Procedural Modeling

CS334 Spring 2021

Daniel G. Aliaga
Department of Computer Science
Purdue University
Procedural Modeling

• Apply algorithms for producing objects and scenes
• The rules may either be embedded into the algorithm, configurable by parameters, or externally provided
Procedural Modeling

• Fractals
• Terrains
• Image-synthesis
  – Perlin Noise
  – Clouds
• Plants
• Cities
• And procedures in general...
Fractals

• Consider a simple line fractal
  – Split a line segment, randomize the height of the midpoint by some number in the $[-r,r]$ range
  – Repeat and randomize by $[-r/2,r/2]$
  – Continue until a desired number of steps, randomizing by half as much each step
Fractals and Terrains

• A similar process can be applied to squares in the xz plane
  – At each step, an xz square is subdivided into 4 squares, and the y component of each new point is randomized
  – By repeating this process recursively, we can generate a mountain landscape
Terrains

- A similar process can be applied to squares in the xz plane
  - At each step, an xz square is subdivided into 4 squares, and the y component of each new point is randomized
  - By repeating this process recursively, we can generate a mountain landscape
Image Synthesis

• Procedurally generate an image (pixels)
Idea: Perlin Noise

• Procedurally generate noise
  – http://js1k.com/demo/543
City Modeling

• Procedural Modeling of Cities
  (more on this later...)
Plant Modeling

• **The Algorithmic Beauty of Plants**
Background: Chomsky Hierarchy

• Type 0 grammars
  – Unrestricted, recognized by Turing machine

• Type 1 grammars
  – Context-sensitive grammars

• Type 2 grammars
  – Context-free grammars

• Type 3 grammars
  – Regular grammars (e.g., regular expressions)
Lindenmayer system (or L-system)

- A context-free or context-sensitive grammar
- All rules are applied in “every iteration” before jumping to the next level/iteration
- Can be deterministic or non-deterministic
L-system

- Variables: a
- Constants: +, - (rotations of + or − 90 degrees)
- Initial string (axiom): s=a
- Rules: a → a+a-a-a+a
(Context-Free) L-system for Plants

Figure 1.24: Examples of plant-like structures generated by bracketed OL-systems. L-systems (a), (b) and (c) are edge-rewriting, while (d), (e) and (f) are node-rewriting.
L-system for Plants (stochastic)

Figure 1.27: Stochastic branching structures
L-system for Plants (3D)

\( n = 5, \delta = 18^\circ \)

\[
\begin{align*}
\omega & : \text{plant} \\
p_1 & : \text{plant} \rightarrow \text{internode} + \left[ \text{plant} + \text{flower} \right] - - // \\
& - - \text{leaf} \text{ internode} + + \text{leaf} - \\
& \text{plant} \text{ flower} + + \text{plant} \text{ flower} \\
p_2 & : \text{internode} \rightarrow \text{F seg} [/ & & \& \text{leaf}] [/ & & \& \text{leaf}] \text{F seg} \\
p_3 & : \text{seg} \rightarrow \text{seg F seg} \\
p_4 & : \text{leaf} \rightarrow [ ^ \{ +f-f-f+f+ +f+f-f-f \} ] \\
p_5 & : \text{flower} \rightarrow [ ^ \{ & & & & \text{pedicel} / \text{wedge} / / / / \text{wedge} / / / / \text{wedge} / / / / \text{wedge} ] \\
p_6 & : \text{pedicel} \rightarrow \text{FF} \\
p_7 & : \text{wedge} \rightarrow [ ^ \{ \& \& \& \& -f+f \} -f+f ]
\end{align*}
\]

Figure 1.26: A plant generated by an L-system

Figure 1.28: Flower field
Recent Result

- SIGGRAPH Asia 2020
- https://www.youtube.com/watch?v=MU9E7xJzVGs
Koch Snowflake
Demo

Shape Grammar

• Is used to generate geometric models from a set of shapes and rules

Shape Grammar
Shape Grammar

rule

DERIVATION

\[
\begin{align*}
\text{rectangle} & \rightarrow \text{grid} \\
\text{grid} & \rightarrow \text{grid} \\
\text{grid} & \rightarrow \text{grid} \\
\text{grid} & \rightarrow \text{grid} \\
\text{grid} & \rightarrow \text{grid} \\
\text{grid} & \rightarrow \text{grid}
\end{align*}
\]
Shape Grammar

OTHER DESIGNS IN THE LANGUAGE
Exercise: let’s make some art!

• What is a shape grammar that makes this?
Exercise: let’s make some art!

• Divide into teams
• What is the shape grammar that makes the art of the previous slide?
• Go!
Shape Grammar

Ice-ray grammar
Shape Grammar

Mughul garden grammar
Shape Grammar

• Style: Mediterranean
Cellular Automata

- A cellular automata (CA) is a spatial lattice of \( N \) cells, each of which is one of \( k \) states at time \( t \).
- Each cell follows the same simple rule for updating its state.
- The cell's state \( s \) at time \( t+1 \) depends on its own state and the states of some number of neighbouring cells at \( t \).
- For one-dimensional CAs, the neighbourhood of a cell consists of the cell itself and \( r \) neighbours on either side. Hence, \( k \) and \( r \) are the parameters of the CA.
- CAs are often described as discrete dynamical systems with the capability to model various kinds of natural discrete or continuous dynamical systems.
Types of Neighborhoods

Many more neighborhood techniques exist!
Classes of cellular automata (Wolfram)

- **Class 1**: after a finite number of time steps, the CA tends to achieve a unique state from nearly all possible starting conditions (limit points).

- **Class 2**: the CA creates patterns that repeat periodically or are stable (limit cycles) – probably equivalent to a regular grammar/finite state automaton.

- **Class 3**: from nearly all starting conditions, the CA leads to aperiodic-chaotic patterns, where the statistical properties of these patterns are almost identical (after a sufficient period of time) to the starting patterns (self-similar fractal curves) – computes ‘irregular problems’.

- **Class 4**: after a finite number of steps, the CA usually dies, but there are a few stable (periodic) patterns possible (e.g. Game of Life) - Class 4 CA are believed to be capable of universal computation.
John Conway’s Game of Life

- 2D cellular automata system.
- Each cell has 8 neighbors - 4 adjacent orthogonally, 4 adjacent diagonally.
- This is the Moore Neighborhood.
John Conway’s Game of Life

- A live cell with 2 or 3 live neighbors survives to the next round.
- A live cell with 4 or more neighbors dies of overpopulation.
- A live cell with 1 or 0 neighbors dies of isolation.
- An empty cell with exactly 3 neighbors becomes a live cell in the next round.
Is it alive?

• [http://www.bitstorm.org/gameoflife/](http://www.bitstorm.org/gameoflife/)

• Compare it to the definitions...
Cellular Automata

• Used in computer graphics:
  – Cellular Texturing
Urban Procedural Modeling

- **Cities**
- **Buildings**
- **CityEngine**
  - [CityEngine](https://www.youtube.com/watch?v=xJCIIE9pulk)
  - (for Unreal: [https://www.youtube.com/watch?v=faOdiVcxRG4](https://www.youtube.com/watch?v=faOdiVcxRG4))
Ex. Split Grammars

- Non-terminals = \{B, I\}
- Terminals = \{G, T, F\}
- Start symbol = B

\[
B & \rightarrow GIT \\
I & \rightarrow IF \\
I & \rightarrow F
\]

\[
h \geq 3 \quad B_{xyz}^{wdh} & \rightarrow G_{x,y,z}^{w,d,1} \quad I_{x,y,z}^{w,d,h-2} \quad T_{x,y,z}^{w,d,1} \\
h \geq 2 \quad I_{xyz}^{wdh} & \rightarrow F_{x,y,z}^{w,d,1} \quad I_{x,y,z}^{w,d,h-1} \\
I_{xyz}^{wdh} & \rightarrow F_{x,y,z}^{w,d,1}
\]

Figure 3: Simple buildings generated by a simple grammar.

Figure 4: Four box buildings.

Figure 5: The Dana Porter library

Ex. Shape Grammar

Figure 14: An example showing spatial subdivisions of functional areas.

Figure 15: Some feasible layouts for a single-bed dorm. The shape grammar is shown in Example 1.

Figure 16: An infeasible layout for a single-bed dorm. The shape grammar is shown in Example 1.

Figure 17: Some feasible layouts for a two-bed dorm. The shape grammar is shown in Example 2.

Figure 18: An infeasible layout for a two-bed dorm. The shape grammar is shown in Example 2.

Legend
- desk
- closet
- bed
- chair

Start symbol = bedroom
Non-terminals = {sleep-area, sleep-area-1, sleep-area-2, study-area, study-area-1, study-area-2}
Terminals = {BED-1, BED-2, DESK-1, DESK-2, CHAIR-1, CHAIR-2, CLOSET-1, CLOSET-2}

bedroom → split \{X, 0.5, 0.5\} sleep-area | study-area
bedroom → split \{Y, 0.5, 0.5\} sleep-area | study-area
sleep-area → split \{X, 0.5, 0.5\} sleep-area-1 | sleep-area-2
sleep-area → split \{Y, 0.7, 0.3\} BED-1 | CLOSET-1
sleep-area → split \{Y, 0.7, 0.3\} BED-2 | CLOSET-2
sleep-area-1 → split \{X, 0.5, 0.5\} study-area-1 | study-area-2
sleep-area-2 → split \{X, 0.5, 0.5\} study-area-1 | study-area-2
study-area → split \{X, 0.2, 0.8\} DESK-1 | CHAIR-1
study-area → split \{Y, 0.2, 0.8\} DESK-1 | CHAIR-1
study-area-1 → split \{X, 0.2, 0.8\} DESK-2 | CHAIR-2
study-area-2 → split \{Y, 0.2, 0.8\} DESK-2 | CHAIR-2

place-at-corner\{sleep-area-1, BED-1, CLOSET-1\}
place-at-corner\{sleep-area-2, BED-2, CLOSET-2\}
place-at-edge\{study-area-1, DESK-1, CHAIR-1\}
place-at-edge\{study-area-2, DESK-2, CHAIR-2\}
Videos and more

• Procedural Modeling of Cities
  – http://www.youtube.com/watch?v=khrWonALQiE

• Procedural Modeling of Buildings
  – http://www.youtube.com/watch?v=iDsSrMkW1uc

• Procedural Modeling of Structurally Sound Masonry Buildings
  – http://www.youtube.com/watch?v=zXBAthLSxSQ

• Image-based Procedural Modeling of Facades
  – http://www.youtube.com/watch?v=SncibzYy0b4

• Image-based Modeling
  – Facades: http://www.youtube.com/watch?v=amD6_i3MVZM
Videos and more

- Our Work:
  - CGVLAB Urban