Simplification Algorithms

- Simplification approaches:
  - Incremental methods based on local updates
    - Mesh decimation
    - Energy function optimization
    - Quadric error metrics
    - Coplanar facets merging
    - Re-tiling
    - Clustering
  - Wavelet-based

- Incremental methods based on local updates...

- Incremental methods based on local updates...

- Local update actions:
  - Vertex removal
  - Edge collapse
  - Triangle collapse

- Different approaches:
  - Mesh decimation
  - Energy function optimization
  - Quadric error metrics
Incremental methods based on local updates...

The common framework:

- **loop**
  - **select** the element to be deleted/collapsed;
  - **evaluate approximation** introduced;
  - **update** the mesh after deletion/collapse;

- until mesh **size/precision** is satisfactory;

---

Energy function optimization

**Mesh Optimization** [Hoppe et al. ’93]

- Simplification based on the iterative execution of:
  - edge collapsing
  - edge split
  - edge swap

---

Algorithm structure

- outer minimization cycle (discrete optimiz. probl.)
  - choose a legal action (edge collapse, swap, split) which reduces the energy function
  - perform the action and update the mesh ($M_i \rightarrow M_{i+1}$)

- inner minimization cycle (continuous optimiz. probl.)
  - optimize the vertex positions of $M_{i+1}$ with respect to the initial mesh $M_0$

- but (to reduce complexity)
  - legal action selection is random
  - inner minimization is solved in a fixed number of iterations

---

approximation quality evaluated with an **energy function**:

$$E(M) = E_{dist}(M) + E_{rep}(M) + E_{spring}(M)$$

which evaluates geometric **fitness** and repr. **compactness**

- $E_{dist}$: sum of squared distances of the original points from $M$
- $E_{rep}$: factor proportional to the no. of vertex in $M$
- $E_{spring}$: sum of the edge lengths
... Energy function optimization: Mesh Optimization ...

Mesh Optimization - Examples

![Examples Image by Hoppe et al.]

... Energy function optimization: Progressive Meshes ...

Progressive Meshes

- Hoppe '96

- execute **edge collapsing only** to reduce the energy function

- **edge collapsing** can be easily inverted ===>
  - store sequence of inverse vertex split transformations to support:
  - multiresolution
  - progressive transmission
  - selective refinements
  - geomorphs

- faster than MeshOptim.

... Energy function optimization: Progressive Meshes ...

- Hoppe '96

- **Preserving mesh appearance**
  - shape and crease edges
  - scalar fields discontinuities (e.g. color, normals)
  - discontinuity curves

- Managed by inserting two new components in the energy function:
  - $E_{\text{scalar}}$: measures the accuracy of scalar attributes
  - $E_{\text{disc}}$: measures the geometric accuracy of discontinuity curves

![Progressive Meshes Image by H. Hoppe]

... Energy function optimization: Mesh Optimization ...

Mesh Optimization - Evaluation

- high quality of the results
- preserves topology, re-sample vertices
- high processing times
- not easy to implement
- not easy to use (selection of tuning parameters)
- adopts a global error evaluation, but the resulting approximation is not bounded
**Progressive Meshes**

Examples

1. Low-res model 5.0 (151 faces)
2. Medium res 10.5 (591 faces)
3. High-res model 15.0 (1,350 faces)

**Progressive Meshes - Evaluation**

- High quality of the results
- Preserves topology, re-sample vertices
- Not easy to implement
- Not easy to use (selection of tuning parameters)
- Adopts a global error evaluation, not-bounded approximation
- Preserves vector/scalar attributes (e.g. color)
- Supports multiresolution output, geometric morphing, progressive transmission, selective refinements
- Much faster than MeshOpt.

An implementation is present as part of DirectX 6.0 tools.

**Decimation**

**Mesh Decimation**

[Schroeder et al'92]

- Based on controlled removal of vertices
- Classify vertices as removable or not (based on local topology / geometry and required precision)

**Loop**

- Choose a removable vertex $v_i$
- Delete $v_i$ and the incident faces
- Re-triangulate the hole

until

- No more removable vertex or reduction rate fulfilled

**General method** (manifold/non-manifold input)

- Topologic classification of vertices
- Evaluation of the decimation criterion (error evaluation)
- Re-triangulation of the removed triangles patch
Topologic classification of vertices

➤ for each vertex: find and characterize the loop of incident faces

➤ interior edge: if dihedral angle between faces < \( k_{\text{angle}} \) (\( k_{\text{angle}} \): user driven parameter)

➤ not-removable vertices: complex, [ corner ]

Decimation criterion -- a vertex is removable if:

🔗 simple vertex:

if distance vertex - face loop average plane is lower than \( \epsilon_{\text{max}} \)

🔗 boundary / interior / corner vertices:

if distance vertex - new boundary/interior edge is lower than \( \epsilon_{\text{max}} \)

adopts local evaluation of the approximation!!

\( \epsilon_{\text{max}} \): value selected by the user

Re-triangulation

🔗 face loops in general non planar! (but star-shaped)

🔗 adopts recursive loop splitting re-triangulation

control aspect ratio to ensure simplified mesh quality

🔗 for each vertex removed:

🔗 if simple or boundary vertex \( \Rightarrow \) 1 loop

🔗 if interior edge vertex \( \Rightarrow \) 2 loops

🔗 if boundary vertex \( \Rightarrow \) - 1 face

🔗 otherwise \( \Rightarrow \) - 2 faces

Decimation - Examples

Decimation - Examples
... Decimation...

Original Mesh Decimation - Evaluation
- good efficiency (speed & reduction rate)
- simple implementation and use
- good approximation
- preserves topology; vertices are a subset of the original ones
- error is not bounded (local evaluation =⇒ accumulation of error!!)

... Enhancing Decimation -- Error Evaluation...

Heuristics proposed for global error evaluation:
- accumulation of local errors (Ciampalini97)
  fast, but approximate
- vertex--to--simplified mesh distance (Stuyck96)
  requires storing which of the original vertices maps to each simplified face;
  very near to exact value (but large under-estimation in the first steps)
  edge of initial mesh $M_0$
  edge of simplified mesh $M_i$
  error magnitude, $\text{dist}(v, M_i)$

... Enhancing Decimation -- Error Evaluation...

Approximation Error Evaluation

Classification of simplification methods based on approximation error evaluation:
- locally-bounded error, based on mesh distances (ex. standard Mesh Decimation)
- globally bounded error, based on mesh distances (ex. Envelopes + enhanced Decimation + others)
- control based on mesh characteristics (ex. vertex proximity, mesh curvature)
- energy function evaluation (ex. Mesh Optim., Progr. Meshes)

User viewpoint:
- simple to grasp, easy
- simple to drive, very handy
- may be misleading, not easy, many parameters to be selected

... Enhancing Decimation -- Error Evaluation...

Heuristics proposed for global error evaluation:
- input mesh -- to -- simplified mesh edges distance (Ciampalini97)
  for each internal edge:
  select sampling points $p_i$ (regularly/random)
  evaluate distance $d(M_0, p_i)$
  sufficiently precise and efficient in time

- input mesh -- to -- simplified mesh distance (Jarny96)
  precise, but more complex in time

- use envelopes (Cohen et al. '96)
  precise, no self-intersections but complex in time and to be implemented
Enhancing Decimation - **Simplification Envelopes**

**Simplification Envelopes** [Cohen et al.’96]

- given the input mesh $M$
- build two envelope meshes $M_+$ and $M_-$ at distance $\varepsilon$ and $\varepsilon'$ from $M$;
- simplify $M$ (following a decimation approach) by enforcing the decimation criterion: a candidate vertex may be removed only if the new triangle patch does not intersect neither $M_+$ nor $M_-$.

... Enhancing Decimation - **Simplification Envelopes** ...

- by construction, envelopes do not self-intersect $\Rightarrow$ simplified mesh is not self-intersecting!!
- distance between envelopes becomes smaller near the bending sections, and simplification harder
- **border tubes** are used to manage open boundaries

... Enhancing Decimation - **Simplification Envelopes** ...

**Simplification Envelopes - Evaluation**

- works on manifold surface only
- bounded approximation
- construction of envelopes and intersection tests are not cheap
- $\Rightarrow$ three times more RAM (input mesh + envelopes + border tubes)
- preserve topology, vertices are a subset of the original, prevents self-intersection

Results

- Simplification times $\sim$ linear with mesh size

*available in public domain*
Construction of a multiresolution model

Keep the *history* of the simplification process:

- when we remove a vertex we have *dead* and *newborn* triangles
- assign to each triangle *t* a *birth error* $t_b$ and a *death error* $t_d$
  equal to the error of the simplified mesh just before the removal of the vertex that caused the birth/death of *t*

By storing the *simplification history* (faces+errors) we can simply extract *any approximation level* in real time.

Real-time resolution management

- by extracting from the *history* all the triangles *t* with $t_b \leq \varepsilon < t_d$
  we obtain a model $M_{\varepsilon}$ which satisfies the approximation error $\varepsilon$
- maintaining the whole history data structure costs approximately 2.5x - 3x the full resolution model

Quadric Error Metrics

Simplification using Quadric Error Metrics

[Garland et al. Sig'97]

- Based on incremental edge-collapsing
- but can also collapse vertex couples which are not connected (topology is not preserved)

Geometric error approximation is managed by simplifying an approach based on plane set distance

[Ronfard, Rossignac96]

- INIT: store for each vertex the set of incident planes
- Vertex_Collapsing $(v_1, v_2) \Rightarrow v_{\text{new}}$
  - plane_set$(v_{\text{new}}) = \text{union of the two plane sets of } v_1, v_2$
  - collapse only if $v_{\text{new}}$ is not "farther" from its plane set than the selected target error $\varepsilon$

**criticism:**

- storing plane sets and computing distances is not cheap!
Quadric Error Metrics solution:
- quadratic distances to planes represented with matrices
- plane sets merge via matrix sums
- very efficient evaluation of error via matrix operations
- but
  - triangle size is taken into account only in an approximate manner (orientation only in Quadrics + weights)

Algorithm structure:
- select valid vertex pairs (upon their distance),
  insert them in a heap sorted upon minimum cost;
- repeat
  - extract a valid pair \( v_1, v_2 \) from heap and contract into \( v_{\text{new}} \);
  - re-compute the cost for all pairs which contain \( v_1 \) or \( v_2 \)
    and update the heap;
- until sufficient reduction/approximation or heap empty

Error Heuristics

Quadric Error for Surfaces

- Let \( n^Tv + d = 0 \) be the equation representing a plane
- The squared distance of a point \( x \) from the plane is
  \[
  D(x) = x(nn^t)x + 2dn^tx + d^2
  \]
- This distance can be represented as a quadric
  \[
  Q = (A,b,c) = (nn^t,dn,d^2)
  \]
  \[
  Q(x) = xAx + 2b^tx + c
  \]
Quadric

- The boundary error is estimated by providing for each boundary vertex \( v \) a quadric \( Q_v \), representing the sum of all the squared distances from the faces incident in \( v \).
- The error of collapsing an edge \( e = (v, w) \) can be evaluated as \( Q_v(v) \).
- After the collapse, the quadric of \( v \) is updated as follow \( Q_v = Q_v + Q_w \).

Error

Domain Error

- The two dataset \( D \) and \( D' \) span different domains \( \Omega, \Omega' \).
- The same problem of classical surface simplification.
- Measure the Hausdorff distance between the boundary surfaces of the two datasets \( D \) and \( D' \):
  \[ e(D, D') = \max \left( \min(\text{dist}(x, y)) \right) \quad x \in \Omega \quad y \in \Omega' \]
  \[ e(D, D') = \max(e(D, D'), e(D', D)) \]
- Various techniques to approximate this distance between two surfaces [Ciampalini et al. 97].

Quadric Error Metrics -- Evaluation

- Iterative, incremental method.
- Error is bounded.
- Allows topology simplification (aggregation of disconnected components).
- Results are very high quality and times incredibly short.
- Various commercial packages use this technique (or variations).
Not-incremental methods:

- coplanar facets merging
  [Hinker et al. '93, Kalvin et al. '96]
- re-tiling
  [Turk '92]
- clustering
  [Rossignac et al. '93, ... + others]
- wavelet-based
  [Eck et al. '95]

Geometric Optimization

- Construct nearly co-planar sets
  (comparing normals)
- Create edge list and remove duplicate edges
- Remove colinear vertices
- Triangulate resultant polygons

Superfaces

- group mesh faces in a set of superfaces:
  - iteratively choose a seed face, as the current
    superface $S_f$
  - find by propagation all faces adjacent to $f$, whose
    vertices are at distance $\varepsilon/2$ from the mean plane to
    $S_f$, and insert them in $S_f$
  - moreover, to be merged each face must have
    orientation similar to those of others in $S_f$
  - straighten the superfaces border
  - re-triangulate each superface

... Coplanar Facets Merging...
Superfaces - an example

- Simplification of a human skull (fitted isosurface), images courtesy of IBM

Re-tiling

- Distribute a new set of vertices into the original triangular mesh (points positioned using repulsion/relaxation to allow optimal surface curvature representation)
- Remove (part of) the original vertices
- Use local re-triangulation

no info in the paper on time complexity!

Superfaces - Evaluation

- slightly more complex heuristics
- evaluation of approximation error is more accurate and bounded
- vertices are a subset of the original ones
- preserves geometric discontinuities (e.g. sharp edges) and topology

Clustered Clustering

- detect and unify clusters of nearby vertices (discrete gridding and coordinates truncation)
- all faces with two or three vertices in a cluster are removed
- does not preserve topology (faces may degenerate to edges, genus may change)
- approximation depends on grid resolution

(figure by Rossignac)
Clustering -- Examples (1)

- Simplification of a table lamp, IBM 3D Interaction Accelerator, courtesy IBM

10,108 facets 1,383 facets 474 facets 46 facets

Clustering -- Examples (2)

- Simplification of a portion of Cluny Abbey, IBM 3D Interaction Accelerator, courtesy IBM France.

46,918 facets 6,181 facets 1,790 facets 16 facets

... Clustering...

Clustering - Evaluation

- high efficiency (but timings are not reported in the paper)
- very simple implementation and use
- low quality approximations
- does not preserve topology
- error is bounded by the grid cell size

Wavelet methods

Multiresolution Analysis [Eck et al. '95, Lounsbery '97]

- Based on the wavelet approach
  - simple base mesh
  - + local correction terms (wavelet coefficients)
- Given input mesh M:
  - partition : build a low resolution base mesh \( K_0 \) with tolerance \( \varepsilon_1 \)
  - parametrization : for each face of \( K_j \) build a parametrization on the corresponding faces of \( M \)
  - resampling : apply \( j \) recursive quaternary subdivision on \( K_j \) to build by parametrization different approximations \( K_j \)
- Supports:
  - bounded error, compact multiresolution repr., mesh editing at multiple scales
Preserving detail on simplified meshes

- **Problem Statement:**
  - how can we preserve in a simplified surface the **detail** (or **attribute value**) defined on the original surface??

- **What one would preserve:**
  - **color** (per-vertex or texture-based)
  - **small variations of shape curvature** (bumps)
  - **scalar fields**
  - **procedural textures** mapped on the mesh

**Multires Signal Processing for Meshes**

- Still the **Partition, Parmetrization and Resampling** approach but the original mesh connectivity is retained:
  - partition is done on the simplified mesh
  - use of a **non-uniform relaxation procedure** (instead of standard triangle quadrissection) that mimics the inverse simplification process
  - Possibility of using signal processing techniques on mesh (eg. Smoothing, detail enhancement ...)

**Approaches proposed in literature are:**

- **integrated** in the simplification process
  (ad hoc solutions **embedded** in the simplification codes)

- **independent** from the simplification process
  (post-processing phase to restore attributes detail)
**Integrated approaches:**

- attribute-aware simplification
  - do not simplify an element \( e \) **IF** \( e \) is on the boundary of two regions with different attribute values
  - or
  - use an enhanced multi-variate approximation evaluation metrics (shape+color+...)
    
    [Hoppe95, Garritz98, Frank et al98, Cohen et al98]

- store removed detail in textures
  - vertex-based [Narasim95, Soucy et al96]
  - texture-based [Kim et al95]

- preserve topology of the attribute field [Hoppe et al98]

**A simple idea:**...Preserving detail: Simplif.-Independent...

- for each texel simplified face:
  - detect the original detail by choosing either the closest point or along the normal.

**Simplification-Independent approach:**

- **higher generality:** attribute/detail preservation is not part of the simplification process
- performed as a post-processing phase (after simplification)
- any attribute can be preserved, by constructing an ad-hoc **texture map**
- Used today in most games...

**an example of color preservation**
...Preserving detail: Simplif.-Independent...

Example of **geometric detail** preservation by **normal mapping**

- Original 20k face
  - Simplified 500 faces

- Original 60k faces
  - Simplified 250 faces