

CSE 167:  
Introduction to Computer Graphics  
Lecture #14: Procedural Modeling

Jürgen P. Schulze, Ph.D.  
University of California, San Diego  
Fall Quarter 2015

# Announcements

---

- ▶ Project 6 due tomorrow
- ▶ Monday: Midterm discussion
- ▶ Next Thursday: Midterm #2
- ▶ Final project description released tomorrow evening

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

# 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.]

# 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
- ▶ 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



[Deussen et al.]

# 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

# 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



# Lecture Overview

---

- ▶ Procedural Modeling
  - ▶ Concepts
  - ▶ Algorithms

# 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)

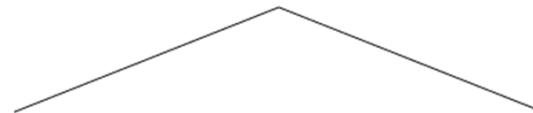
```
Start with single horizontal line segment.  
Repeat for sufficiently large number of times  
{  
  Repeat over each line segment in scene  
  {  
    Find midpoint of line segment.  
    Displace midpoint in Y by random amount.  
    Reduce range for random numbers.  
  }  
}
```

## ► Similar for triangles, quadrilaterals

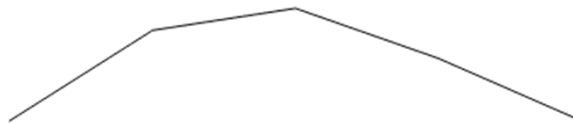
Step 0



Step 1



Step 2



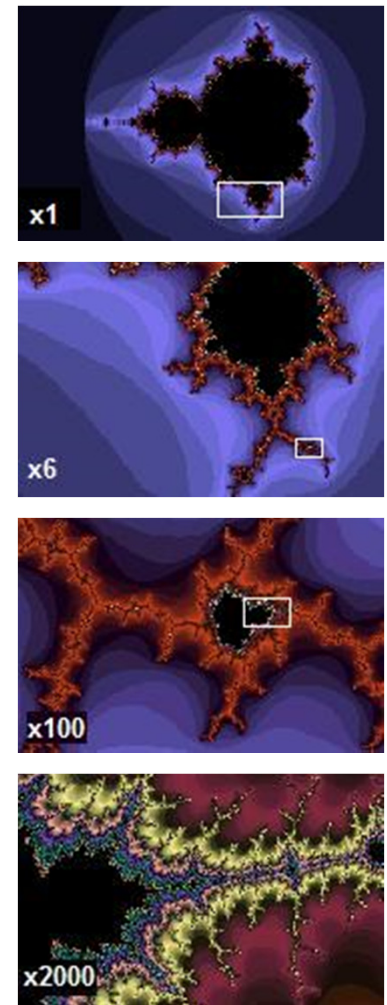
Step 3



Result: Mountain Range

# 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

# Video

---

- ▶ 3D Mandelbrot Zoom

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



# Fractal Landscapes

---

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



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

# L-Systems

---

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

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

- ▶  $V$  = alphabet, set of symbols that can be replaced (variables)
- ▶  $S$  = set of symbols that remain fixed (constants)
- ▶  $\omega$  = 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

---

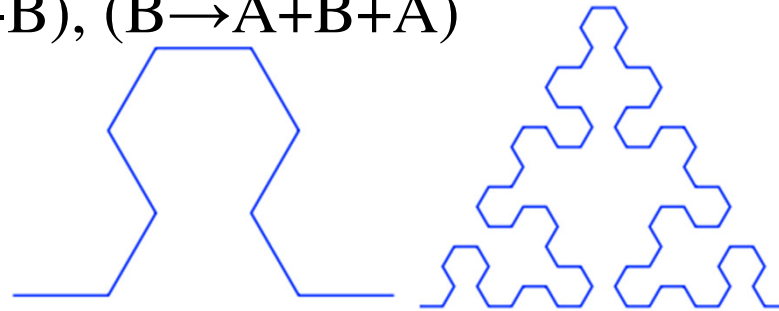
- ▶ Origin: functional programming language Logo
  - ▶ Dialect of Lisp
  - ▶ Designed for education: drove a mechanical turtle as an output device
- ▶ Turtle interpretation of strings
  - ▶ State of turtle defined by  $(x, y, \alpha)$  for position and heading
  - ▶ Turtle moves by step size  $d$  and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



# Example: Sierpinski Triangle

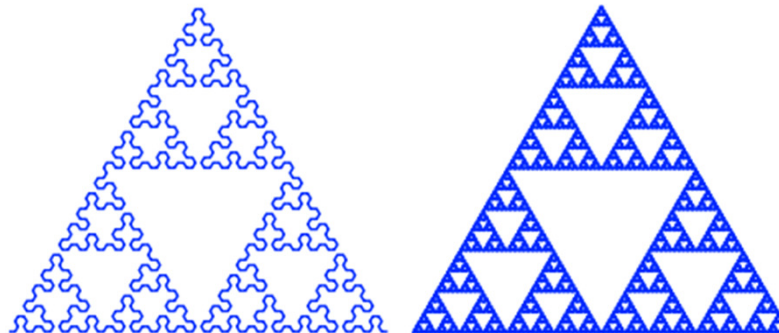
- ▶ Variables: A, B
  - ▶ Draw forward
- ▶ Constants: + , -
  - ▶ Turn left, right by 60 degrees
- ▶ Start: A
- ▶ Rules:  $(A \rightarrow B-A-B)$ ,  $(B \rightarrow A+B+A)$

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:**  
 $(X \rightarrow F-[[X]+X]+F[+FX]-X), (F \rightarrow FF)$



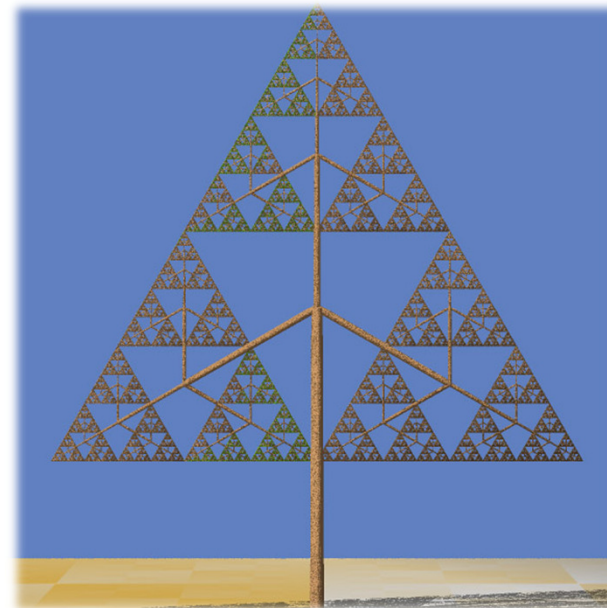
[Wikipedia]

# 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

# Algorithmic Beauty of Plants

---

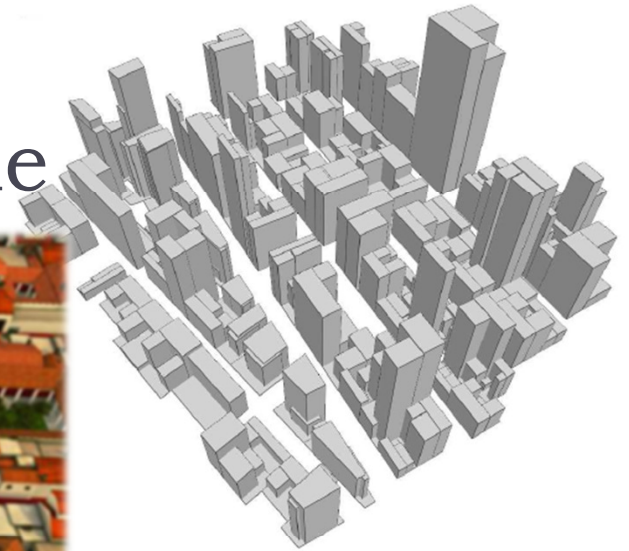
- ▶ Book “The Algorithmic Beauty of Plants” by Przemyslaw Prusinkiewicz and Aristid Lindenmayer, 2004
- ▶ On-Line at: <http://algorithmicbotany.org/papers/#abop>



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

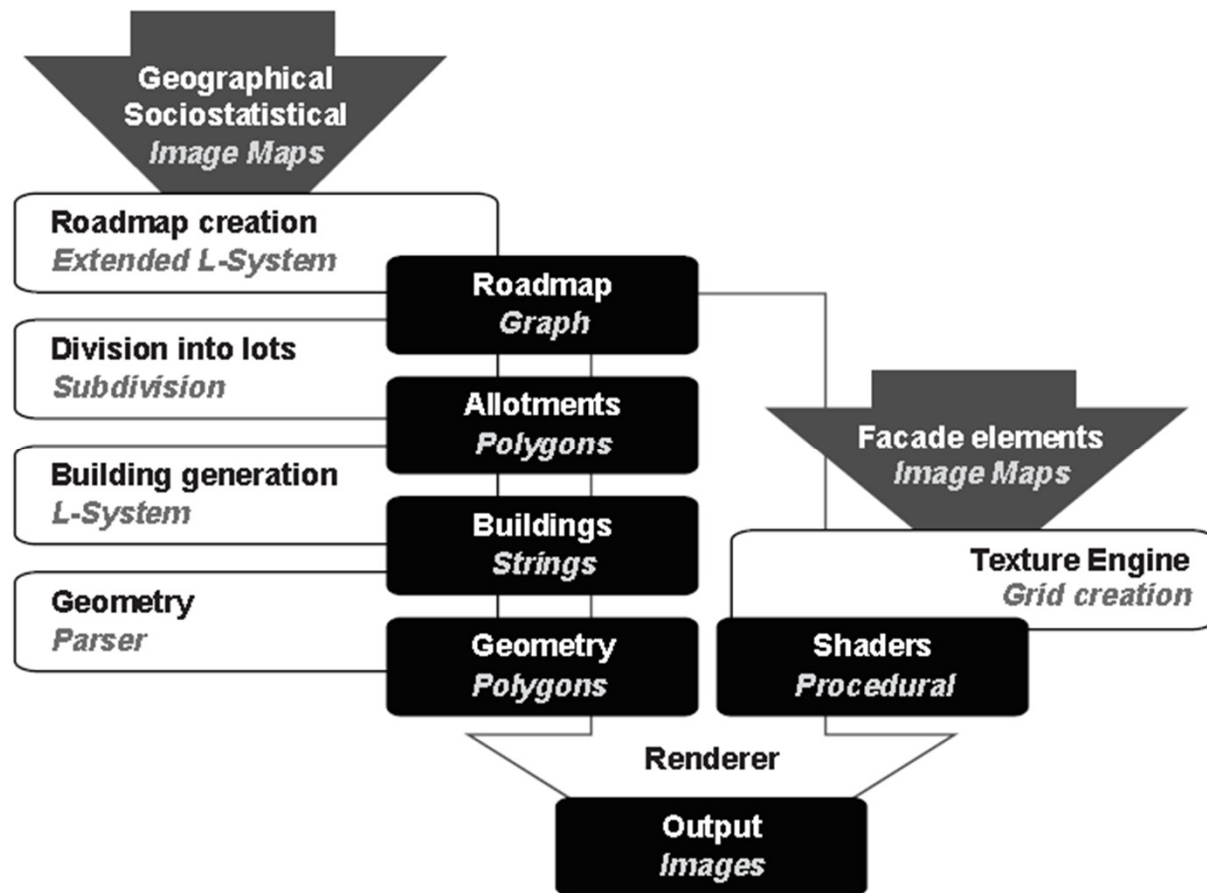


# Buildings, Cities: CityEngine



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

# CityEngine: Pipeline



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

# Shape Grammar

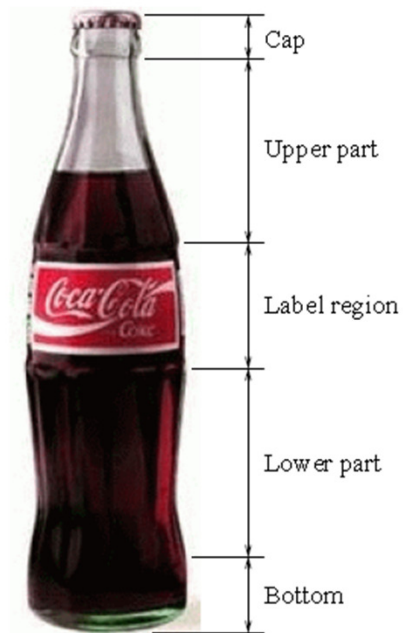
---

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

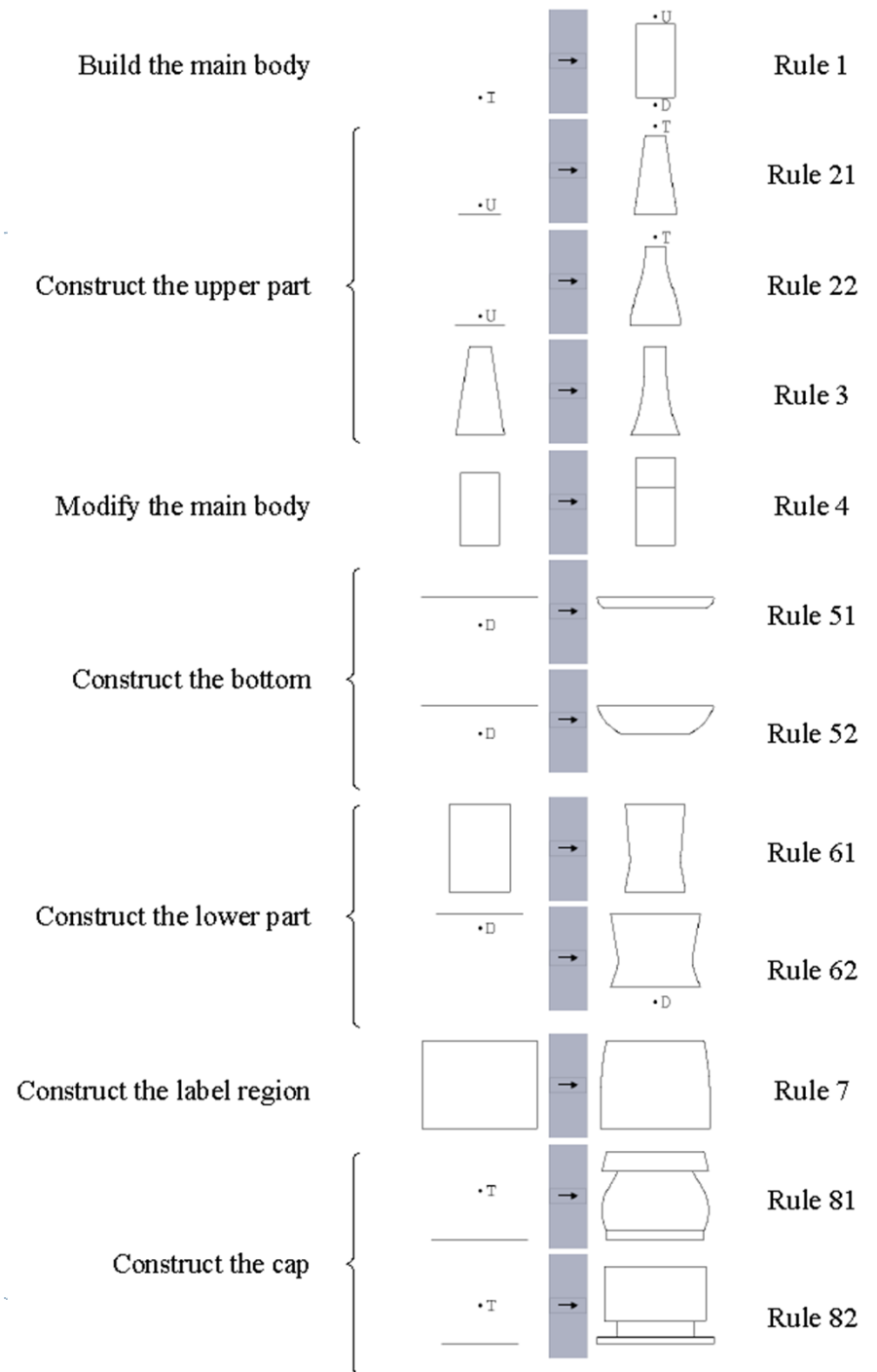
# Example: Coca-Cola Bottle



Evolution of Coca-Cola bottles



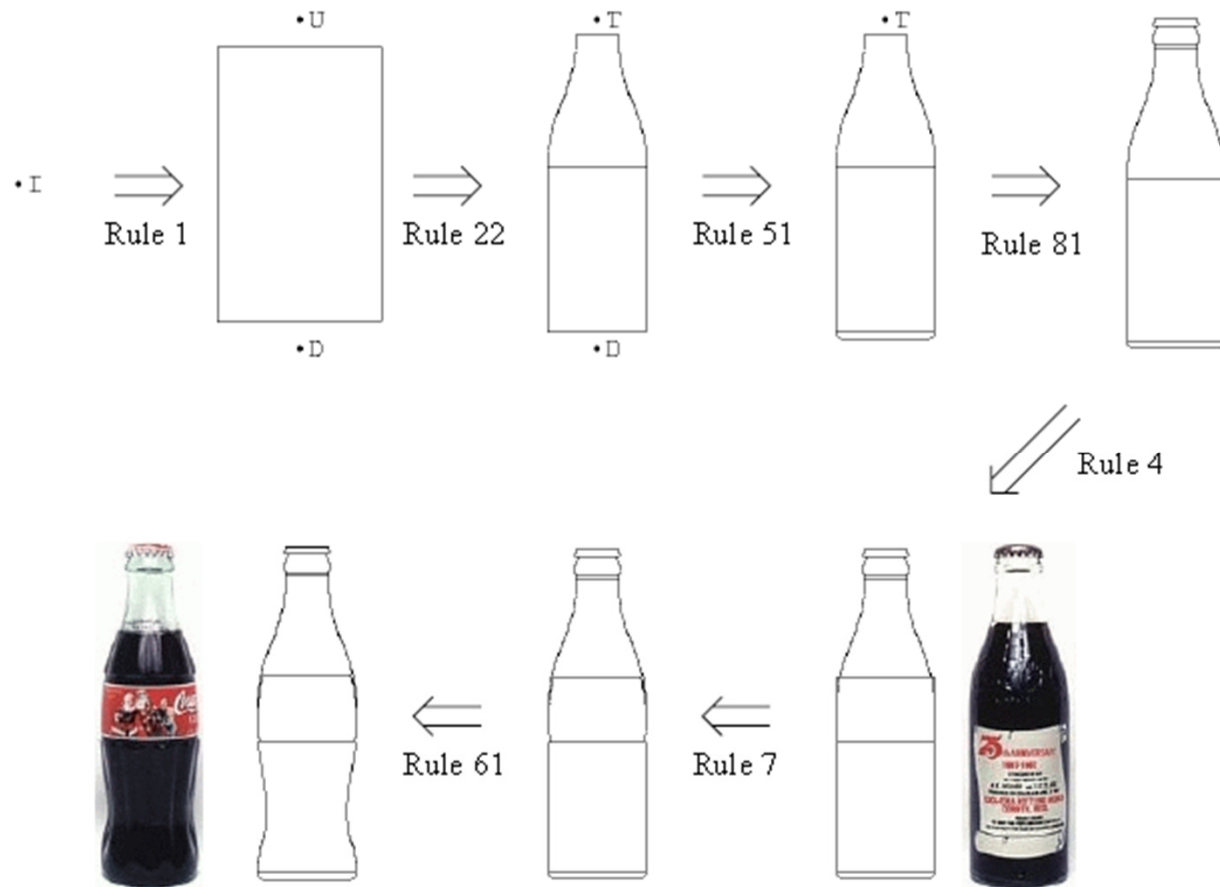
Division of a Coca-Cola bottle





# 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\\_lpkOYBg&hd=1](http://www.youtube.com/watch?v=wqu_lpkOYBg&hd=1)
- ▶ Single, 177 KB EXE file!
- ▶ <http://www.farbrausch.de/>

