Mesh Simplification

Simplifier

University of British Columbia

Motivation

- Reduce information content
- Accelerate rendering
- Multi-resolution models

University of British Columbia
Level of Detail (LOD)

- Refined mesh for close objects
- Simplified mesh for far

Progressive Meshes

- Single operation
  - Typically focus on quality

Continuous - Progressive mesh
- Focus on speed
- Requires preprocessing
- Time/space/quality tradeoff
Quality (e.g. Creases Preserving)

Methodology

- Sequence of local operations
  - Involve near neighbors - only small *patch* affected in each operation
  - Each operation introduces error
  - Find and apply operation which introduces the least error
Simplification Operations (1)

- Decimation
  - Vertex removal:
    - $v \leftarrow v-1$
    - $f \leftarrow f-2$

  Remaining vertices - subset of original vertex set

Simplification Operations (2)

- Decimation
  - Edge collapse
    - $v \leftarrow v-1$
    - $f \leftarrow f-2$

  Vertices may move
Simplification Operations (3)

- Contraction
  - Pair contraction

- Vertices may move

Error Control

- Local error: Compare new patch with previous iteration
  - Fast
  - Accumulates error
  - Memory-less

- Global error: Compare new patch with original mesh
  - Slow
  - Better quality control
  - Can be used as termination condition
  - Must remember the original mesh throughout the algorithm
Local vs. Global Error

2000 faces  488 faces  488 faces

Simplification Error Metrics

- Measures
  - Distance to plane
  - Curvature
  - Usually approximated
    - Average plane
    - Discrete curvature

\[ \Sigma \alpha / 2\pi \]
The Basic Algorithm

- Repeat
  - Select the element with minimal error
  - Perform simplification operation (remove/contract)
  - Update error (local/global)

- Until mesh size / quality is achieved

Implementation Details

- Vertices/Edges/Faces data structure
  - Easy access from each element to neighboring elements
  - Use priority queue (e.g. heap)
    - Fast access to element with minimal error
    - Fast update
**Vertex Removal Algorithm**

- **Simplification operation:** Vertex removal
- **Error metric:** Distance to average plane
- **May preserve mesh features (creases)**

**Algorithm Outline**

- Characterize local topology/geometry
- Classify vertices as removable or not
- Repeat
  - Remove vertex
  - Triangulate resulting hole
  - Update error of affected vertices
- Until reduction goal is met
Characterizing Local Topology/Geometry

Decimation Criterion

- $E_{MAX}$ - user defined parameter
- Simple vertex:
  - Distance of vertex to the face loop average plane $< E_{MAX}$
- Boundary vertices:
  - Distance of the vertex to the new boundary edge $< E_{MAX}$
**Triangulating the Hole**

- Vertex removal produces non-planar loop
  - Split loop recursively
  - Split plane orthogonal to the average plane
- Control aspect ratio
- Triangulation may fail
  - Vertex is not removed

**Example**

*Simplifier*
Pros and Cons

- **Pros:**
  - Efficient
  - Simple to implement and use
    - Few input parameters to control quality
  - Reasonable approximation
  - Works on very large meshes
  - Preserves topology
  - Vertices are a subset of the original mesh
- **Cons:**
  - Error is not bounded
    - Local error evaluation causes error to accumulate

Edge Collapse Algorithm

- **Simplification operation:**
  - Edge collapse (pair contraction)
- **Error metric:**
  - Distance, pseudo-global
Distance Metric: Quadrics

- Choose point closest to set of planes (triangles)

- Sum of squared distances to set of planes is quadratic ⇒ has a minimum

Quadrics

- Plane
  - \(Ax + By + Cz + D = 0\), where \(A^2 + B^2 + C^2 = 1\)
  - \(p = [A, B, C, D]\), \(v = [x, y, z, 1]\), \(v^T p = 0\)

- Quadratic distance between \(v\) and \(p\):
  \[
  \Delta_p(v) = (v^T p)^2 = (v^T p) (v^T v) = v^T (p^T p) v
  \]
  \[
  K_p = \begin{bmatrix}
  A^2 & AB & AC & AD \\
  AB & B^2 & BC & BD \\
  AC & BC & C^2 & CD \\
  AD & BD & CD & D^2
  \end{bmatrix}
  \]
Distance to Set of Planes

\[ \Delta(v) = \sum_{p \in \text{planes}(v)} \Delta_p(v) \]
\[ = \sum_{p \in \text{planes}(v)} (v K_p v^T) \]
\[ = v (\sum_{p \in \text{planes}(v)} K_p) v^T \]
\[ = v Q_v v^T \]

After \(v_1, v_2\) are contracted to \(v\),
\[ Q_v \leftarrow Q_{v_1} + Q_{v_2} \]

Pseudo-global

All original planes persist during the entire simplification process

Contracting Two Vertices

- Goal: Given edge \(e = (v_1, v_2)\), find contracted \(v = (x, y, z, 1)\) that minimizes \(\Delta(v)\):
  \[ \frac{\partial \Delta}{\partial x} = \frac{\partial \Delta}{\partial y} = \frac{\partial \Delta}{\partial z} = 0 \]
- Solve system of linear normal equations:
  \[
  \begin{bmatrix}
    q_{11} & q_{12} & q_{13} & q_{14} \\
    q_{21} & q_{22} & q_{23} & q_{24} \\
    q_{31} & q_{32} & q_{33} & q_{34} \\
    0 & 0 & 0 & 1
  \end{bmatrix}
  \begin{bmatrix}
    v_1 \\
    v_2 \\
    v_3 \\
    v_4
  \end{bmatrix}
  =
  \begin{bmatrix}
    0 \\
    0 \\
    0 \\
    1
  \end{bmatrix}
  \]

- If no solution - select the edge midpoint
Selecting Valid Pairs for Contraction

- **Edges:**
  \[ \{(v_1, v_2) : (v_1v_2) \text{ is in the mesh} \} \]

- **Close vertices:**
  \[ \{(v_1, v_2) : ||v_1 - v_2|| < T \} \]
  
  - Threshold T is input parameter

Algorithm

- Compute \( Q_v \) for all the mesh vertices
- Identify all valid pairs
- Compute for each valid pair \((v_1, v_2)\) the contracted vertex \( v \) and its error \( \Delta(v) \)
- Store all valid pairs in a priority queue (according to \( \Delta(v) \))
- While reduction goal not met
  - Contract edge \((v_1, v_2)\) with the smallest error to \( v \)
  - Update the priority queue with new valid pairs
Examples

Dolphin (Flipper)

Original - 12,337 faces

2,000 faces

300 faces (142 vertices)

Examples

Budha

Simplifier

Original - 12,000

2,000 faces

298 faces (140 vertices)
Pros and Cons

Pros
- Error is bounded
- Allows topology simplification
- High quality result
- Quite efficient

Cons
- Difficulties along boundaries
- Difficulties with coplanar planes
- Introduces new vertices not present in the original mesh