Chapter 7

1. Introduction
2. The Computer Graphics Pipeline
3. Object Representation
4. Object Transformation
5. 3D – Projections, Camera, and Lighting
6. Scene Representation and Interaction
7. **Advanced Texturing and Shading**
Level-of-Detail Methods

• With perspective projection, objects that are farther away from the camera map to a smaller region on screen
Level-of-Detail Methods

- **Level-of-Detail (LOD)** techniques improve rendering performance by reducing details of objects that are far away from the camera.

![Model in full resolution vs simplified model](image)
Level-of-Detail Methods

- **Discrete LOD** techniques specify regions where different levels of detail are to be used.

  - Range 1: Full resolution
  - Range 2: Medium resolution
  - Range 3: ...

- Popping artifacts can be compensated by alpha blending or morphing.
Level-of-Detail Methods

- **Continuous LOD** techniques generate appropriate details based on distance
  - If the object can be represented as a function, we can create vertices on-demand
  - No need to create multiple resolutions in a pre-processing step
Mesh Simplification

- **Mesh simplification** techniques define how to reduce the geometric complexity of a mesh

edge collapse
Mesh Simplification

- Mesh simplification operations need to be applied carefully
- Goal is to reduce number of vertices while staying close to original

identical number of primitives
Mesh Simplification

- **Hausdorff distance** can be used to compute quality of simplified mesh

\[
d_H(A, B) = \max_{p \in A} \min_{q \in B} \| p - q \|
\]

one-sided Hausdorff distance

\[
d_H(A, B) = \max\{d_H(A, B), d_H(B, A)\}
\]

two-sided

\[
d_H(B, A)
\]

\[
d_H(A, B)
\]
Mesh Subdivision

- **Subdivision** creates additional polygons or details

**Loop** subdivision
divide triangle into four sub-triangles, weigh positions for smoothing

CC-SA Simon Fuhrmann
Mesh Subdivision

• **Loop Subdivision**
  1. Subdivide mesh
  2. Weigh new and old vertices based on neighbor positions

![Diagram of Loop Subdivision](image-url)
Mesh Subdivision

• **Loop Subdivision**
• Special rules can be applied locally to retain sharp edges

CC-SA Philip Rideout
Procedural Methods

• Realistic graphics need high levels of detail
• Modeling geometry explicitly creates large amounts of primitives

• **Procedural methods** describe objects in an algorithmic manner
  • Algorithms/Functions can be evaluated in run-time
  • Geometric representation is only generated when needed
Procedural Methods

- Typical application scenarios:
  - Clouds, terrain, trees

Jason Seward
Language-Based Models

- Language-based symbols or **tree grammars** use replacement rules (**productions**) to create hierarchical objects

Rules

\[
\begin{align*}
A & \rightarrow AB \\
B & \rightarrow A
\end{align*}
\]

A

AB

ABA

ABAAB
Language-Based Models
Language-Based Models

- **Koch curve** and **Koch snowflake**
  - Actions: Forward, Left turn (60), Right turn (60)

Rules:

\[ F \rightarrow F L F R R F L F \]

- [Koch curve](#)
- [Koch snowflake](#)
Language-Based Models

• Adding push and pop operations (also now: 27 degrees)
  • Self-similarity (c.f., fractals)

Rules

\[ F \rightarrow F[RF]F[LF]F \]

Rules can be applied randomly to increase variance
Procedural Noise

- Adding ‘randomness’ to models can increase visual detail and perceived realism, and make them more interesting
  - **White noise** is not optimal, because samples are completely uncorrelated
  - We want to be able to generate the same randomness every time we generate the object

- For computer graphics ‘controllable’ noise is important

Coherent noise

Incoherent noise
Perlin Noise

- **Perlin noise** is widely used in computer graphics
  - Repeatable noise
  - Coherent noise - smooth (no discontinuities)
  - Uniform feature sizes
  - Form of gradient noise (noise values are generated in-between)

random gradients

noise value created as weighted sum of four linear functions
Perlin Noise

Basic Perlin noise

Sum of two Perlin noise functions with different scales

\[ \text{Noise}(p) = \sum_{i}^{n} \frac{1}{i} \text{Noise}(i \cdot p) \]