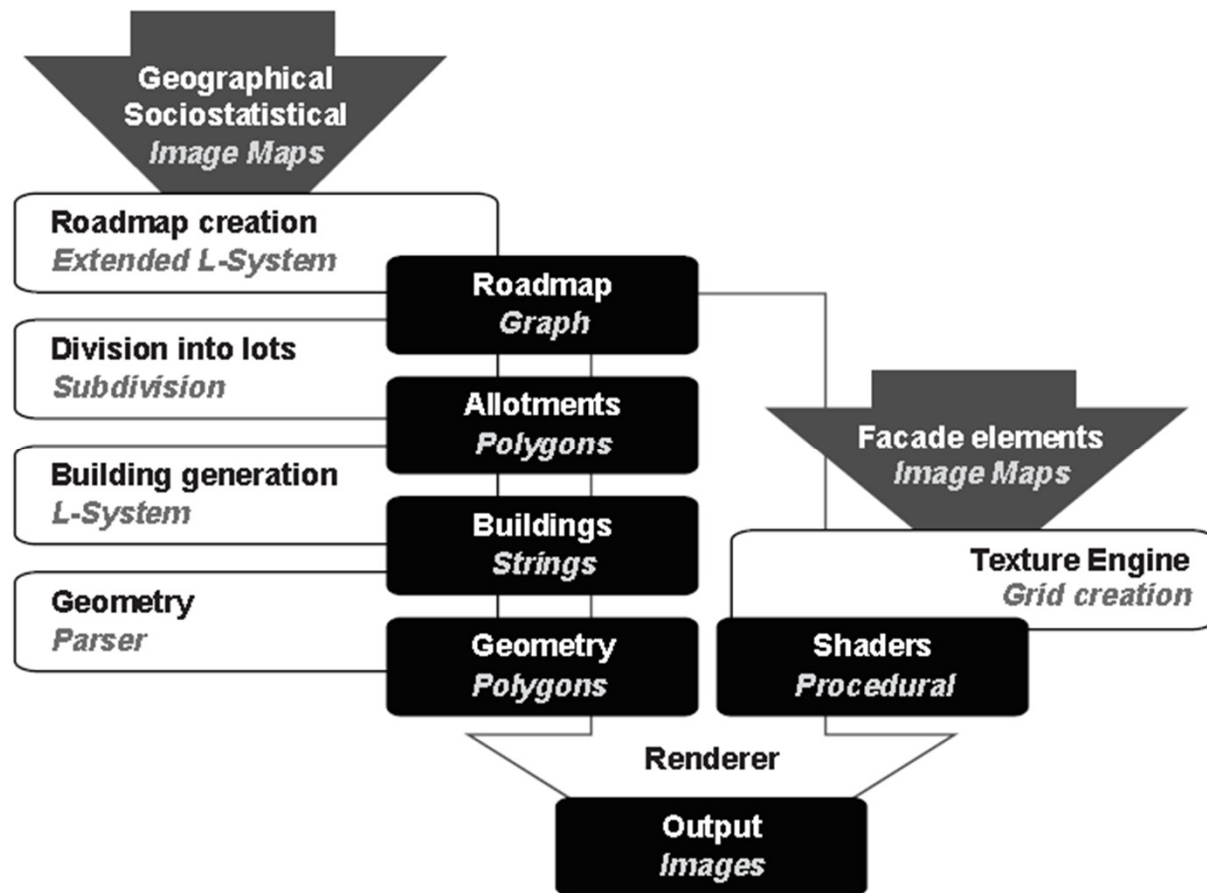▸ On-Line at: http://algorithmicbotany.org/papers/#abop

[Prusinkiewicz, http://algorithmicbotany.org/papers/positional.sig2001.pdf]

UCSD

Buildings, Cities: CityEngine

http://www.esri.com/software/cityengine/

UCSD

CityEngine: Pipeline

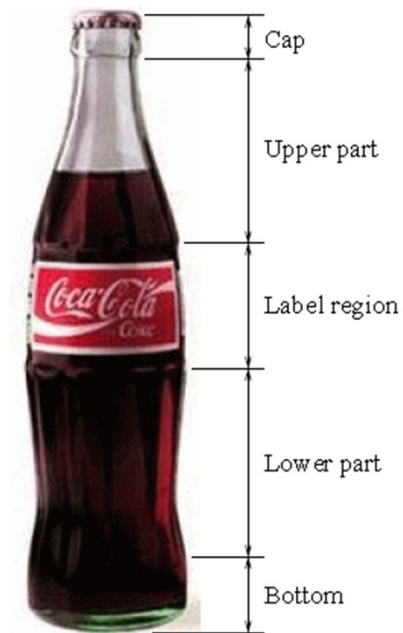Parish, Mueller: "Procedural Modeling of Cities", ACM Siggraph 2001

UCSD

Shape Grammar

▸ **Shape Rules**

  ▸ Defines how an existing shape can be transformed

▸ **Generation Engine**

  ▸ Performs the transformations
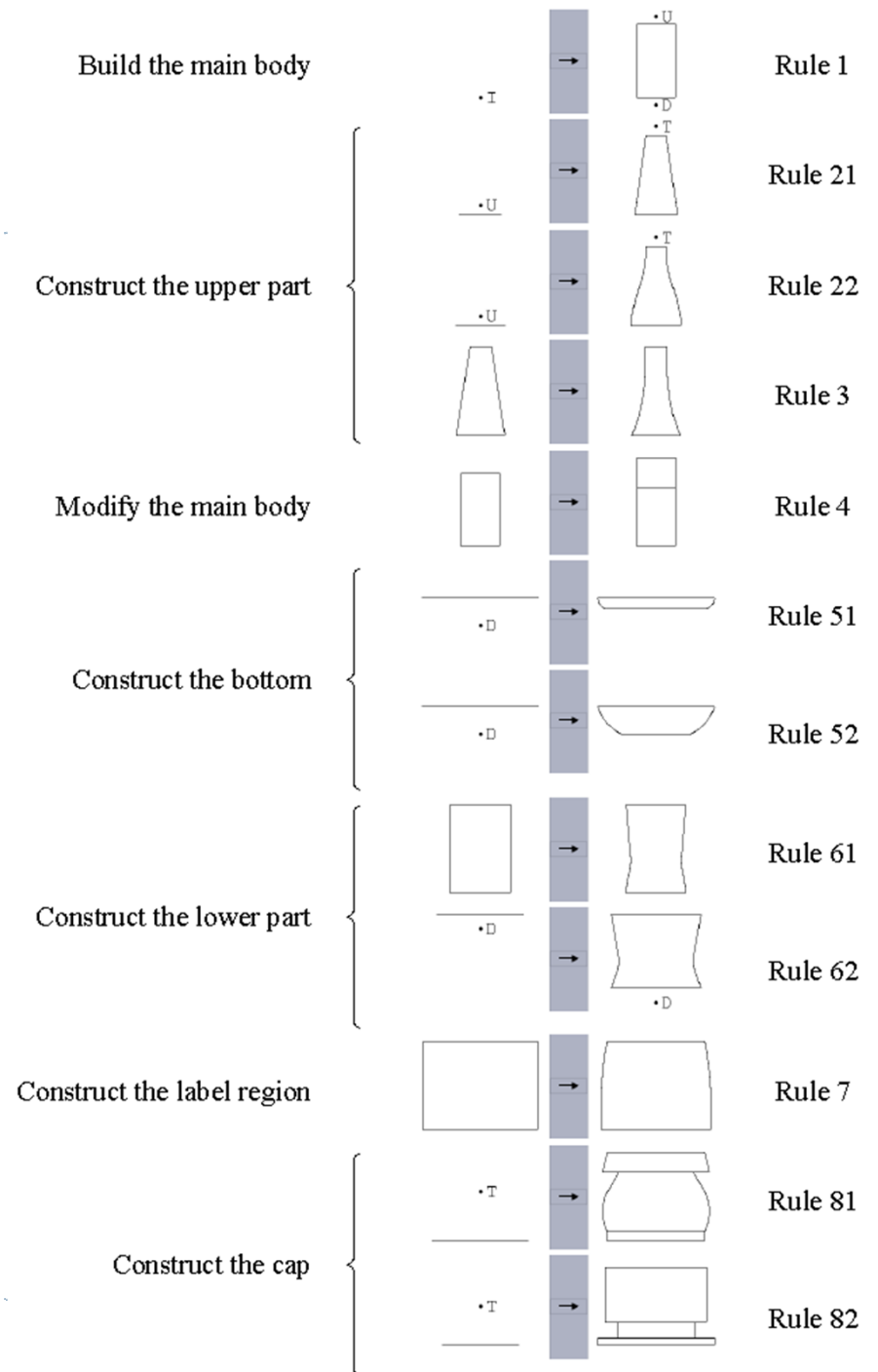
▸ **Working Area**

  ▸ Displays created geometry

UCSD

Example: Coca-Cola Bottle
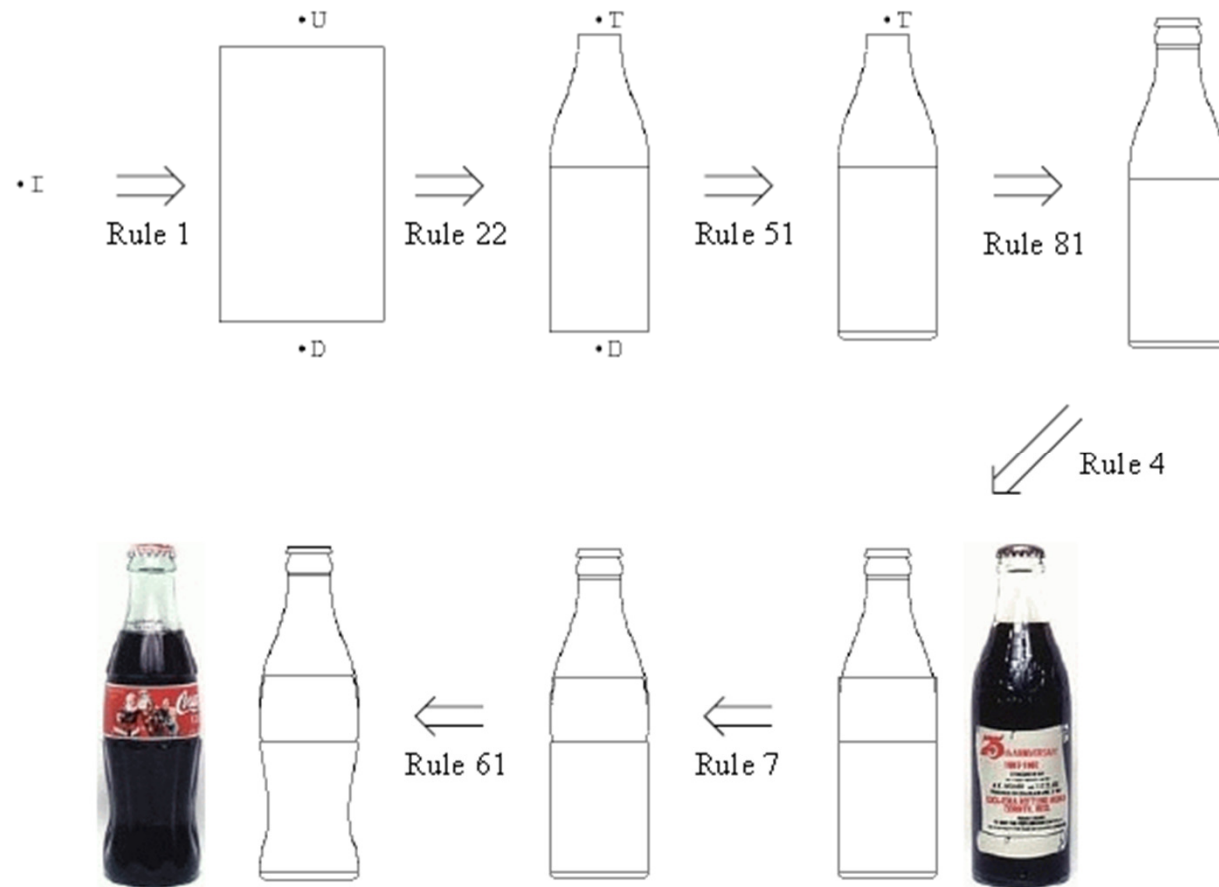
Evolution of Coca-Cola bottles

Division of a Coca-Cola bottle

Cap

Upper part

Label region

Lower part

Bottom

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

25

Shape Computation Example

▸ Shape computation for two existing Coca-Cola bottles

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/

UCSD