Modeling

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

Alternatives

- Data-driven modeling
  - Scan model geometry from real world examples
  - Use laser scanners or similar devices
  - Photographs as textures
- Procedural modeling
  - Construct 3D models and textures using algorithms

Procedural modeling

- Wide variety of techniques for algorithmic model creation
- Used to create models that are too complex (or tedious) for a person to build
  - Trees, landscapes (mountains), clouds, etc.
  - 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 (procedure)
  - Often including randomness to add variety
  - E.g., a single tree pattern can be used to model an entire forest

Procedural modeling

- Constructive solid geometry (CSG)
- Randomness & recursion
- Fractal height fields
- L-systems
- Noise textures

Constructive solid geometry

- Start with simple primitives
  - Cubes, cylinders, spheres, cones, pyramids
  - Solids, interpret as sets of points in 3D
- Combine them using Boolean operations on sets
  - Subtraction, union, intersection
Constructive solid geometry

- Can construct complicated objects
- Implemented in CAD tools, but also game engines

Procedural modeling

- Constructive solid geometry (CSG)
  - Randomness & recursion
  - Fractal height fields
  - L-systems
  - Noise textures

Randomness

- Use some sort of randomness to make models more interesting, natural, less uniform, clean
- 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 tree is built using several random numbers, then the entire tree can be rebuilt by just resetting the seed to its initial value
  - If the seed is set to a different value, 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 objects that are self-similar, fractals
  - Object that look the same when viewed at different scales
- For example, the shape of a coastline may appear as a jagged line when we view a map of California
  - As we zoom in closer and closer, 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

Recursion

- Regular repeated patterns, or irregular with randomness
- Random fractals are useful for creating a wide variety of natural shapes
  - Mountain landscapes
  - Trees, self similar branching structures
- For procedural modeling, we may borrow some fractal concepts, but we rarely deal with true mathematical fractals with infinite detail
- Think of fractals as techniques for generating randomness in some limited range of scales

Procedural modeling

- Constructive solid geometry (CSG)
  - Randomness & recursion
  - Fractal height fields
  - L-systems
  - Noise textures
**Height fields**
- Landscapes are often constructed as height fields
- Regular grid in the ground plane (assume xz plane here)
- Store a height (y) value at each point
- Can store large terrain in memory
  - No need to store xz coordinates, connectivity
- Shaped terrain by operations that modify the y coordinates
- Can interpret height values as gray scale values
  - Apply image processing tools

**Fractal landscapes**
- Random midpoint displacement algorithm
  - Recursively subdivide triangles
  - Randomly displace edge midpoints
  - Reduce size of displacement as triangles get smaller
  - Similar for quadrilaterals

**Procedural modeling**
- Constructive solid geometry (CSG)
- Randomness & recursion
- Fractal height fields
- L-systems
- Noise textures

**L-systems**
- Developed by a biologist (Lindenmayer) in 1968 to study growth patterns of algae
- Defined by a 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
- Difference between L-systems and formal languages
  - In each step, apply all possible production rules
  - L-systems are subsets of formal languages

**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)

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Sierpinski triangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td><img src="http://www.planetside.co.uk/gallery/f/tg09" alt="2 iterations" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="http://www.planetside.co.uk/gallery/f/tg09" alt="4 iterations" /></td>
</tr>
<tr>
<td>6</td>
<td><img src="http://www.planetside.co.uk/gallery/f/tg09" alt="6 iterations" /></td>
</tr>
<tr>
<td>9</td>
<td><img src="http://www.planetside.co.uk/gallery/f/tg09" alt="9 iterations" /></td>
</tr>
</tbody>
</table>

**Fractal landscapes**
- Add textures, nice rendering...
- Terragen, free software
  [http://www.planetside.co.uk/terragen/](http://www.planetside.co.uk/terragen/)
Fractal fern

- Variables: X, F
  - X: no drawing operation
  - F: move forward
- Constants: +, −
  - Turn left, right
- Start: X
- Rules:
  \[(X \rightarrow F - ([X] + X) + F[+FX] - X), (F \rightarrow FF)\]
- Stochastic L-system
  - If there is more than one production rule for a symbol, randomly choose one

Fractal trees

- Recursive generation of trees in 3D
  [http://web.comhem.se/solgrop/btree.htm](http://web.comhem.se/solgrop/btree.htm)
- Model trunk, branches as cylinders
- Change color from brown to green at certain level of recursion

Algorithmic beauty of plants

- Online book on algorithmic beauty of plants by Prusinkiewicz
  [http://algorithmicbotany.org/papers/#abop](http://algorithmicbotany.org/papers/#abop)


Buildings, cities

Pascal Mueller

Procedural modeling

- Constructive solid geometry (CSG)
- Randomness & recursion
- Fractal height fields
- L-systems
- Noise textures

Noise

- Form of randomness to model natural, organic looking 2D and 3D structures
- Noise represents a distribution of randomness over some space (2D or 3D)
- Not entirely random, as two points nearby will have a similar value
- Noise has a frequency associated with it
- By combining noise patterns of different frequencies, one can make more complex turbulence patterns

2D noise  Turbulence
Noise
- Often used as a 3D solid texture
  - Texture is defined in 3D
  - Surface visualizes a 2D slice through 3D volume

Perlin noise
- Efficient way to generate noise textures procedurally
- Band limited
  - Features have a certain uniform size
- Translation invariant
  - “Looks the same everywhere”
- Rotation invariant
  - No preferred orientation

Perlin noise in 3D
Goal: create random 3D pattern with uniform feature size
- Generate randomly oriented unit vectors (gradients) on a 3D grid
- Use Hermite interpolation to noise value
- Leads to band limited interpolant

Perlin noise in 1D
- Blending function
  \[ s(x) = 6x^5 - 15x^4 + 10x^3 \]
- Continuous derivatives at boundaries
  \[ s'(0) = 0, s'(1) = 0, s''(0) = 0, s''(1) = 0 \]
### 3D perlin noise

- Noise [Ken Perlin]
- Noise bump mapped [Ken Perlin]

### Fractal noise and turbulence

- Fractal noise
  \[ F(x) = \sum i a_i \cdot N(x \cdot f_i) \]
- Turbulence
  \[ T(x) = \sum i \|a_i \cdot N(x \cdot f_i)\| \]
- Frequencies usually in octaves \( f_i = 2f_{i-1} \)
- Amplitude \( a_i = p \cdot a_{i-1} \)
- Persistency \( p < 1 \), often \( p = 0.5 \)

### Fractal noise in 1D

- \( N(x) \)
- \( 0.5N(2x) \)
- \( 0.25N(4x) \)
- \( 0.125N(8x) \)

### Fractal noise in 1D

- \( N(x) + 0.5N(2x) + 0.25N(4x) + 0.125N(8x) \)

### 3D Perlin noise

- Noise
- Turbulence

### Variations on noise

- Noise
- \( \sin(x + \text{sum 1/f(noise)}) \)

- sum 1/f(noise)
- sum 1/f(noise)
### Other effects

- **Use colormap and look-up with noise offset**
  - Wood
    - color = colormap(radius+noise)
- **Marble**
  - Parallel plane + noise
  - color = colormap(x+turbulence)

### Corona

- **Recipe**
  - Create a smooth gradient function that drops off radially from bright yellow to dark red
  - Phase shift this function by adding a turbulence texture to its domain
  - Place a black cutout disk over the image

### Corona

- **Animation**
  - Scale up over time
  - Use 3D noise for time

### More on Perlin noise

- Easy to read intros
  - [http://freespace.virgin.net/hugo.elias/models/m_perlin.htm](http://freespace.virgin.net/hugo.elias/models/m_perlin.htm)
- Technical paper on 3D implementation
  - [http://mrl.nyu.edu/~perlin/noise/](http://mrl.nyu.edu/~perlin/noise/)

### Hardware implementation

- Noise evaluation using programmable shaders

### Cellular noise

- Place random points in a grid
- Use distance to neighbors as a noise function
  \[ F_n(x) = \text{distance to n-th neighbor} \]
- Fast implementation:
  “A cellular texture basis function”, Steven Worley
Fractal cellular noise

$$F_n^s(x) = \sum_i a_i F_n(x \cdot f_i)$$

Summary procedural modeling

- Many different forms of procedural modeling
- Advantages
  - Reduces manual work
  - Produces highly detailed models
  - Decreases memory requirements
- Disadvantages
  - Sometimes tricky to control, requires experience
  - Some effects hard to model
- Often combination with other techniques
  - E.g., city modeling: use L-system to define structures, hand-painted textures for visual realism