Rendering Large Models

(in real time)

- A large model may contain billions of polygons and take up gigabytes of memory
- Such models arise naturally in:
  - Architectural walkthroughs
  - CAD (aircraft, ships, oil platforms…)
  - Training simulators and games
- Generally, users wish to move about the model, looking around, and see in real-time a stream of images from the current viewpoint
  - Real-time means at least 10 frames-per-second, but 20 frames-per-second is much, much better

Tools for Large Models

- Visibility Culling: Avoid touching data for parts of the model the viewer cannot see
- Model Simplification (LOD): Reduce the complexity of the model with minimal impact on visual quality
- Image-Based Representations: Replace geometry with images
- Database Management: Ensure that data required for the current frame is in memory, even if the entire model won't fit

Simplification Overview

- Run-time impact of simplification:
  - Reduces the cost of transforming the model
  - Reduces memory bandwidth to graphics subsystem
  - Little effect on rasterization (same pixels touched)
- Useful properties of simplification algorithm:
  - Automatic - models are too big to examine by hand
  - Few restrictions on input data (eg allow non-manifold meshes)
  - Major reduction in model complexity
  - Minimal increase in visual error
  - Fast – both preprocessing and run-time

Aspects of Simplification

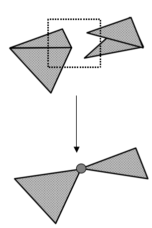
- Simplification operations: How are the geometry and surface properties modified?
- Error Measures: How is error measured and controlled by the algorithm?
- Are the simplified models static or dynamic?
  - Static: One of a few pre-computed models is chosen to be rendered
  - Dynamic: The simplification is done on the fly
- Run-time Control: How is it decided which model to render?

Simplification Operations

- Various basic operations are possible:
  - Vertex cluster
  - Vertex remove
  - Edge collapse
  - Vertex pair
  - Many minor variations
- Each operation reduces the complexity by a small amount and influences a small region
- Apply many operations in sequence to achieve large reductions

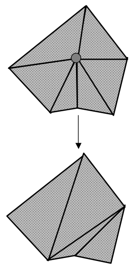
Vertex Cluster

- Merge a set of vertices based on geometric proximity
- Triangles may disappear, become lines or become points
  - Can choose to remove or render degeneracies
- Very general, and doesn't rely on manifold meshes
- Can have a major impact on topology
  - Hard to make it look good



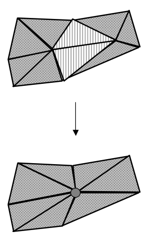
Vertex Remove

- Remove a vertex and adjacent faces
- Re-triangulate to fill hole
  - 2 fewer triangles
  - Exponential number of re-triangulations
- Requires a manifold surface about the vertex
- Preserves the local topological structure
  - Generally looks better



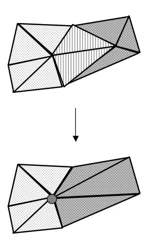
Edge Collapse

- Merge two edge vertices into one, thus removing the edge
  - Choose position and attributes for the new vertex
- Delete degenerate triangles
  - Reduction of 2 for manifold edge
- The removal can be animated for a smooth transition
  - Move vertices toward new location, then delete them

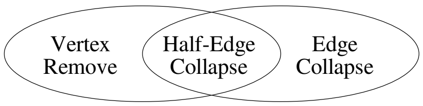


Half-Edge Collapse

- Collapse the edge to one of its endpoint vertices
- Vertex sets remain a subset of the original set – property of vertex removal
- Smooth transitions still possible – property of edge collapse

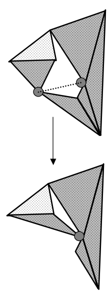


Vertex Remove vs. Edge Collapse



- Neighborhood: About 1 vertex vs. about 2 vertices
- New vertices: None vs. one in free position
- New tesselation: Many possible vs. one
- Smooth transitions: Difficult vs. natural

Vertex Pair



- Merge any two vertices
  - Choice based on geometry, topology, attributes, etc
- More flexible than edge collapse
  - Can change topology quickly
- More local control than vertex cluster
  - Choose vertices based on more than just proximity

Choosing an Operation

- Attention to topology improves appearance
- Supporting non-manifolds increases robustness and domain
- Collapse-type operations support smooth transitions
- Vertex removal affects a smaller portion of the mesh than edge collapse
- A subset of vertex removal is equivalent to a subset of edge collapse

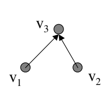
Simplification Algorithms

- Rank the possible operations according to the error they will introduce
  - Must be able to measure error
  - Error metrics make all the difference
- Repeatedly:
  - Do the operation that introduces the least error
  - Re-evaluate error in neighborhood of operation
- Almost all good simplification algorithms fit into this framework
  - Note greedy approach does not in general ensure optimal result

Error Metrics

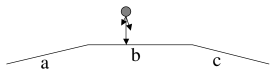
- Always include a geometric component:
  - Vertex-Vertex distance
  - Vertex-Plane Distance
  - Point-Surface Distance
  - Surface-Surface Distance
- May include attribute metrics
  - Aim to preserve pixel colors
  - Components for normal vectors, texture coordinates…

Vertex-Vertex Distance



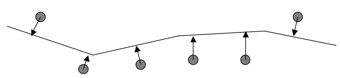
- $E=\max(\|v_3-v_1\|, \|v_3-v_2\|)$
- Works during topological changes
  - Vertex clustering, Vertex pair
  - Rossignac and Borrel 93, Luebke and Erikson 97
- A loose metric for collapse type operations
  - Vertices don’t move very far, but surface deviation may be high

Vertex-Plane Distance



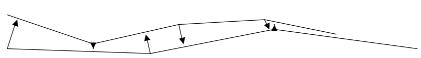
- Store a set of planes with each vertex
  - Error based on distance from the vertex to the planes
  - Merge the plane sets when vertices are merged
  - Tries to keep vertices near original surface
- Ronfard and Rossignac 96
  - Store planes, use max distance
- *Error Quadrics* – Garland and Heckbert 96
  - Quadratic form instead of planes, use sum of square distances

Point-Surface Distance



- Measure the distance from a set of points to the simplified surface
  - Point set representative of original surface
  - Use sum of squares distances
- Hoppe 93 and 96
- Approximation to surface-surface distance
- Expensive to compute

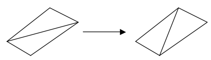
Surface-Surface Distance



- Bounds the maximum distance between the input and simplified surfaces
  - Tolerance volumes – Gueziec 96
  - Simplification Envelopes – Coehn/Varshey 96
  - Hausdorf Distance – Klein 96
  - Mapping Distance – Bajaj/Schikore 96, Cohen et. al. 97
- Arguably best measure

Error Metric Notes

- A good metric allows you to transform error in object space into error in screen space
  - Simplifies decision of which model to display
- Note that the metric are very different
  - Consider an edge-swap:
  - E=0 at verts and edges
  - E≠0 everywhere else



- Run-time metrics may take view into account