Procedural Modeling of Cities

Jonathon Gu, Lily Riopelle, Cody Skinner, Alex Wheelock, James Wilcox
The Inspiration

“Procedural Modeling of Buildings”
Muller et al.

“A Survey of Procedural Techniques for City Generation”
Kelly, McCabe
Road Grammar

Our grammar for generating the roads is an L-system, so at each iteration, all possible production rules are applied simultaneously.

For example, given the production rule,

\[ p_1: A \rightarrow a \]

the word AA expands to aa, and not aA.

All production rules for the road grammar are of the form:

\[ p_n: A(x) > t : x == 0 \{ /* code */ \} \rightarrow A(x-1)t \]

A tree-like structure generated by an L-system
Motivation for Arbitrary Code

The nine production rules never change.

Instead, the `globalGoals` and `localConstraints` functions, along with their inputs, completely determine road style.

\[
\begin{align*}
\omega: & \quad R(0, \text{initRuleAttr}) \ ?I(\text{initRoadAttr}, \text{UNASSIGNED}) \\
p1: & \quad R(\text{del}, \text{ruleAttr}) : \text{del}<0 \rightarrow \epsilon \\
p2: & \quad R(\text{del}, \text{ruleAttr}) > ?I(\text{roadAttr}, \text{state}) : \text{state}==\text{SUCCEED} \\
& \quad \{\text{globalGoals}(\text{ruleAttr}, \text{roadAttr}) \text{ creates the parameters for: } \text{pDel}[0-2], \text{pRuleAttr}[0-2], \text{pRoadAttr}[0-2]\} \\
& \quad \rightarrow +(\text{roadAttr}.\text{angle})F(\text{roadAttr}.\text{length}) \\
& \quad \text{B}(\text{pDel}[1], \text{pRuleAttr}[1], \text{pRoadAttr}[1]), \\
& \quad \text{B}(\text{pDel}[2], \text{pRuleAttr}[2], \text{pRoadAttr}[2]), \\
& \quad \text{R}(\text{pDel}[0], \text{pRuleAttr}[0]) ?I(\text{pRoadAttr}[0], \text{UNASSIGNED}) \\
p3: & \quad R(\text{del}, \text{ruleAttr}) > ?I(\text{roadAttr}, \text{state}) : \text{state}==\text{FAILED} \rightarrow \epsilon \\
p4: & \quad \text{B}(\text{del}, \text{ruleAttr}, \text{roadAttr}) : \text{del}>0 \rightarrow \text{B}(\text{del}-1, \text{ruleAttr}, \text{roadAttr}) \\
p5: & \quad \text{B}(\text{del}, \text{ruleAttr}, \text{roadAttr}) : \text{del}=0 \\
& \quad \rightarrow [R(\text{del}, \text{ruleAttr}) ?I(\text{roadAttr}, \text{UNASSIGNED})] \\
p6: & \quad \text{B}(\text{del}, \text{ruleAttr}, \text{roadAttr}) : \text{del}<0 \rightarrow \epsilon \\
p7: & \quad R(\text{del}, \text{ruleAttr}) < ?I(\text{roadAttr}, \text{state}) : \text{del}<0 \rightarrow \epsilon \\
p8: & \quad ?I(\text{roadAttr}, \text{state}) : \text{state}==\text{UNASSIGNED} \\
& \quad \{\text{localConstraints}(\text{roadAttr}) \text{ adjusts the parameters for: } \text{state}, \text{roadAttr}\} \\
& \quad \rightarrow ?I(\text{roadAttr}, \text{state}) \\
p9: & \quad ?I(\text{roadAttr}, \text{state}) : \text{state}!==\text{UNASSIGNED} \rightarrow \epsilon
\end{align*}
\]
Motivation for Arbitrary Code

The nine production rules never change.

Instead, the globalGoals and localConstraints functions, along with their inputs, completely determine road style.

- $\text{globalGoals}(\text{ruleAttr}, \text{roadAttr})$ creates the parameters

- $\text{localConstraints}(\text{roadAttr})$ adjusts the parameters
Global Goals

Attempts to lay down roads according to global goals, including
- Road style
- Population density
Local Constraints

- Check Road Legality:
  - Prevent Roads from crossing water, entering parks
  - Allow specific behaviors given certain constraints, e.g. highways -> bridges

- Modify endpoints of roads based on context
  - Other roads/intersections in vicinity

Muller et al. "Procedural Generation of Cities"
Block Discovery

Given an arbitrary list of roads segments, how does one find the city blocks defined by those segments?

Construct an undirected graph such that:
- Each road segment corresponds to two, disconnected edges representing the left and right sides of the road
- Two edges share a node if the sides of the road segments they represent intersect

Given such a graph, all connected components that form a cycle represent a city block.
Plot Layout

Given a set of blocks, how does one lay out realistic building plots that don't intersect with each other or roads?

- Recursively subdivide longest nearly-parallel roads
- Throw out any resulting plots that are too small or that have no road access

Muller et al. "Procedural Generation of Cities"
Challenges

The strange structure of the grammar required 3,000 lines of code and 40 man hours to implement.

The grammar alone doesn't give much without good globalGoals and localConstraints, both of which are finicky and difficult to implement well.

The grammar does not allow for intermediate results, and roads could not be constructed at all until its completion.
Lessons

Simplify! (and then extend)

Source: http://www.shamusyoung.com/
Building Grammar

Shape grammars provide an expandable model for the stochastic generation of realistic building exteriors.

Grammar consists of productions:

\[ A : h > 9 \rightarrow \text{Scale}(3, 5, 3) \text{ Subdiv} \ (\ 'X', \ 3, \ R) \ \{B, \ C\} : 0.45 \]

Each production in the shape grammar is constructed using:
- Non-terminals (ID's)
- Conditions
- Commands
- Probabilities
Basic Commands

Each of the commands performed in a production has an execute() method that performs the necessary matrix multiplication to manipulate the scope (or reference frame) of the current shape:

- Push / Pop
- Scale
- Translate
- Rotate
- Insert

\[
A \Rightarrow [ \text{Scale}(3, 5, 3) \text{ Translate}(0, 0, 6) \text{ Insert("cube.off") } ]
\]
Subdivide Command

**Subdivide:** Given an axis and numerical parameters, subdivide splits a scope into smaller scopes along the specified axis and at intervals corresponding to the argument values.

**Facade** ➔ \textit{Subdiv ( 'Y', 3.5, 0.3, 3, 3, 3) \{floor, ledge, floor, floor, floor\}}

**Floor** ➔ \textit{Subdiv ( 'X', 2, R, R, 2) \{B, A, A, B\}}
Our Grammar interpreter performs the following algorithm for each expansion step:

1. Select the first active node $N$ in our derivation tree
2. Select the production $p$ for node $N$
3. In correct order, execute the commands for production $p$
   a. Add children to $N$ accordingly
4. Set $N$ to inactive

$N \rightarrow \text{Scale}(3, 5, 3) \text{ Add } \{A\} \quad \text{Translate}(0, 0, 6) \text{ Add } \{B\}$
Ambient Occlusion

A relatively efficient rendering algorithm that produces shadows that are not uniformly dark and therefore are more realistic.