Geometric Modeling in Graphics

Part 3: Mesh simplification

Martin Samuelčík
www.sccg.sk/~sambelcik
samuelcik@sccg.sk
Mesh simplification

- Reducing number of vertices, edges, polygons
- Mesh decimation, mesh reduction, ...
- Creating levels of detail (LOD) meshes, several versions of same mesh with different number of polygons
- Lots of algorithms, lots of similar approaches
- Comparing versions using distance (Hausdorff, …)

Geometric Modeling in Graphics
Mesh simplification

- Preserving topology (number of holes) vs reducing topology

**Static**
- Creating several levels of detail in preprocess stage
- Almost no processing on the fly
- Visualization-ready preparation of levels

**Dynamic**
- Level of details is created on the fly
- Encoding continuous spectrum of details
- Progressive transmission

**View-dependent**
- Dynamic selection of LOD based on view criteria
Simplification algorithms

- **Sampling**
  - Sample mesh surface with points or voxels
  - Use smoothing on sampled points
  - Triangulate processed sampled points
  - For smooth objects

- **Adaptive subdivision**
  - Find base mesh as simplest level of detail
  - Levels are created from base mesh using subdivision
  - For models where base mesh is easy to find (terrain, …)
Simplification algorithms

- **Decimation**
  - Iteratively removing vertices or faces
  - Retriangulating hole after each step
  - Which vertex, face to remove at each step?
  - Usually simple and topology preserving

- **Vertex merging**
  - Iteratively collapsing two or more vertices into one
  - Which vertices to merge at each step?
  - What is new position of merged vertex?
  - Edge-collapse – merging two connected vertices
  - Can modify topology
Triangle mesh decimation

- Schroeder, Zarge, Lorenson: Decimation of Triangle Meshes
- Deleting chosen vertex at each step of decimation and triangulating resulting hole
- 1. Characterizing local topology for each vertex
  - Feature edges defined by feature threshold angle

---

Geometric Modeling in Graphics
Triangle mesh decimation

2. Evaluating decimation criteria
   - Simple vertex $v$ – distance to average plane $d$, $P_i$ is area of triangle
     \[ N = \frac{\sum_m n_i P_i}{\sum_m P_i}, \quad n = \frac{N}{|N|}, \quad x = \frac{\sum_m x_i P_i}{\sum_m P_i} \quad d = |n(v - x)|. \]
   - Boundary and interior edge vertex – distance to line created by other two boundary vertices
   - Corner or complex vertex – usually not removed

3. Pick vertex with lowest criteria and remove it together with incident triangles
   - Using priority queue
   - Preserve feature edges
Triangle mesh decimation

4. Triangulate resulting hole
   - Non planar triangulation of vertices loop
   - Triangulate one (removed simple, boundary vertex) or two loops (removed interior edge vertex)
   - Generate non-intersecting, non-degenerated triangulation
   - If triangulation can not be performed, do not remove vertex and triangles
   - Use triangulation schemes based on recursive loop splitting

5. Finish vertex removal loop when some criterion is reached
   - Number of vertices is below threshold
   - Number of vertices is below percentage
   - Removal of any vertex will cause in non-manifold or degenerated situation
Triangle mesh decimation

Implementation


Full Resolution
(569K Gouraud shaded triangles)

75% decimated
(142K Gouraud shaded triangles)

75% decimated
(142K flat shaded triangles)

90% decimated
(57K flat shaded triangles)
Simplification envelopes

- Cohen, …: Simplification Envelopes
- http://gamma.cs.unc.edu/ENVELOPES/

- Envelope – two offset surfaces, outer envelope displaces each vertex of the original mesh along its normal by $\varepsilon$, inner envelope displaces each vertex by $-\varepsilon$
- For orientable manifold triangle meshes
- Iteratively remove triangles or vertices and retiangulate the resulting holes, keeping the simplified surface within the envelopes
- Strict preservation of topology
Vertex clustering 1

- Rossignac, Borrel: Multi-Resolution 3D Approximations for Rendering Complex Scenes
- Vertex merging algorithm over uniform grid
- [http://www.cc.gatech.edu/~jarek/papers/VertexClustering.pdf](http://www.cc.gatech.edu/~jarek/papers/VertexClustering.pdf)
- Not requiring manifold topology, not preserving topology
  1. Assign importance for each vertex, based on sum of areas of incident triangles and "curvature" of vertex
  2. Triangulate faces and put 3D uniform grid over model
  3. Merge all vertices of one cell into one with highest importance
  4. Remove all degenerated triangles
Vertex clustering 2

- Low, Tan: Model Simplification Using Vertex Clustering
- Floating-cell clustering, working at one vertex at a time
- Paper works in real time environment, using view-dependent LOD mesh creation

1. Grade each vertex, compute weight using 2 factors
   - Factor 1 - Cosine of inverse of the maximum angle between all pairs of incident edges on the vertex
   - Factor 2 - Length of the longest among all of the edges incident upon the vertex
   - Sort vertices based on weight

2. Triangulate each face
Vertex clustering 2

3. Put box with user defined size at vertex with highest weight
   - Vertex is at the center of box

4. Merge all vertices that are inside box to one vertex with highest weight
   - Remove merged vertices from list
   - Remove degenerated triangles

5. Repeat merging process for next highest weighted vertex

6. Repeat process until some threshold is reached

Worse control over number of vertices in simplified mesh

Because of sorting, time complexity is $O(n \log(n))$
Voxel-based simplification

- He, Hong, …: Voxel Based Object Simplification
- [https://www.cs.umd.edu/gvil/papers/he_voxel.pdf](https://www.cs.umd.edu/gvil/papers/he_voxel.pdf)
- Requiring well-defined, closed-mesh, manifold mesh
- Superimposing a 3D uniform grid of voxels over the polygonal geometry
- Sampling mesh by assigning each voxel a value of 0 or 1 according to whether the sample point of that voxel lies inside or outside the object
- Applying low-pass filter on voxel values – Gauss, …
- Using Marching cubes to generate polygonal mesh from filtered values in uniform grid using isovalue 0.5
- Good for meshes without very sharp vertices or edges
Voxel-based simplification
QEM simplification

- Garland, Heckbert: Surface Simplification Using Quadric Error Metrics
  - [http://cseweb.ucsd.edu/~ravir/190/2016/garland97.pdf](http://cseweb.ucsd.edu/~ravir/190/2016/garland97.pdf)
- Iterative contraction of vertex pairs, possible connection of two unconnected vertices that are close enough
- New computation of vertex-merge error
- For triangular meshes
- No requirement for manifold topology
- No topology preservation
- Probably best combination of efficiency, fidelity, and generality
QEM pair selection

- Picking valid vertex pair \((v_1,v_2)\) for contraction at initialization time
  - \(v_1,v_2\) is an edge
  - \(|v_1-v_2|<T\), \(T\) is user defined threshold
- Threshold \(T=0\) gives simple edge contracting algorithm
- Positive \(T\) gives algorithm ability to connect unconnected parts and to change genus of mesh
- With bigger \(T\), we can move to \(O(n^2)\) pairs, slowing algorithm significantly
- When pair \((v_1,v_2)\) is contracted into vertex \(v\), all candidate pairs containing \(v_1\) and \(v_2\) are updated with \(v\)

Geometric Modeling in Graphics
QEM error approximation

- Introducing cost of contraction to select one best pair to contract during a given iteration
- Associating 4x4 matrix $Q_v$ with each vertex $v$
  - Construct plane $p$: $ax+by+cz+d = 0, a^2+b^2+c^2=1$ for each triangle incident to vertex $v$
  - Compute fundamental error quadric matrix $K_p$
  - $Q_v$ is then sum of all $K_p$
- Error (cost) at vertex $v$ is $\Delta(v)=v^TQ_vv$
- After contraction $(v_1,v_2)\rightarrow v$, the new error matrix is $Q_v=Q_{v_1}+Q_{v_2}$
- After contraction $(v_1,v_2)\rightarrow v$, the position of $v$ is such that it minimizes $\Delta(v)$, e.g. $v=Q_v^{-1}0^T$
- Cost of contraction $(v_1,v_2)\rightarrow v$ is $\Delta(v)=v^TQ_vv$
QEM sum

1. Compute the $Q$ matrices for all the initial vertices.
2. Select all valid pairs.
3. Compute the optimal contraction target $v$ for each valid pair $(v_1, v_2)$. The error $v^T(Q_{v_1} + Q_{v_2})v$ of this target vertex becomes the cost of contracting that pair.
4. Place all the pairs in a heap keyed on cost with the minimum cost pair at the top.
5. Iteratively remove the pair $(v_1, v_2)$ of least cost from the heap, contract this pair, and update the costs of all valid pairs involving $v_1, v_2$. Remove also all collapsed triangles.
QEM

Implementation
http://www.cs.cmu.edu/~./garland/quadrics/qslim.html

Geometric Modeling in Graphics
Progressive meshes

- Hoppe

- Edge-collapse simplification scheme on triangular meshes
- Requiring manifold topology, preserving topology
- Introducing energy function for mesh
- Algorithm evaluates all edges that can be collapsed according to their effect on energy function and sorts them into a priority queue
- After collapse of edge with lowest energy, energy reevaluates and resorts nearby edges into the queue
Progressive meshes

- Position of new vertex vs after \((v_s, v_t)\) collapse can be \(v_s, v_t\) or \((0.5v_s + 0.5v_t)\)
- Process repeats until topological constraints prevent further simplification – base mesh
- Vertex split (vsplit) – inverse to edge collapse (ecol)
- Progressive mesh – from base mesh to any level of details using several consecutive vsplits
- Using in mesh compression, progressive transmission

![Progressive meshes example](image)

(a) Base mesh \(M^0\) (150 faces)  
(b) Mesh \(M^{2^5}\) (500 faces)  
(c) Mesh \(M^{2^{25}}\) (1,000 faces)  
(d) Original \(M = M^n\) (13,546 faces)
Progressive meshes
Quad mesh simplification

- Tarini, ...: Practical quad mesh simplification
- [Link](http://vcg.isti.cnr.it/Publications/2010/TPCPP10/)

Requiring regularity over quad mesh

- Composing all iterations of quad mesh simplification to be as regular (as homeometry) as possible
  - Edges have same length $l$
  - Diagonals of faces have same length $l\sqrt{2}$

How far is mesh $M$ from being homeometric – objective function

- $e$ spans over all edges of mesh
- $d$ spans over all edges of mesh

$$\mu = \sqrt{\frac{\text{Area}(M)}{|M|}}, \quad \sum_{e \in M^E}(|e| - \mu)^2 + \sum_{d \in M^D}(|d| - \sqrt{2}\mu)^2$$
Quad mesh simplification

0. [Convert input mesh into quad mesh $M_0$]

1. Initial global smoothing of mesh $M_0$ (minimizing function)

2. Iteratively process mesh $M_i$ to produce mesh $M_{i+1}$ until user-defined criterion is met. In each loop:
   a. for a fixed number of times:
      i. choose shortest edge or diagonal
      ii. perform any profitable local optimizing-operation, until none is available
      iii. select and perform a local coarsening-operation and cleaning operation on chosen elements such that operations minimizes objective function as best as possible
   b. local smoothing

3. Final global smoothing of mesh $M_n$
Quad mesh local operations

- Edge collapse
- Diagonal collapse
- Edge rotate
- Vertex rotate
- Singlet removal
- Doublet removal

Geometric Modeling in Graphics
Quad mesh simplification
Terrain visualization

- View above terrain surface – some parts are close, some away – using LOD (Level Of Detail), each part of terrain is rendered in some detail based on distance from camera.
- Needed structure for storing all levels of detail for each part of terrain.
- Terrain
  - Height field
Terrain visualization

- Creating complete quadtree over height field
- Traversing tree during rendering – based on distance of camera and node area, the traverse is stopped or continued
- View-dependent, subdivision based, dynamic LOD creation
Terrain visualization

- Problems with the edge between areas on different levels in quadtree
- Solution using triangulation = connection of two consecutive quadtrees
Terrain visualization
The End for today