CS 563 Advanced Topics in Computer Graphics
Polygonal Techniques

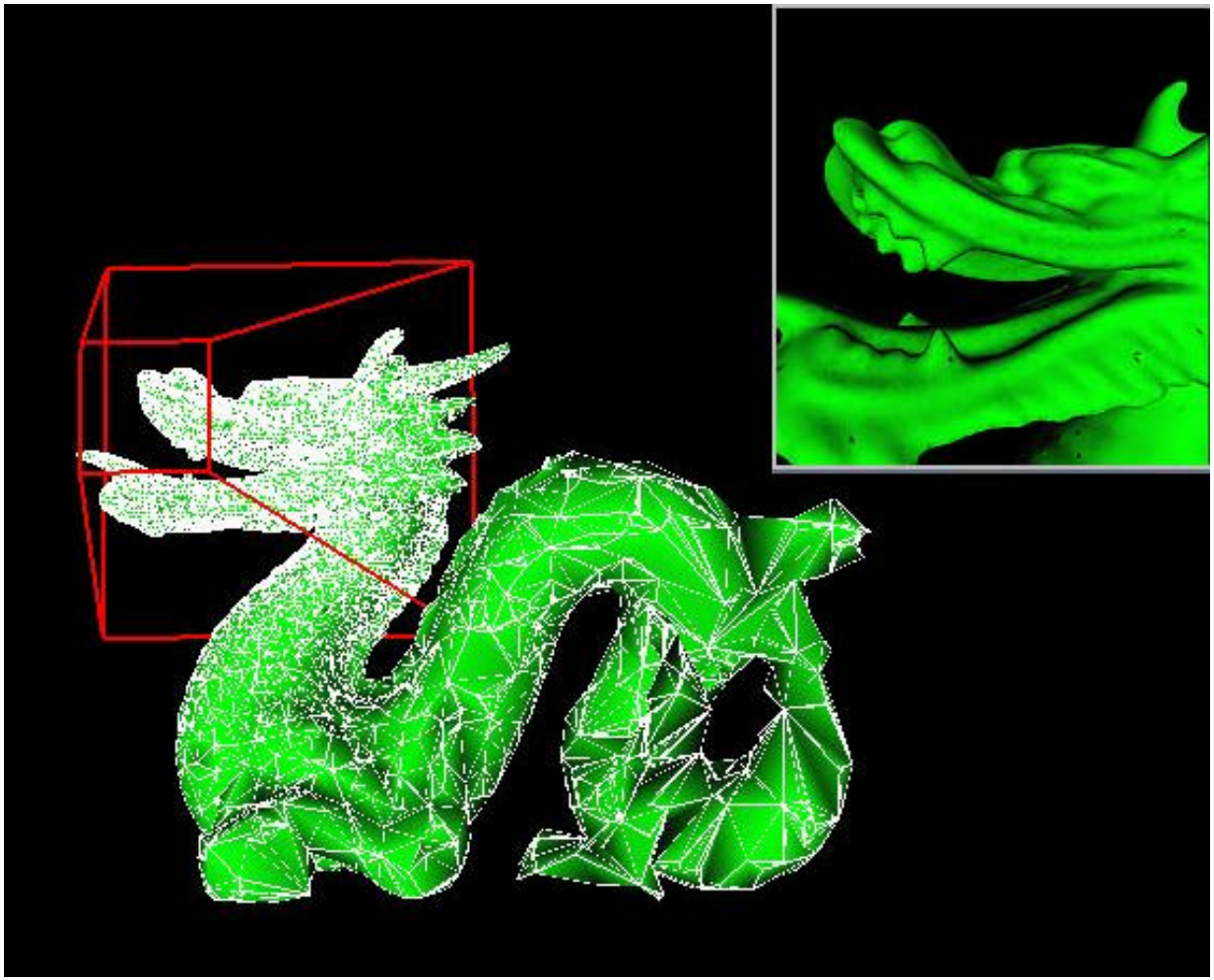
by Linna Ma

What I'll Talk About

- Introduction
- Tessellation and Triangulation
- Consolidation
- Triangle Strips, Fans and Meshes
- Simplification

Introduction

Polygonal Models



Introduction

- Definitions
  - Tessellation: Polygon have to be split into more tractable primitives Eg.Triangles or quadrilaterals..
  - Consolidation: encompasses merging and linking polygonal data

    ->deriving new data from polygonal data sets

    (normal vector or orientation .....)
  - Simplification: removing unneeded polygons

    -> improve speed

Introduction

- Relations
  - Tessellation ensures data displaying correctly
  - Consolidation improves data display by sharing computations
  - Simplification more improves data display by removing unneeded polygons

Introduction

Sources of 3D Data

- Various ways to describe models
  - by directly typing in the data
  - by writing programs creating such data
    (Procedural modeling)
  - by transforming data found in other forms into surfaces or volumes
  - by using three-dimensional scanners
  - by some combination of these techniques
  - *polygonal data* generated by these methods
    ➔ our focus!

Introduction

- Two main type of modeler
  - solid-based modeler
  - surface-based modeler
- Solid-based modeler
  - in the area of CAD
  - emphasize modeling tool
  - (actual machining processes such as cutting, drilling..)
  - faceter : software that turns internal model representation into polygons

Introduction

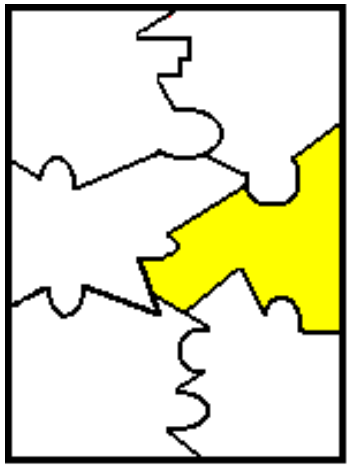
- Surface-based modeler
  - do not have a built-in concept of solidity
  - all objects are thought of surfaces
  - they may also use faceters such as spline surfaces
  - direct manipulation of surfaces
- Other types of modelers
  - implicit surface creation systems
  - satellite imagery, medical scanning devices, three dimensional scanners, image registration techniques
- The key
  - understand how data was created and for what purpose

What I'll Talk About

- Introduction
- Tessellation and Triangulation
- Consolidation
- Triangle Strips, Fans and Meshes
- Simplification

Tessellation

- History
  - The word tessellation comes from Latin word *tessella*
  - Meaning “a square tablet”
  - The square tablets were used to make ancient Roman mosaics
- Tessellation: the process of splitting a polygon into a set of polygons
- For example: A tessellation is like a puzzle that repeats a particular pattern



Tessellation

- variety of reasons
  - APIs and hardware are optimized for triangles

  (triangles almost like atoms .....)
  - may only handle convex polygons
  - catch shadows and reflected light using radiosity techniques
  - non graphical reason....

Tessellation

- Various types of tessellation

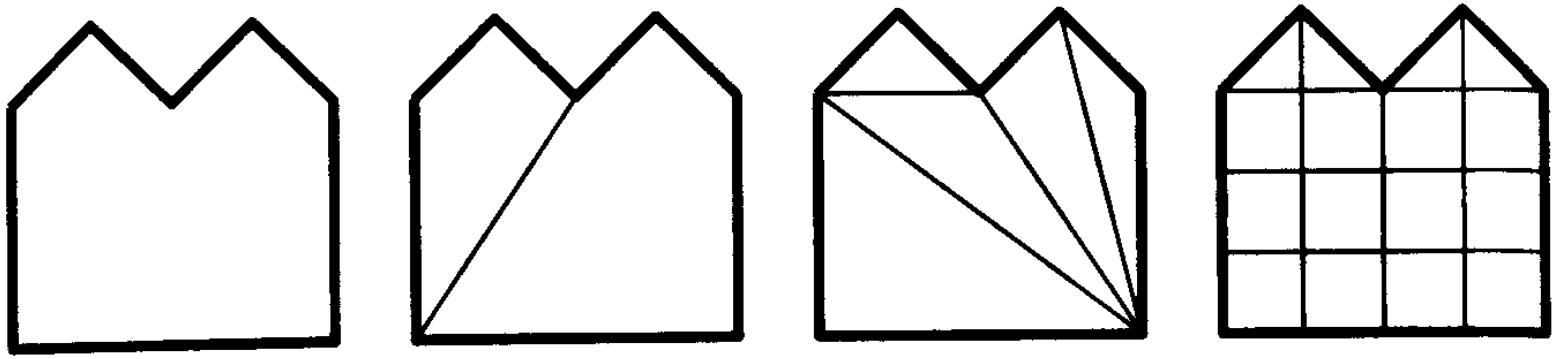
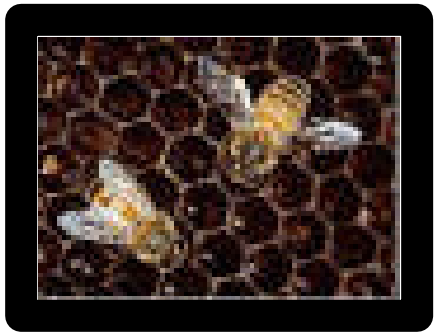


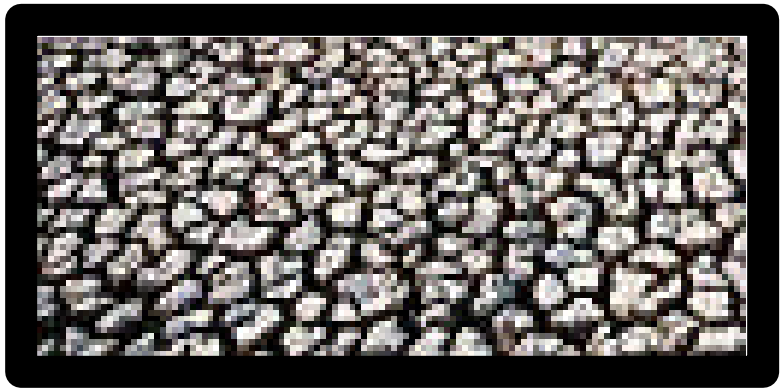
Figure 9.1. Various types of tessellation. The leftmost polygon is not tessellated, the next is partitioned into convex regions, the next is triangulated, and the rightmost is uniformly meshed.

Tessellation

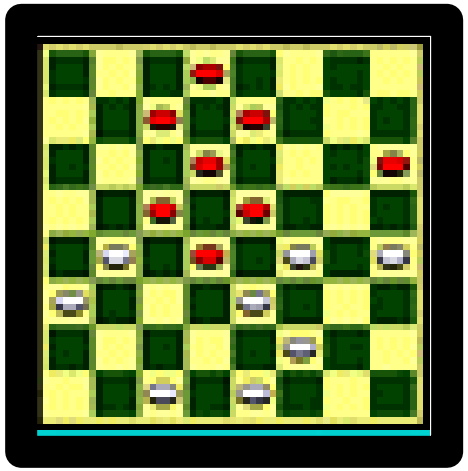
In nature and life, they appear often:



Honeycombs...



Mud flats...

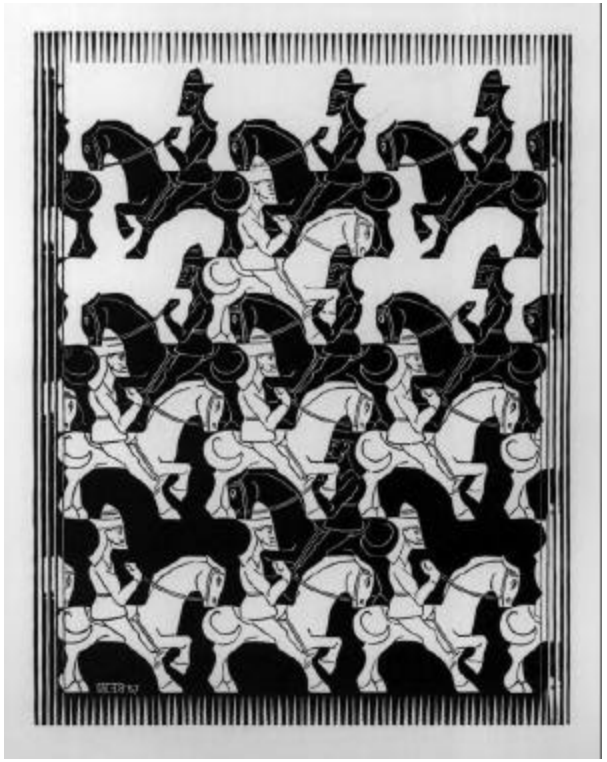


In games, like checkers..

Tessellation

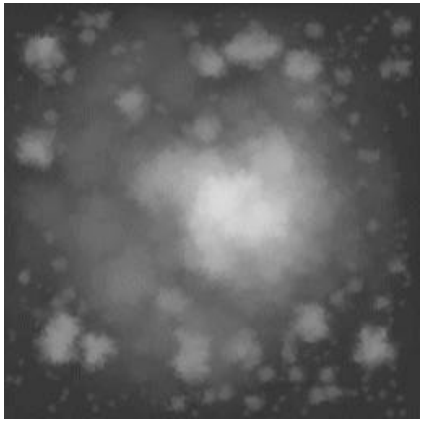
Sun and Moon

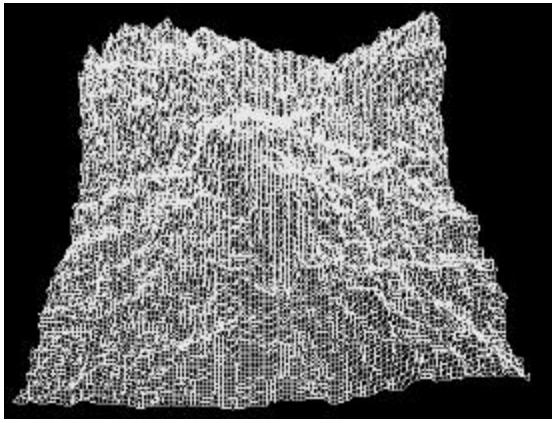
Horsemen

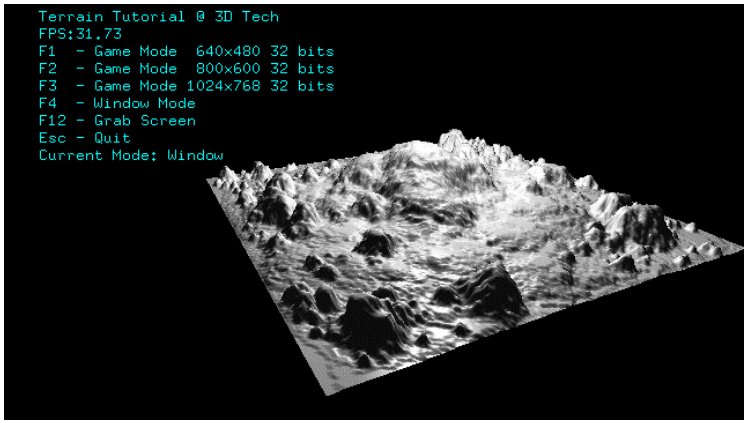


Tessellation

- More examples:







Tessellation

- Tessellation Algorithm
  - writing a robust and general tessellator is a difficult undertaking
  - various subtle bugs, pathological cases and precision problems....
- steps for surface tessellation
  - how best to project 3D into 2D
  - (x,y,z) -> (x,y) or (y,z) or (x,z)
  - the best plane is the largest projected area
    - compute directly the area
    - throw away the coordinates corresponding to the coordinate of greatest magnitude in the polgon's normal

      ex) polygon normal (-5,2,4) -> throw away x coordinate
  - the area and normal test are not always equivalent

Tessellation

- badly warped
- self intersection -> laborious comparision

  10.12 line/line intersection test
- hourglass or bowtie quadrilateral

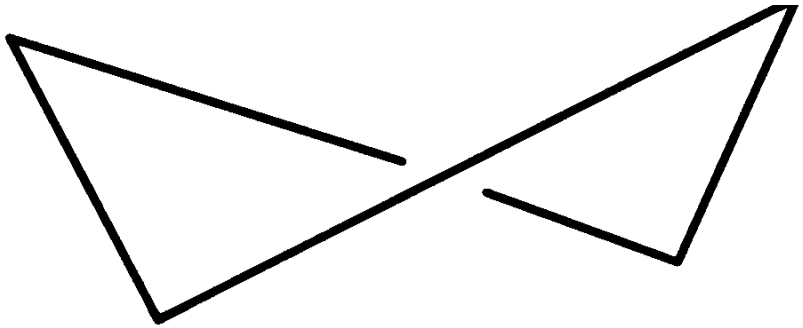
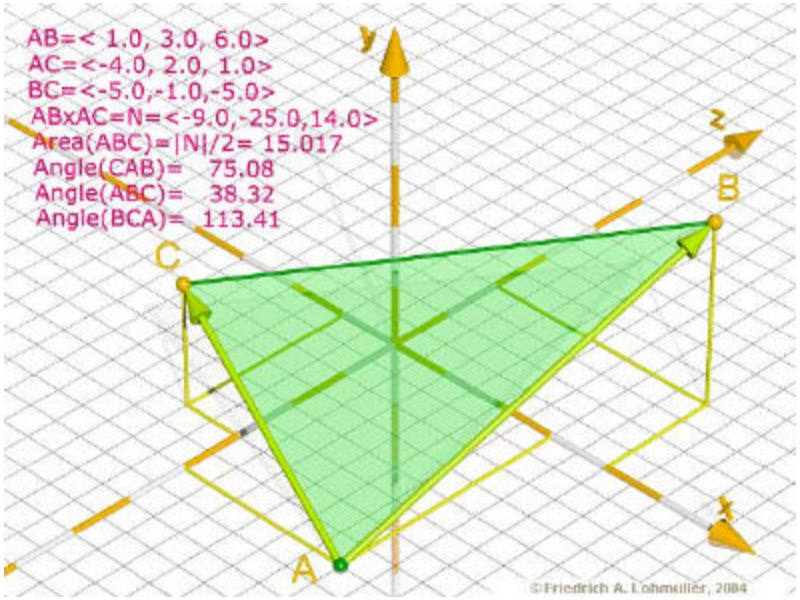


Figure 9.2. Warped quadrilateral viewed edge-on, forming a bowtie or hourglass figure.

Tessellation

- Triangulation



- Why do we use triangles in graphics?
  - Normals are unique: Important for lighting
  - Space filling: triangles tile so they can be used to represent surfaces.
  - Atomic: All polygons decompose into triangles

Tessellation

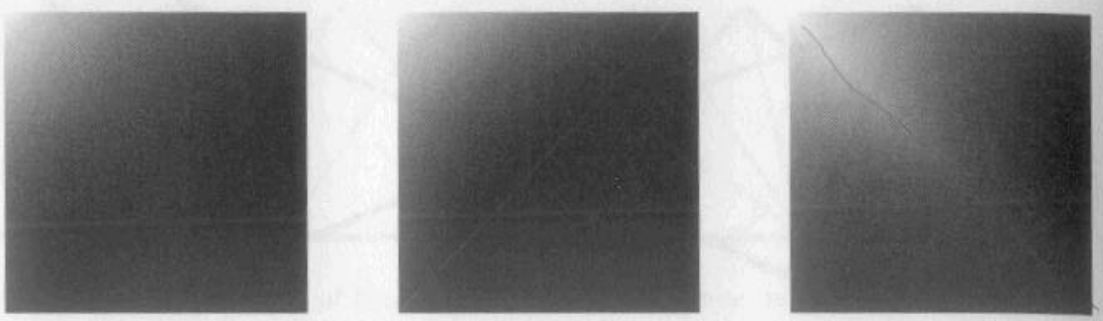


Figure 9.4. The left figure is rendered as a quadrilateral; the middle is two triangles with upper-right and lower-left corners connected; the right shows what happens when the other diagonal is used. The middle figure is better visually than the right one.

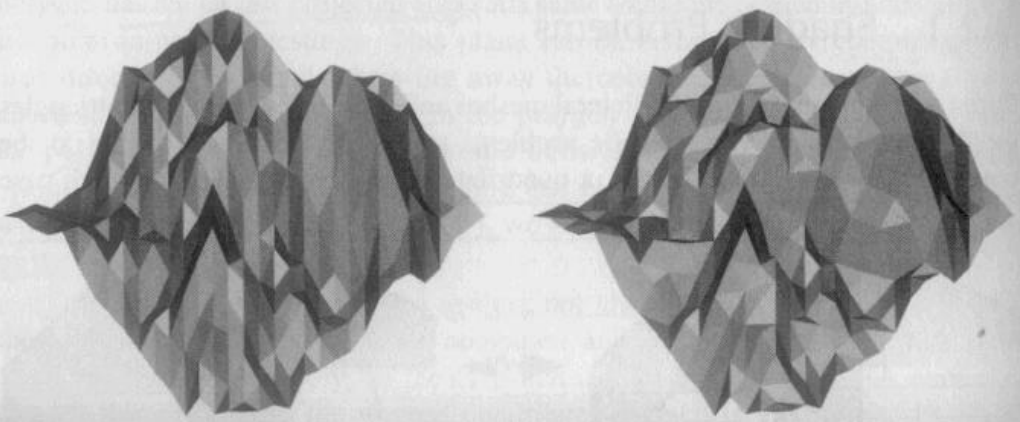


Figure 9.5. On the left is a height field mesh with each quadrilateral split along the same diagonal (the view looks along these diagonals—note how visible these are). On the right, the split is made using the diagonal whose corners are closer in height, and so breaks up the pattern formed.

Shading Problem

Tessellation

- different ways to split a quadrilateral
  - minimize differences, shortest diagonal
  - smaller difference between the colors
    (for radiosity solution or prelit quadrilaterals)
  - the largest angle
  - closest in size
  - closest in height
  - most different in height

Tessellation

- problems of triangulation
  - # a warped quadrilateral
    - inherent limitation of triangulation
      cannot warp any part of an image on a single triangle
  - # Shading problem
    - neither does pure Gouraud shading using the untessellated quadrilateral
    - colors are different according to the way splitting the quadrilateral

Tessellation

- Rotation Invariant

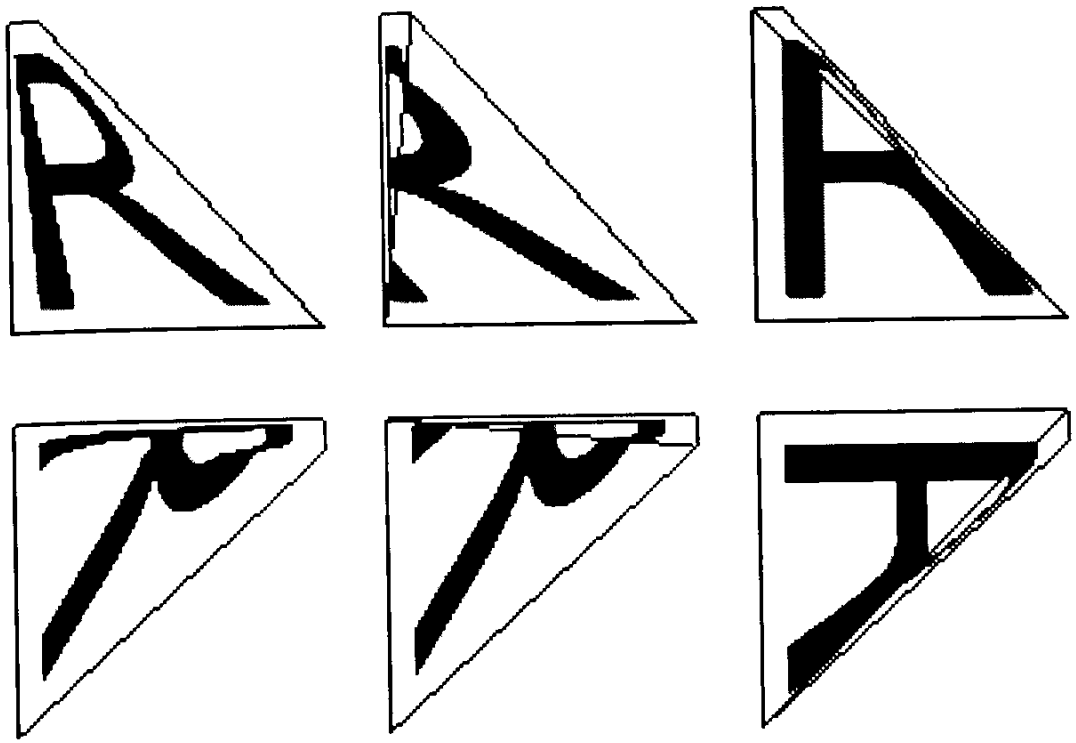


Figure 9.6. The top left shows the intent of the designer, a distorted quadrilateral with a square texture map of an "R." The two images on the right show the two triangulations and how they differ. The bottom row rotates all of the polygons; the Gouraud-interpolated quadrilateral changes its appearance.

Tessellation

- Edge Cracking

  - the points generated for one spline curve do not match those generated by its neighbor

  - edge stitching : the process of fixing these cracks

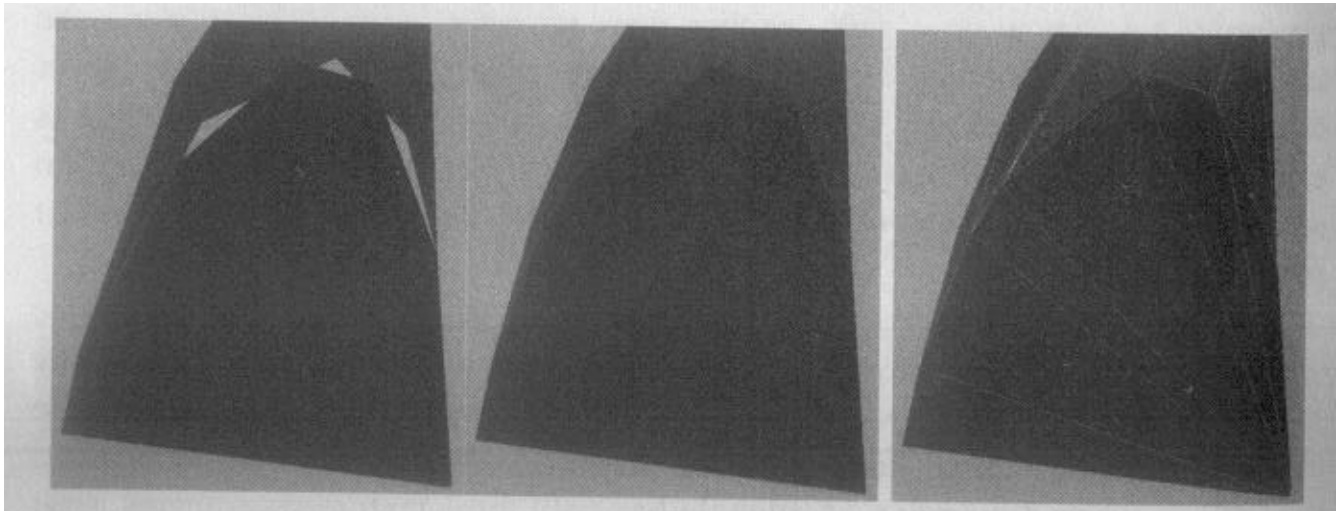


Figure 9.7. The left figure shows cracking where the two surfaces meet. The middle shows the cracking fixed by matching up edge points. The right shows the corrected mesh.

Tessellation

- T – vertices

  - encountered when joining *flat surfaces*
  - occur radiosity solution
  - occur when model surfaces meet but do not share all vertices.
  - even though the edges should theoretically meet perfectly, if the renderer does not have enough precision in representing vertex locations on the screen

Tessellation

- the way to ameliorate T-vertices

  - graphic hardware with subpixel addressing

  - find such edges and make sure to share vertices with all bordering faces (totally eliminated)

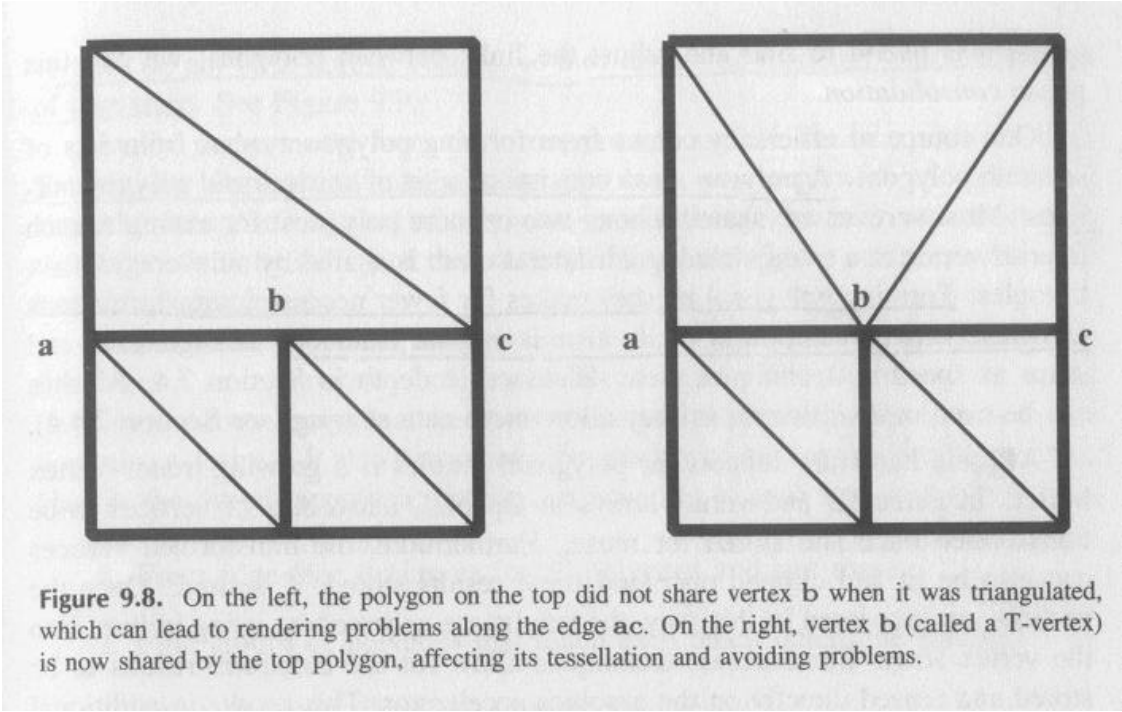


Figure 9.8. On the left, the polygon on the top did not share vertex b when it was triangulated, which can lead to rendering problems along the edge ac. On the right, vertex b (called a T-vertex) is now shared by the top polygon, affecting its tessellation and avoiding problems.

Tessellation DONE!

What I'll Talk About

- Introduction
- Tessellation and Triangulation
- Consolidation
- Triangle Strips, Fans and Meshes
- Simplification

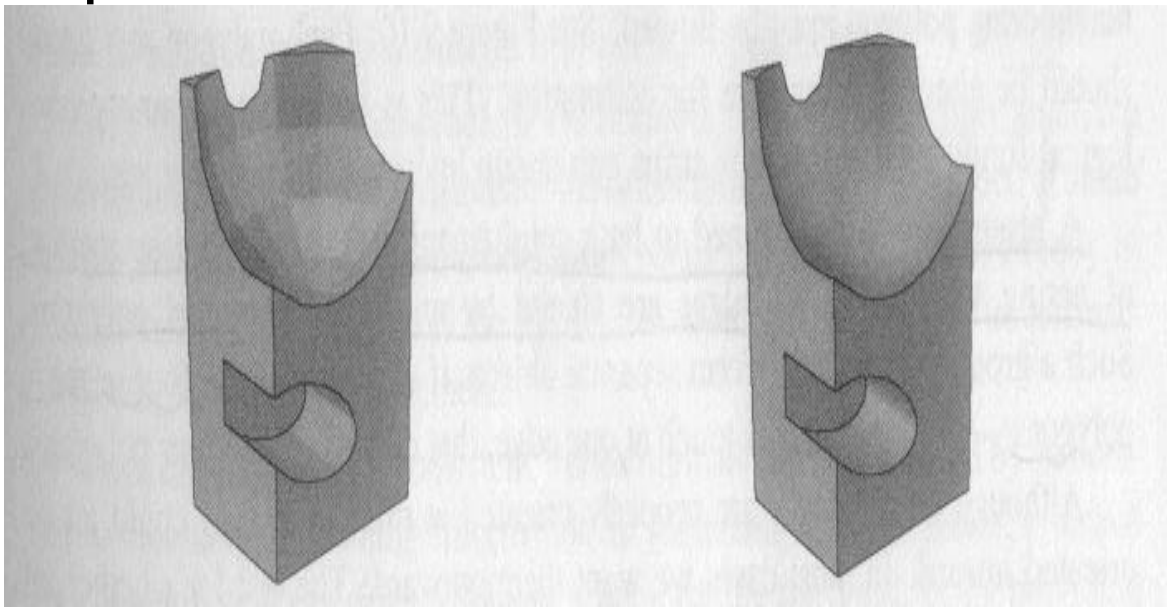
Consolidation

- Consolidation
  - find and adjust the links between polygons

- *Polygon mesh* for efficiency
  - consists of a set of vertices and polygon outlines
  - fewer necessary transformations and more efficient clipping, while also saving on memory
  - triangle fans and strips
  - API and hardware support for polygonal meshes
    - Direct3D : vertex buffers
    - OpenGL : vertex arrays

Consolidation

- *Backface culling* for efficiency
  - discussed in depth in 7.1
  - solid (= maniford or closed)
  - none of the backfaces should ever be visible from the outside
- Importance for orientation and normals



Consolidation

- Algorithm forming orientations and normals (no information of orientations and normals)
  1. Form edge-face structures for all polygons and sort these.
  2. Find groups of polygons that touch each other: determine solidity.
  3. For each group, flip faces to gain consistency.
  4. Determine what the inside of the mesh is, and flip all faces if needed.
  5. Find smoothing groups and compute vertex normals.
  6. Find boundaries.
  7. Create polygonal meshes.

Consolidation

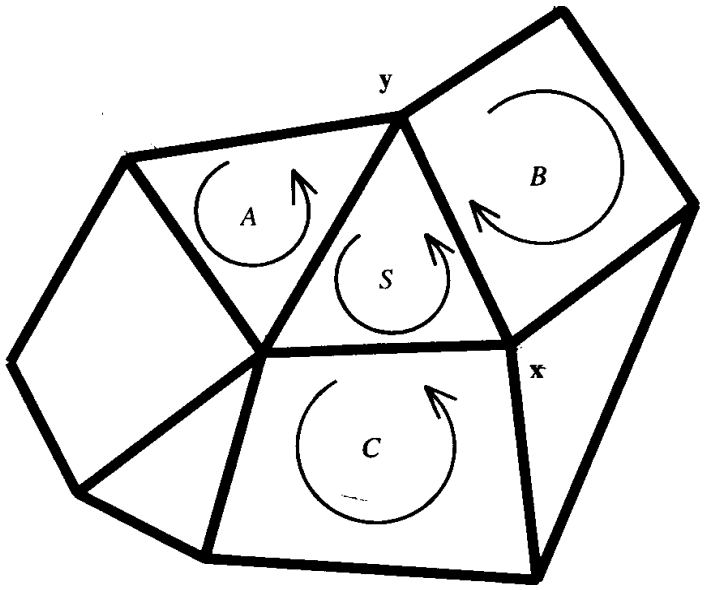
1. Form edge-face structures for all polygons and sort these
   - Form a list of all edges in the entire set of polygons, with each edge referring back to the face with which it is associated.
   - Each edge is stored with the vertex order
   - After edge list is formed, compare vertex pairs.
     - Checking edges with no length, and identical edges

2. Find groups of polygons that touch each other
   : determine solidity
   - Test whether all its degrees are shared by an even number of polygons.
   - If any edges share four or more polygons -> separated object

Consolidation

3. For each group, flip faces to gain consistency

- We want all polygons to have counter-clockwise outlines.
- Orientation check
  - If the direction of traversal for the edge is the same for both polygons, then the neighboring polygon must be flipped.



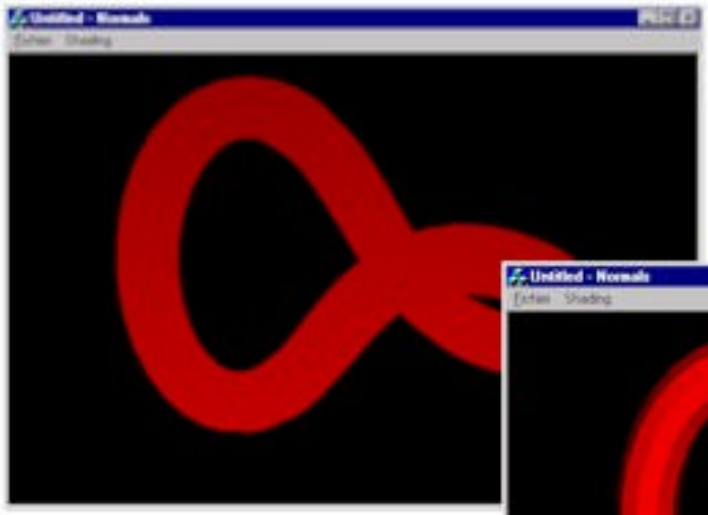
Consolidation

4. Determine what the inside of the mesh is, and flip all faces if needed
   - Signed volume
     - Get the first center point of the group's bounding box.
     - Check the sign of the volume of tetrahedron formed by a triangle and the center point.

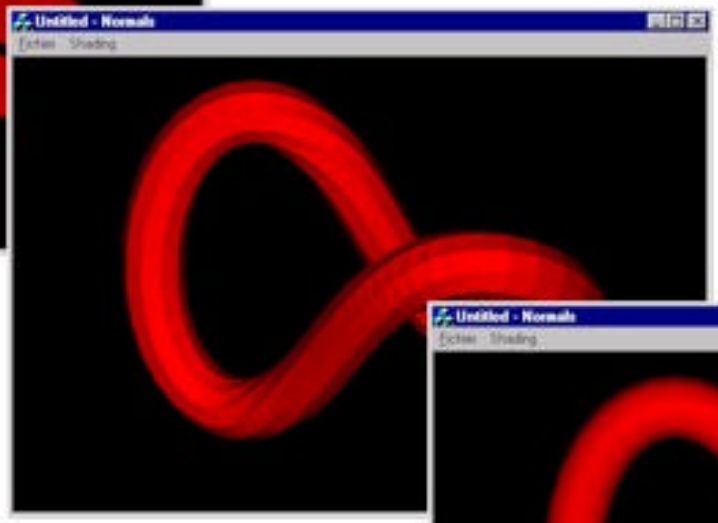
Consolidation

5. Find smoothing groups and compute vertex normals
   - Smoothing group information
   - Smoothing angle
     - Surface normal vs. specified angle
     - Determine smoothing group
   - Compute vertex normals
     - Average normals of polygons

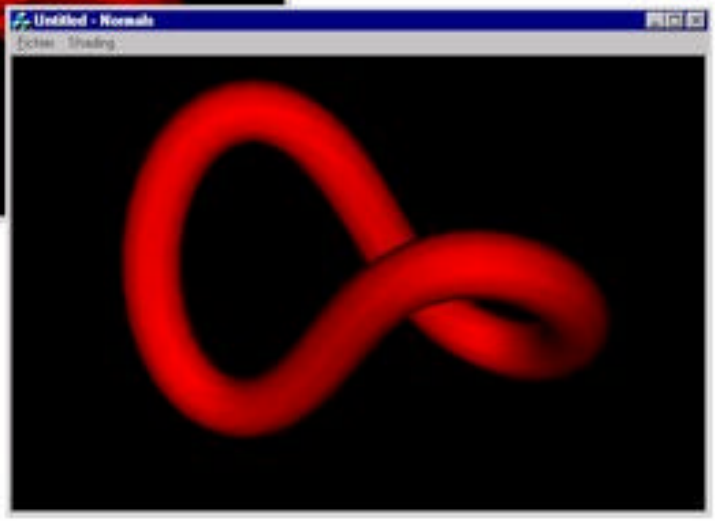
Consolidation



No normals
No shading



Face normals
Flat shading



Vertex normals
Smooth shading

Consolidation

6.Find boundaries

identifying all edges in smoothing group

7. Create polygonal meshes

Consolidation DONE!

What I'll Talk About

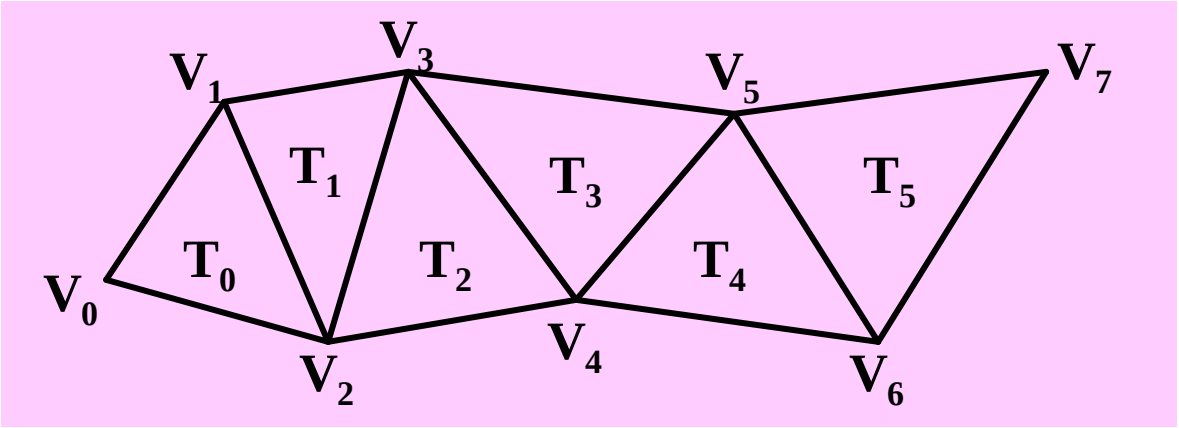
- Introduction
- Tessellation and Triangulation
- Consolidation
- Triangle Strips, Fans and Meshes
- Simplification

Triangle Strips, Fans and Meshes

- An extremely common way to increase graphics performance is to send fewer than three vertices per triangle to the graphics pipeline.
- The speed-up,avoiding redundant operations
  - Lighting calculations
  - Clipping
  - Matrix transformations, etc.
- Two popular methods of using less data
  - Triangle strips
  - Triangle fans

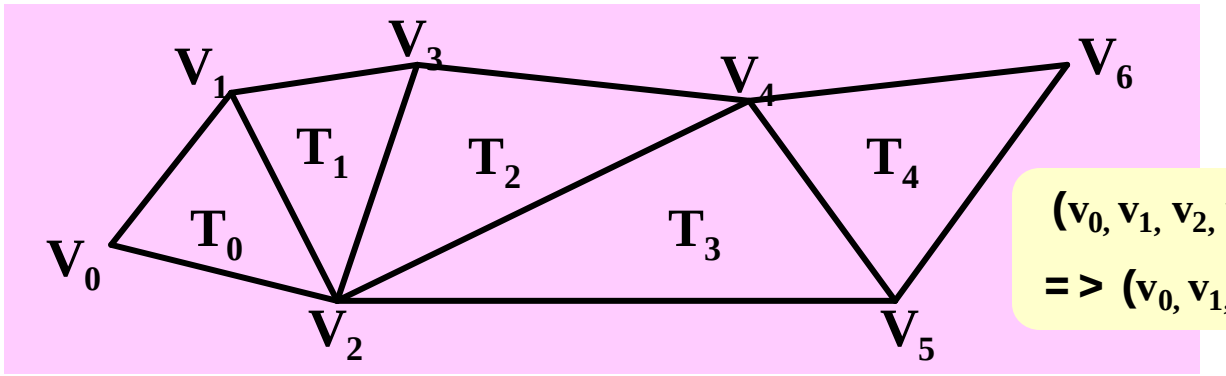
Strips(1)

- Triangle strips
  - A sequence of connected triangles
  - A triangle strip of *n* vertices is defined as an ordered vertex list $\{v_0, v_1, \ldots, v_{n-1}\}$, and triangle *i* is *delta* $v_i v_{i+1} v_{i+2}$.
  - A triangle strip of *n* vertices has *n-2* triangles.
  - The average number of vertices : $v_a = (3+(\boldsymbol{m}-1))/m = 1+2/m$
  - The fastest way to draw 3D surfaces



Strips(2)

- Swap
  - In order to create longer and more efficient triangles strips
  - A penalty of one vertex per swap
  - Swap cost < Restarting cost
  - Iris GL : an actual command for doing a swap
  - OpenGL, Direct3D : a swap command must be implemented by resending a vertex



Strips(3)

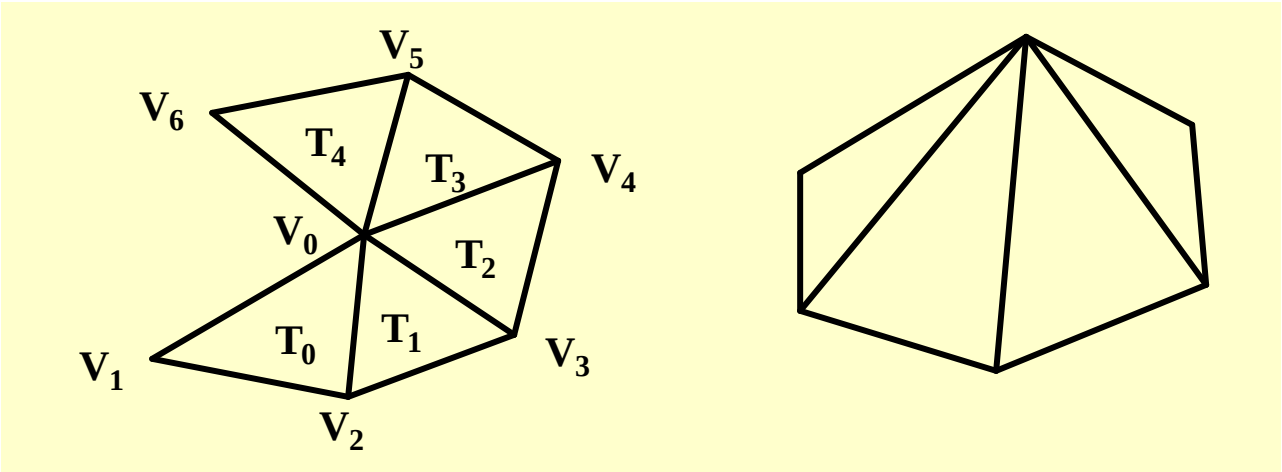
- How to implement triangle strips (using OpenGL) **ex).**
  { v_0, v_1, v_2, v_3, v_2, v_4, v_5, v_6 }

```
glBegin (GL_TRIANGLE_STRIP);
    glVertex3fv ( v0 );
    glVertex3fv ( v1 );
    glVertex3fv ( v2 );
    glVertex3fv ( v3 );
    glVertex3fv ( v2 );
    glVertex3fv ( v4 );
    glVertex3fv ( v5 );
    glVertex3fv ( v6 );
glEnd();
```

Triangle Drawn	R1	R2	vNew	P
				R1
	v_0		v_0	R2
	v_0	v_1	v_1	R1
(v_0, v_1, v_2)	v_2	v_1	v_2	R2
(v_2, v_1, v_3)	v_2	v_3	v_3	R1
(v_2, v_3, v_2)	v_2	v_3	v_2	R2
(v_2, v_3, v_4)	v_2	v_4	v_4	R1
(v_2, v_4, v_5)	v_5	v_4	v_5	R2
(v_5, v_4, v_6)	v_5	v_6	v_6	R1

Fans

- Triangle fans
  - A triangle fan of *n* vertices is defined as an ordered vertex list $\{v_0, v_1, \ldots, v_{n-1}\}$, and triangle *i* is *delta* $v_0 v_{i+1} v_{i+2}$. ($0 <= i < n - 2$)
  - Useful for rendering filled arcs and circles



Creating Strips(1)

- Softwares(GL) for creating strips
  - IRIS Performer (SGI, 1997)
  - OpenGL Optimizer (SGI, 1997)
- Obtaining optimal triangle strips
  - NP-complete problem.
  - Heuristic methods : close to the lower bound on the number of strips
- Two methods for constructing triangle strips
  - The SGI algorithm
  - STRIPE

Creating Strips(2)

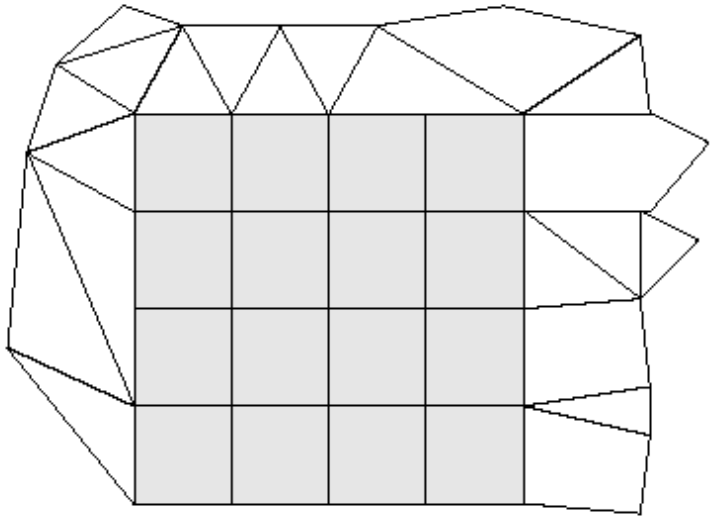
- The SGI algorithm
  - K. Akeley et al. “tomesh.c”, C *Program on SGI Developer's Toolbox CD*, 1990.
  - Work only for fully triangulated models
  - Greedy algorithm (locally optimal method)
  - The next triangle in the triangle strip it chooses always has the lowest degree.
  - Linear time implementation
    - Hash tables : store the adjacency data structure
    - Priority queue : for finding the starting triangle

Creating Strips(3)

- The STRIPE algorithm
  - To gain efficiency by acting on a global level
  - Take advantage of the fact that models often have polygons with more than three vertices.
  - Such polygons can be triangulated in a variety of ways

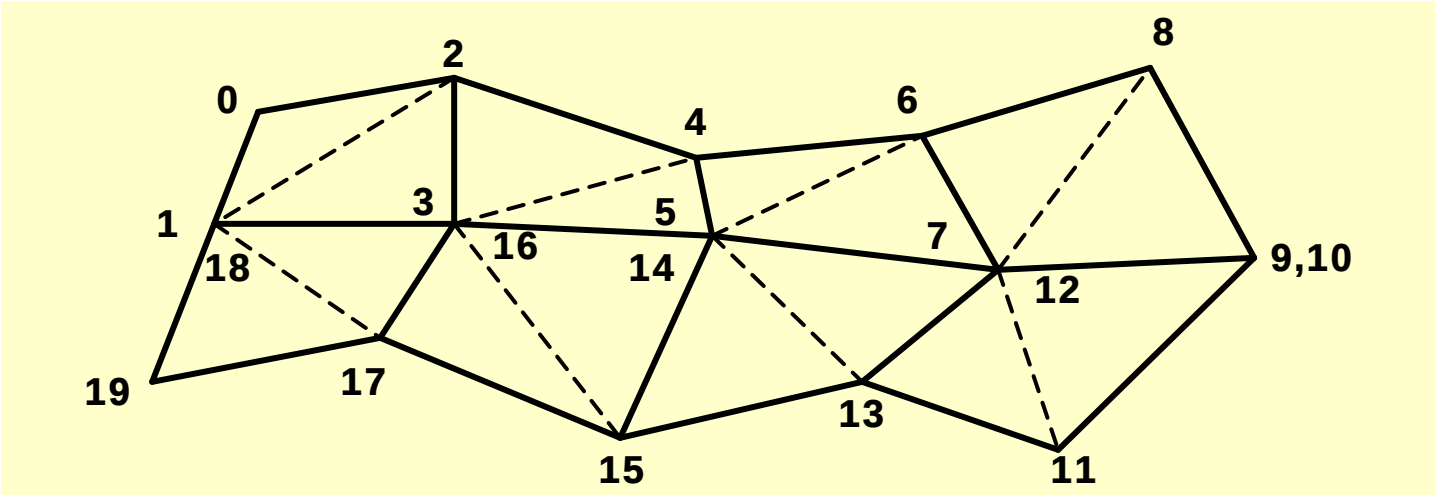
Creating Strips(4)

- Evans et al. (1996)
  - The idea
    - Typical polyhedral models contain many quadrilateral faces
    - These are often connected in large regions
    - Easy to turn into long triangle stripes
  - Patchification : to find large patches (rectangular regions made of quads



Creating Strips(5)

- To convert a patch into a triangle strip
  - Turn each row of the patch into a triangle strip
  - Connect the next row by turning the corner
- It **costs three swaps** at each turn of a patch
- Alternatively it **costs two extra vertices** if **one vertex is doubled** (because turns are expensive)



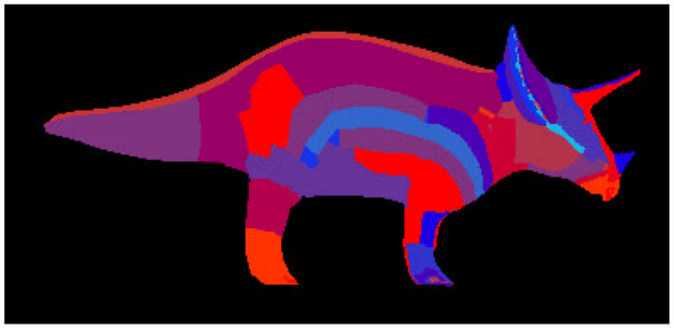
Creating Strips(6)

- Experimental results
  - Practically no need to develop better algorithm

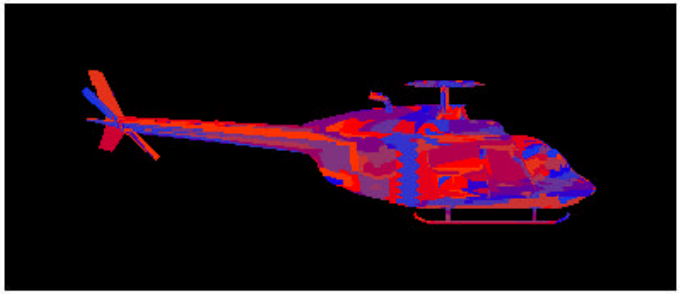
    (because the current algorithms are close to the theoretical lower bound)
  - Between 10% and 30% better than the SGI algorithm
  - Better performance more from reducing the number of swaps than from reducing the number of strips

Creating Strips(7)

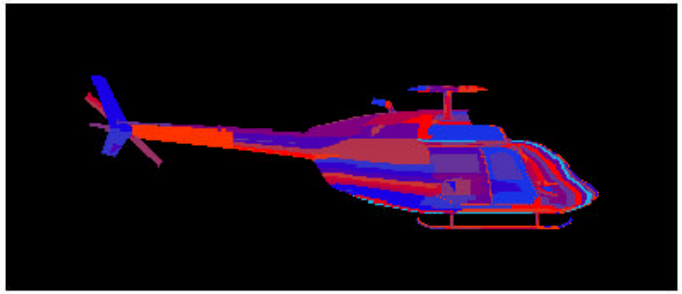
SGI



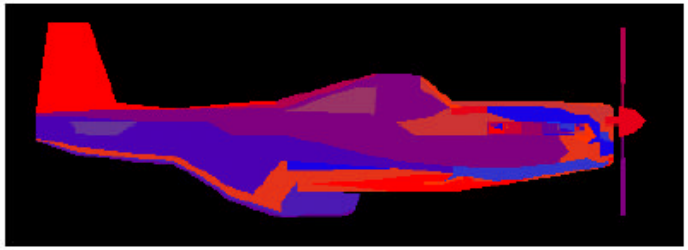
Stripe



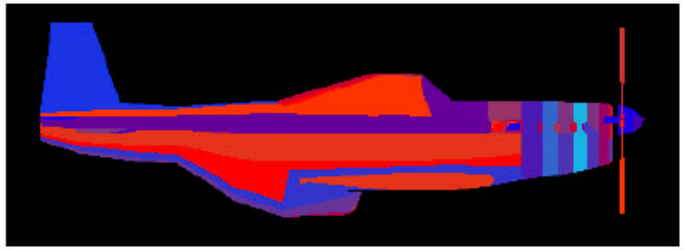
SGI



Stripe



SGI



Stripe

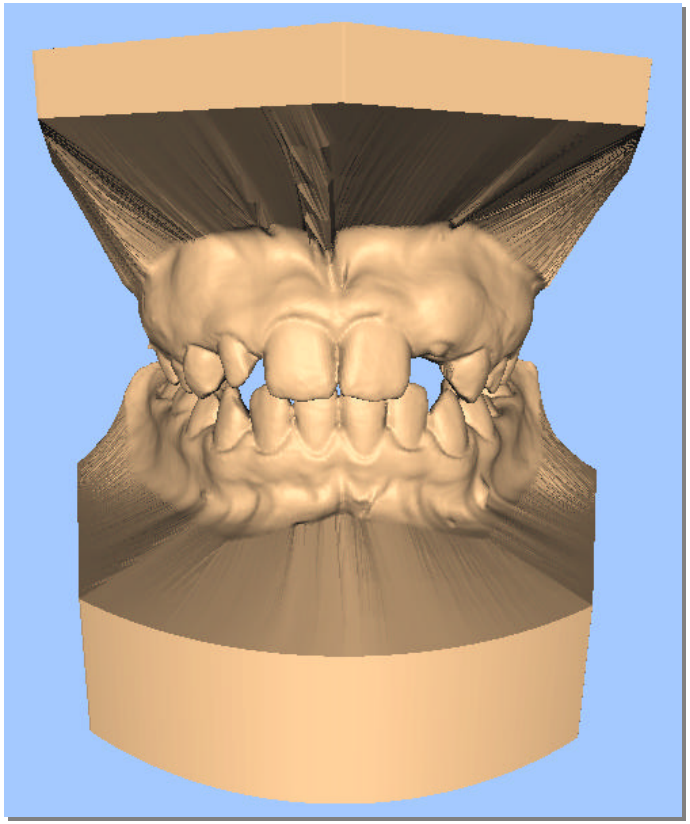
Triangle Strips, Fans and Meshes DONE!

What I'll Talk About

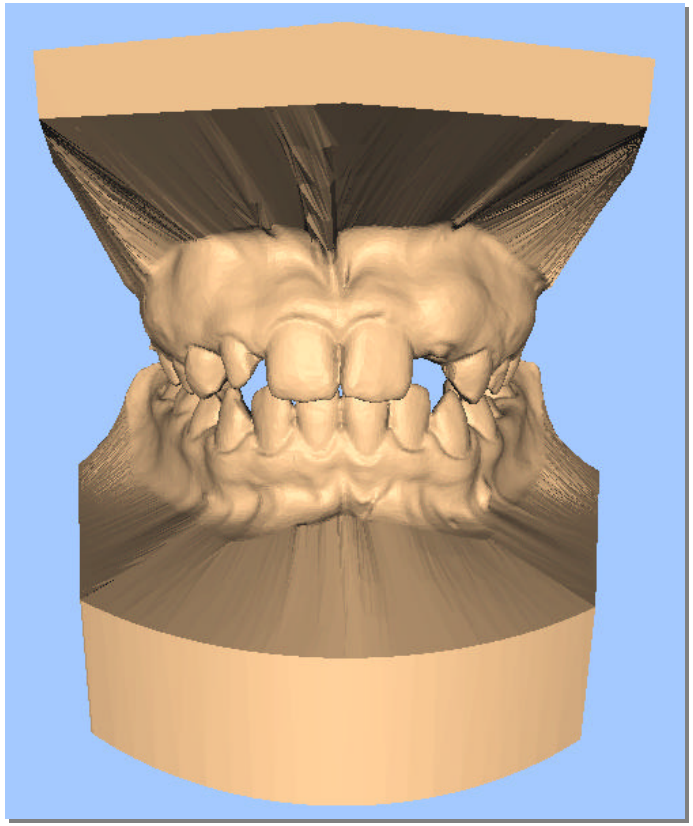
- Introduction
- Tessellation and Triangulation
- Consolidation
- Triangle Strips, Fans and Meshes
- Simplification

Simplification

A non-economical model



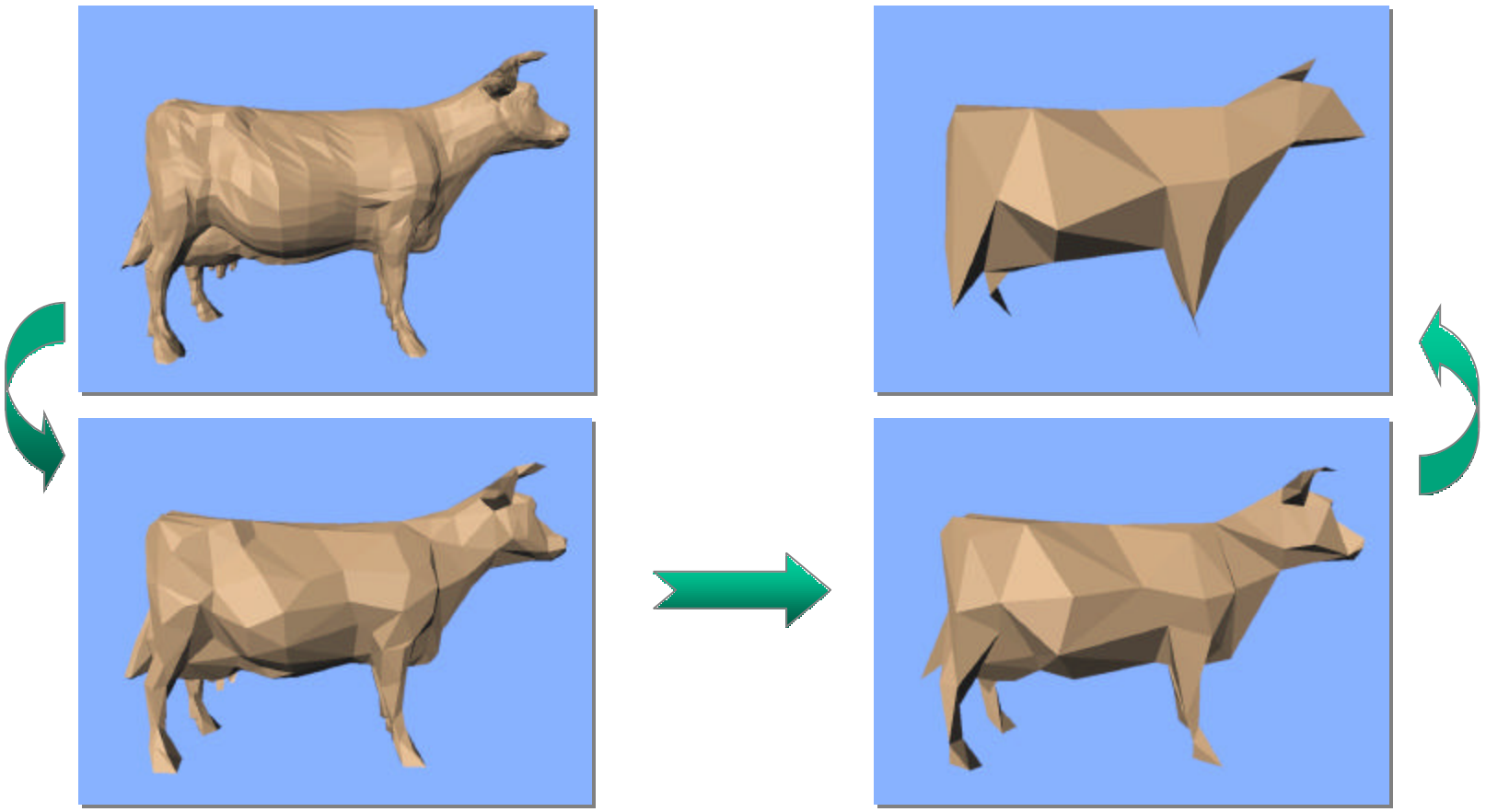
424,376 faces



60,000 faces

Simplification

Automatic surface simplification



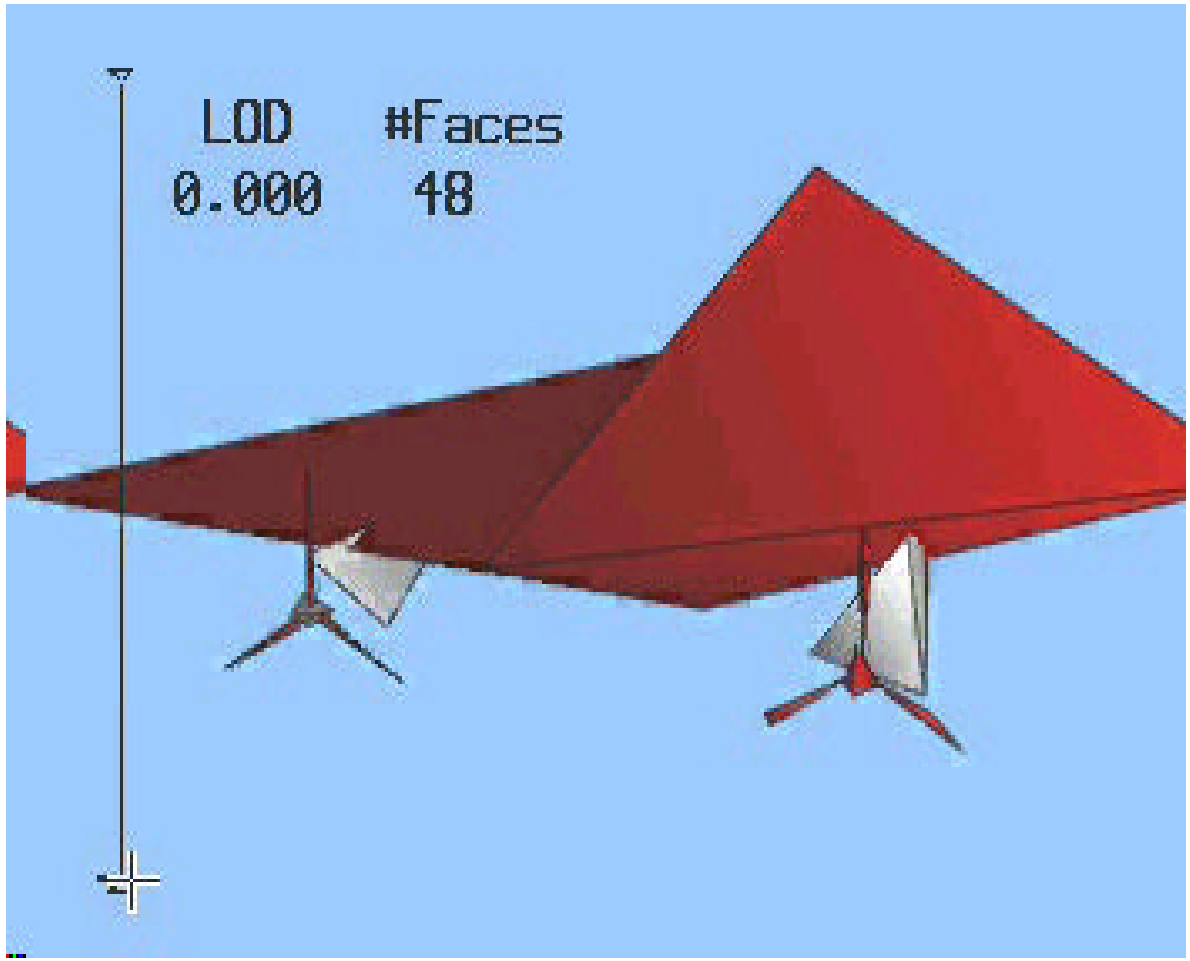
Simplification

- Subset placement strategy
  - Melax (1998)
  - Faster (fewer possibilities)
  - Lower-quality

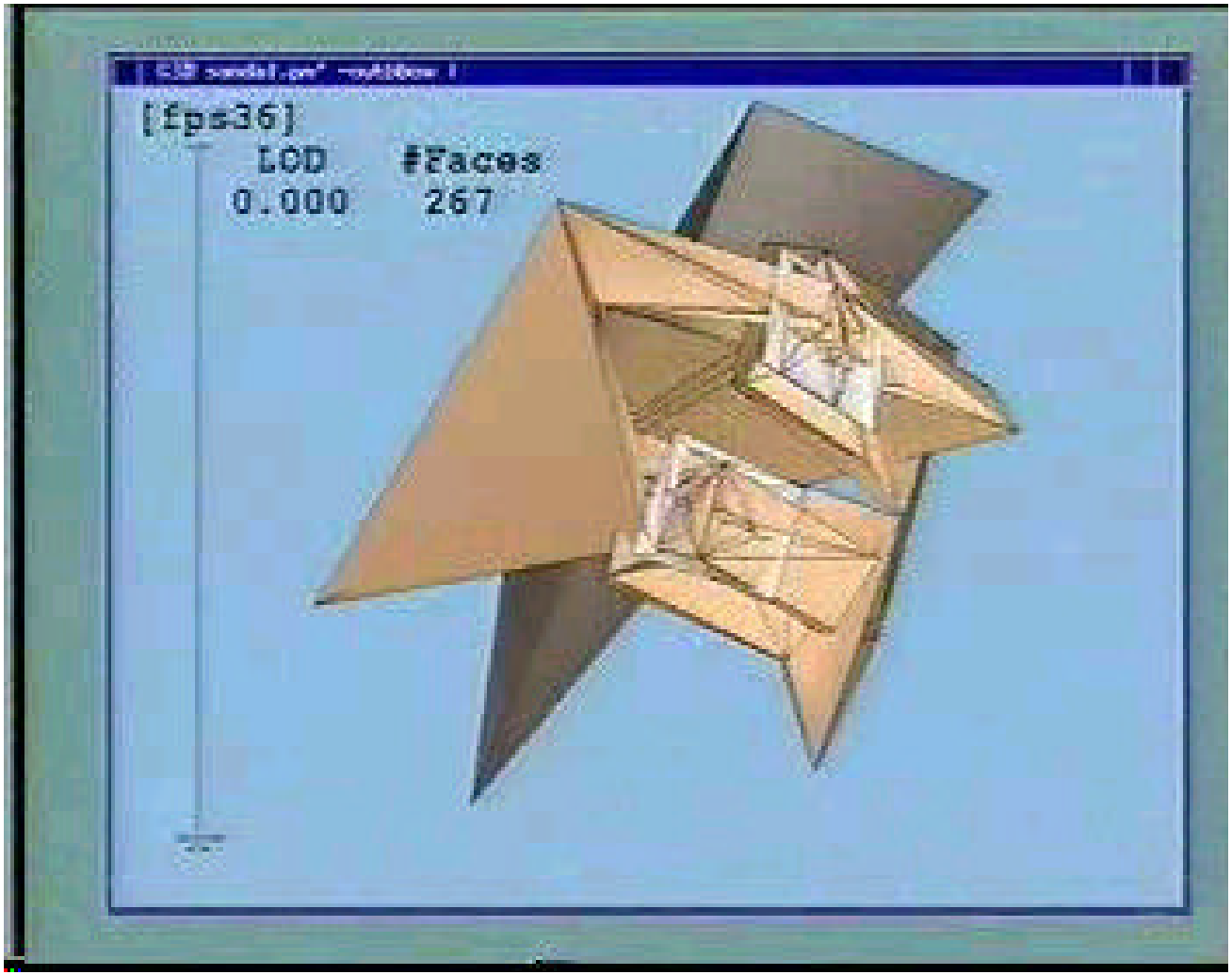
Simplification

- optimal placement strategy
  - Examine a wider range of possibilities
  - Hoppe (1996)
  - u, v both move to some location (midpoint)
  - Garland and Heckbert(1998)
  - Solve a quadratic equation to find an optimal position
  - Higher quality
  - Extra processing (wider range)

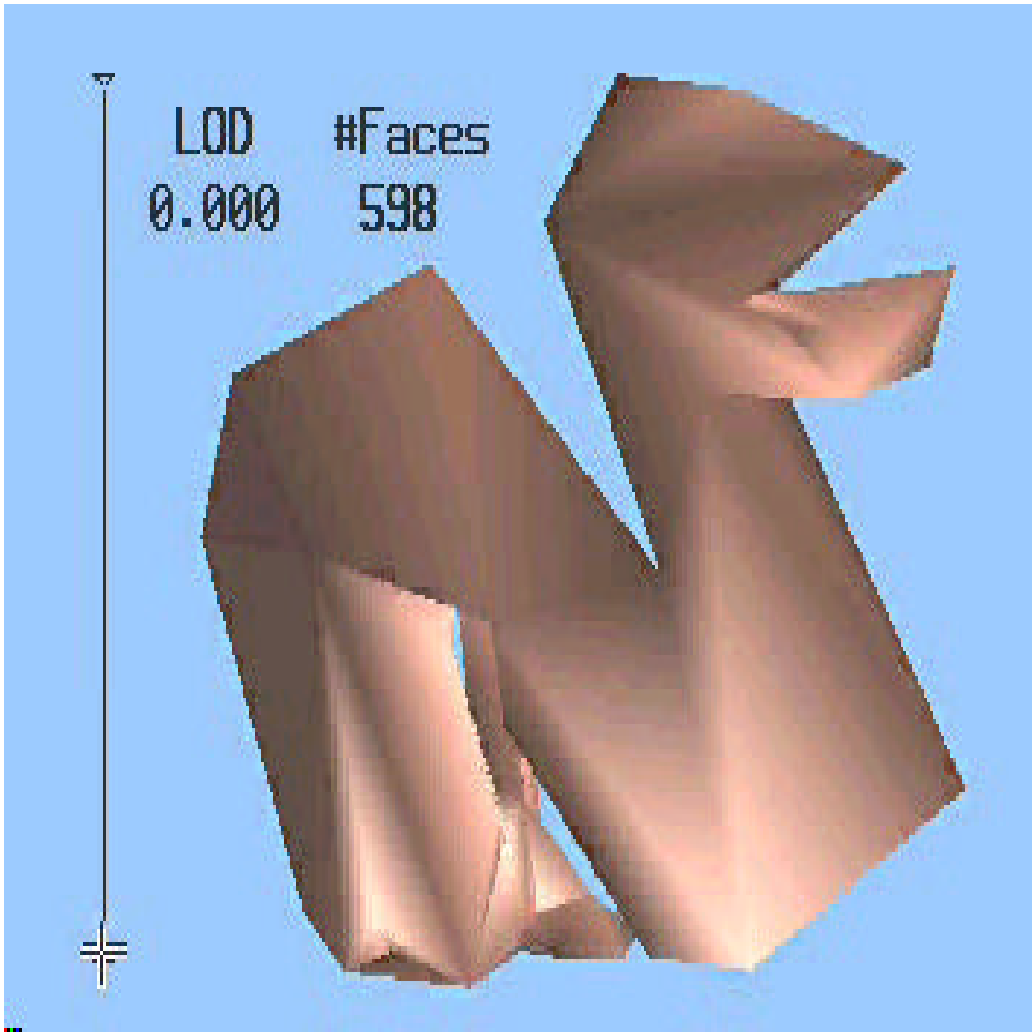
Simplification



Simplification



Simplification



Simplification

- Cost Function

  - edge collapse is analyzed
  - the smallest cost value is performed next
  - trade off speed, quality, robustness, simplicity..

$$c(v) = \sum_{i=1} (n_i \cdot v + d_i)$$

v : new location m : planes

n : plan's normal d : offset from the origin

Simplification

- Modification of cost function

  - weight value by the areas of the triangles
  - based on maintaining other surface features
    - *boundary curves,*
    - *the locations where there are material changes,*
    - *texture map edges,*
    - *color-per- vertex changes*

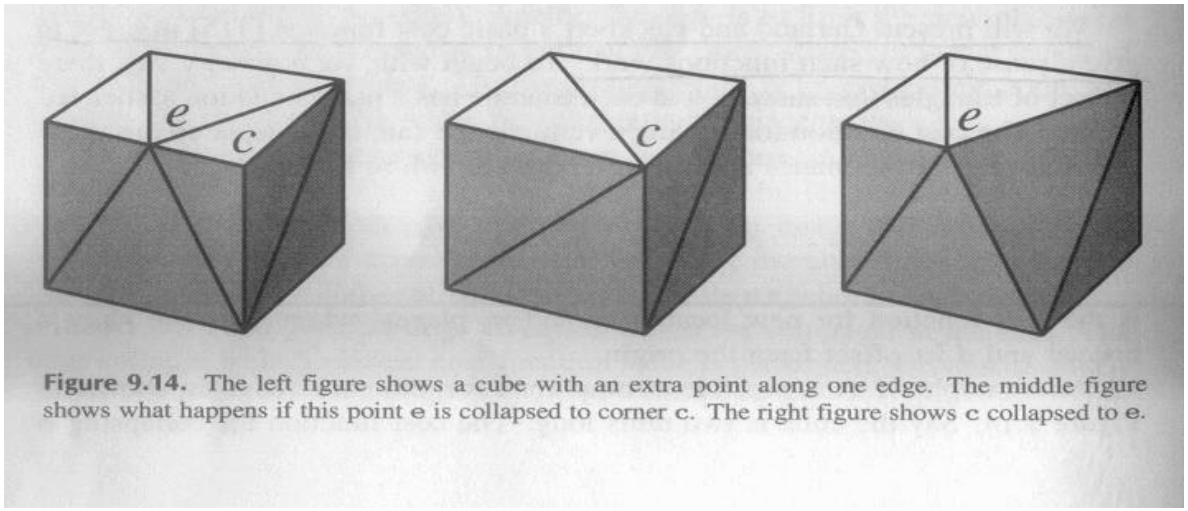


Figure 9.14. The left figure shows a cube with an extra point along one edge. The middle figure shows what happens if this point e is collapsed to corner c. The right figure shows c collapsed to e.

Simplification

- Reversibility

  - storing the edge collapses and reconstruct

  - useful for network transmission

- Produce LOD models

  popping effect

  solution : alpha blending

  geomorphs (increase of decrease LOD)

Simplification

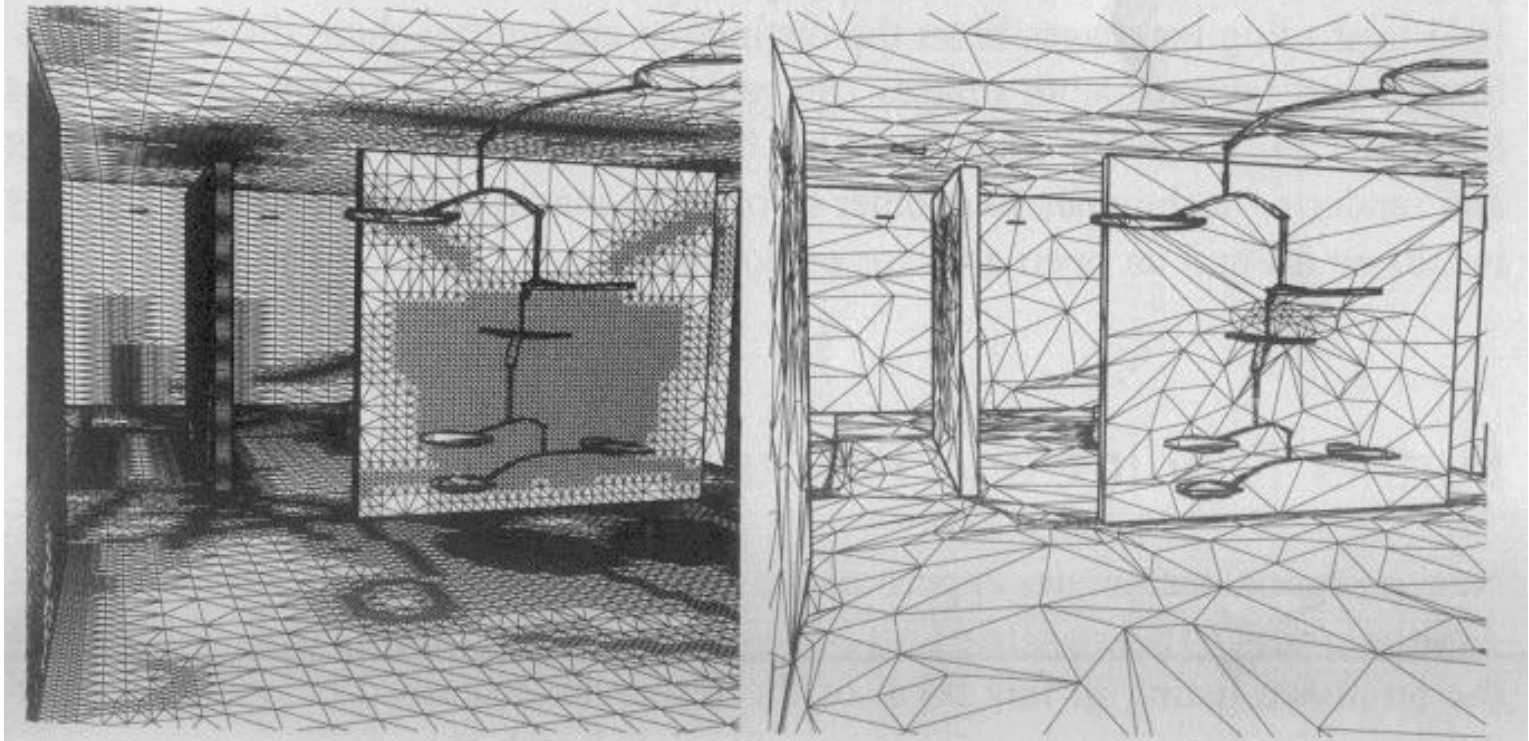
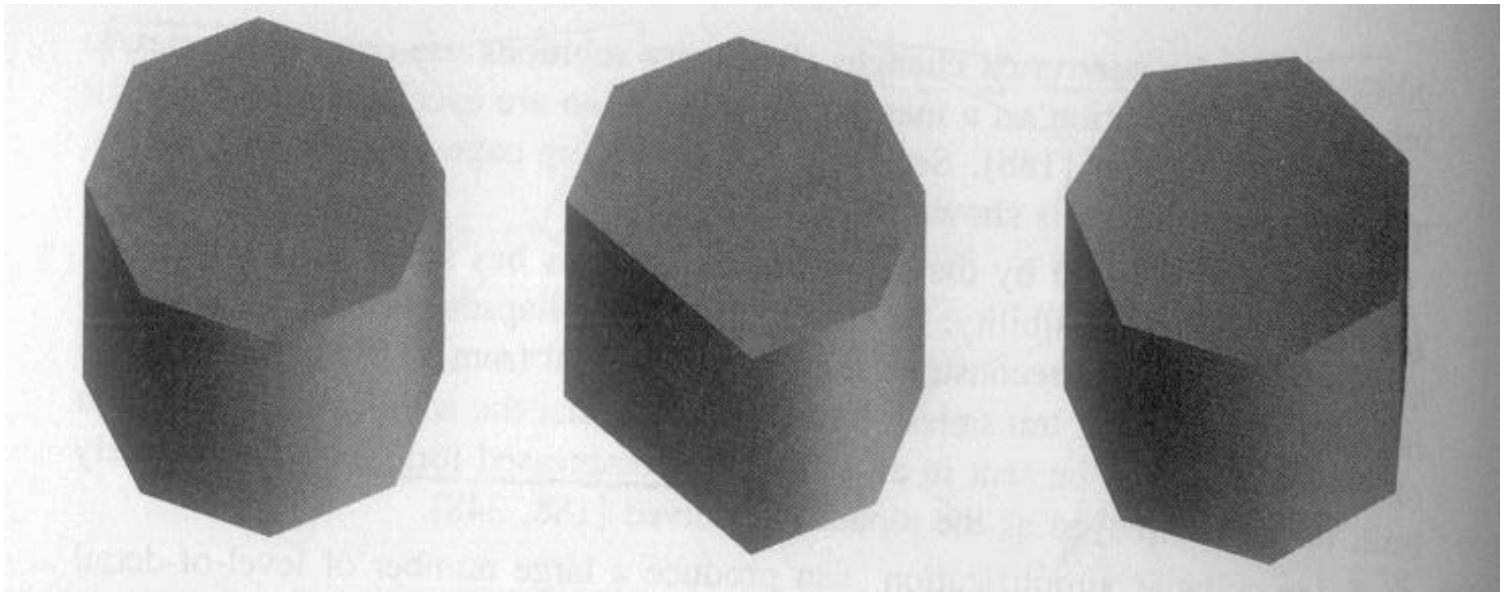


Figure 9.15. Simplification of radiosity solution. Left is the original mesh (150,983 faces), right is simplified mesh (10,000 faces). *(Courtesy of Hugues Hoppe, Microsoft Research.)*

Simplification

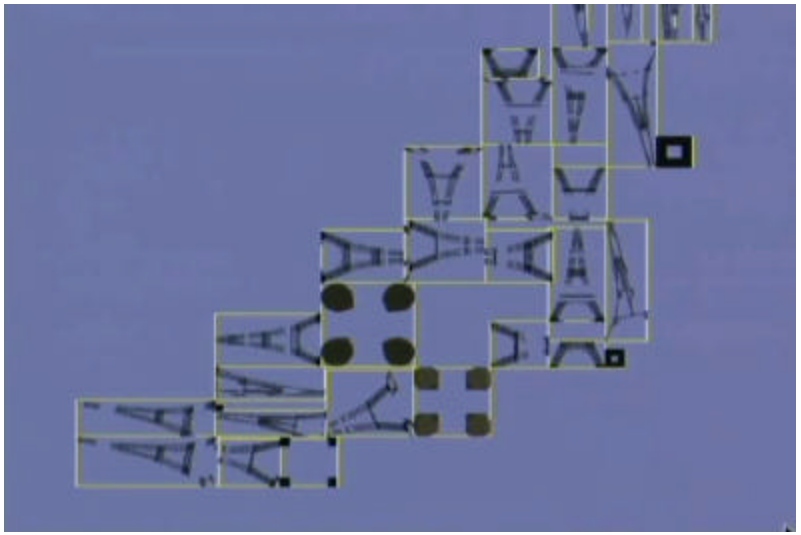
- limitation

  - know nothing about visually important elements, underlying topology or symmetry

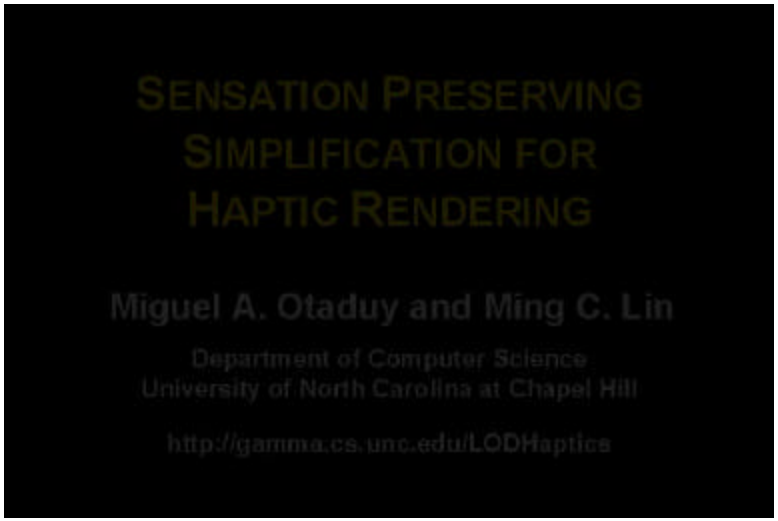


Simplification

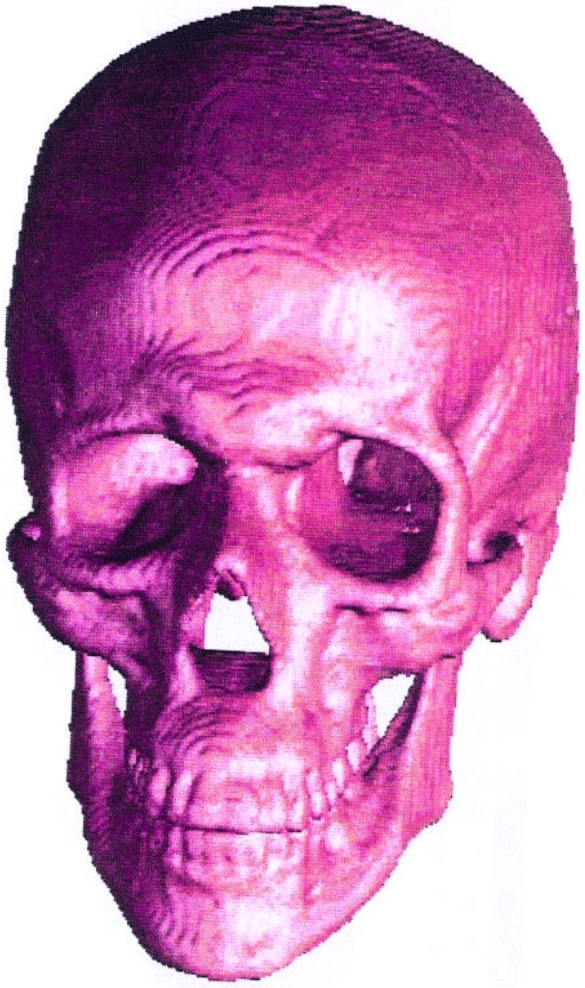
- Share Funny Demos:



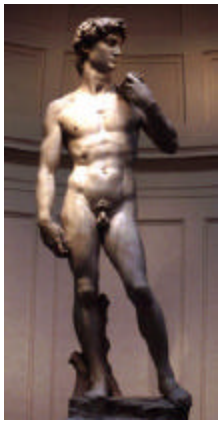
- Online Demo:

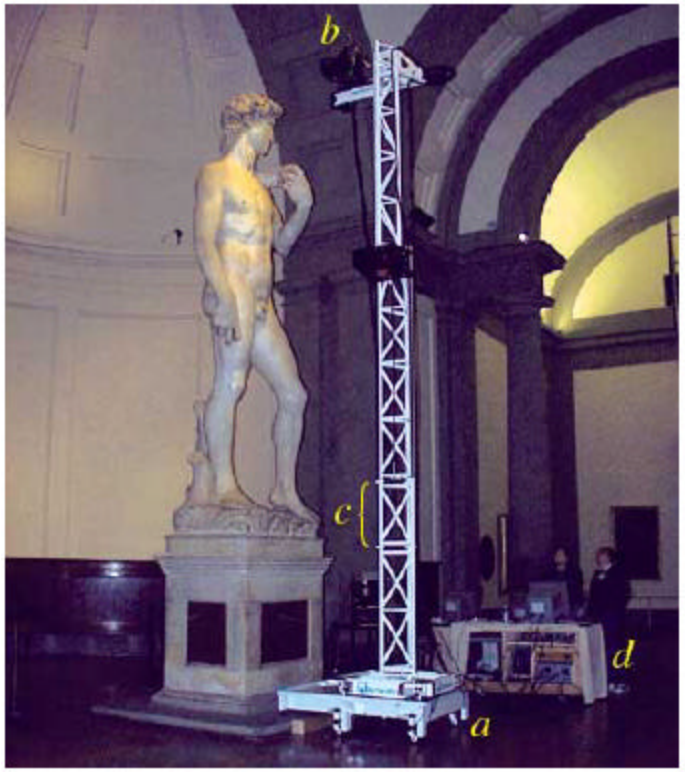


- Polyhedral skull model

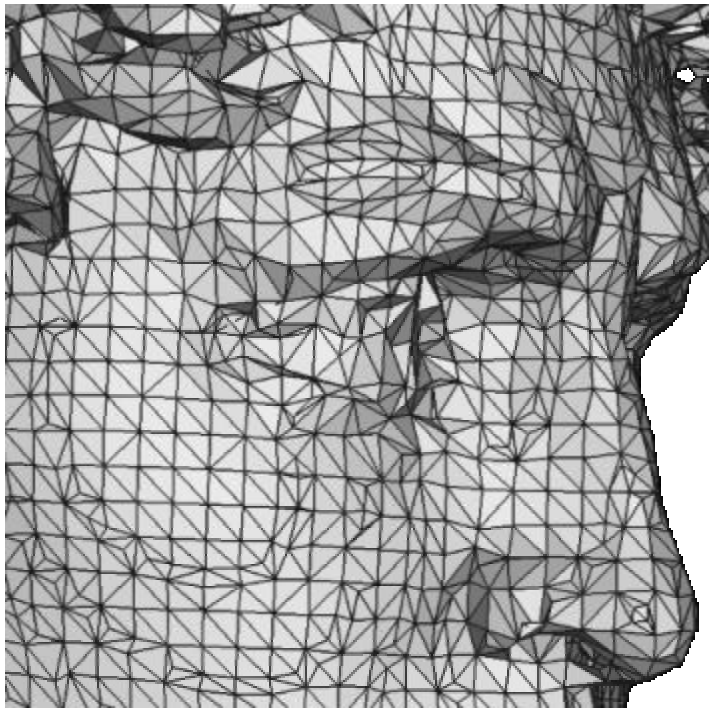


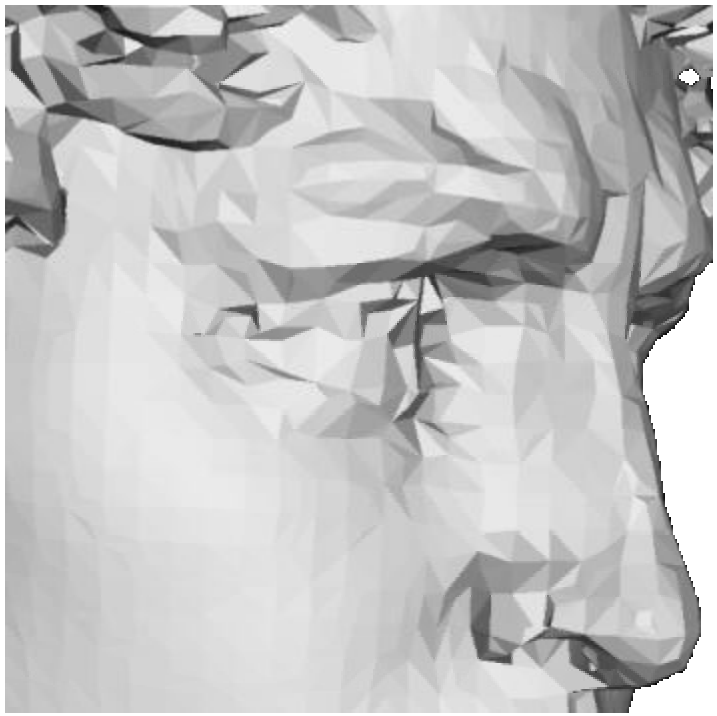
- Laser Scans





- EYE MESH





Simplification

- Share Funny Demos:

Original
(372 million triangles)

1 million triangles

100,000 triangles

10,000 triangles

Simplification

- Share Funny Demos:

Original
(28 million faces)

Phase I: 200x115x344
(322,437 faces)

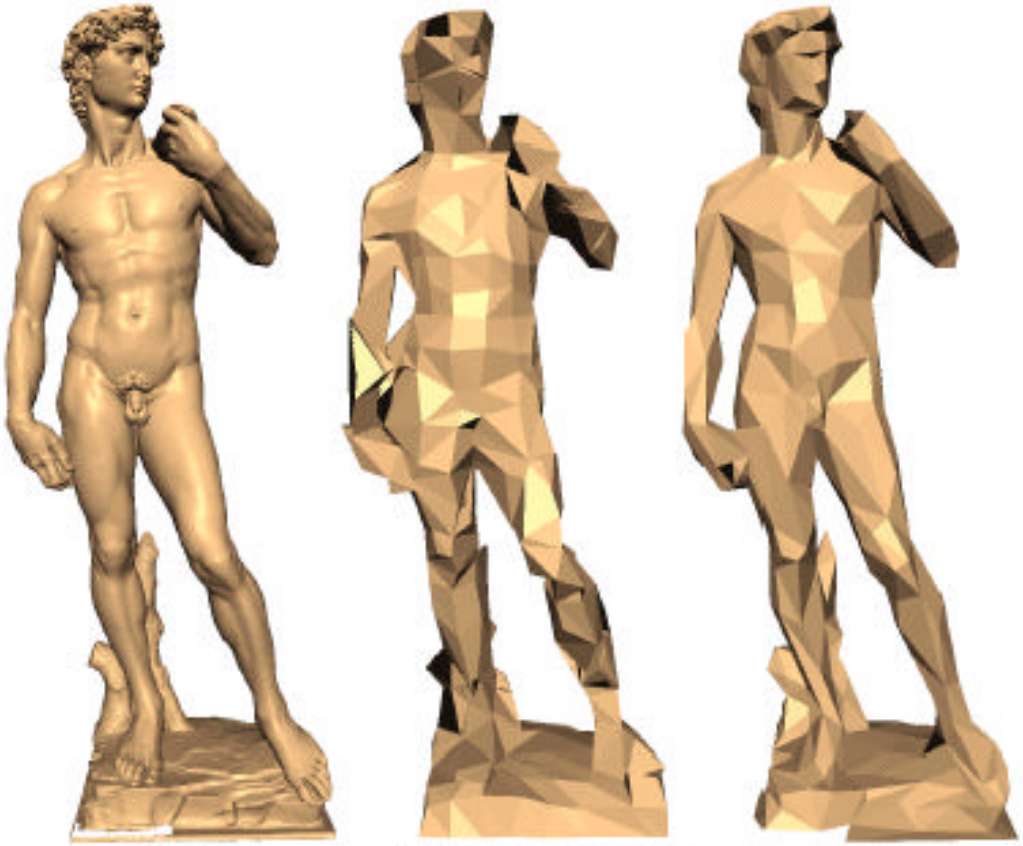
Phase I: 100x57x172
(81,547 faces)

Phase I: 50x29x86
(20,127 faces)

Phase I: 25x14x43
(4511 faces)

Simplification

- Share Funny Demos:



Original
(8 million triangles)

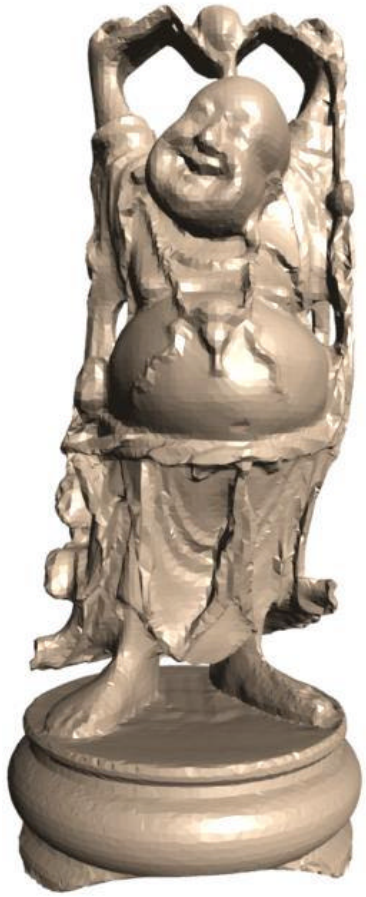
Uniform clustering
(1157 triangles)

Multiphase
(1000 triangles)

Simplification

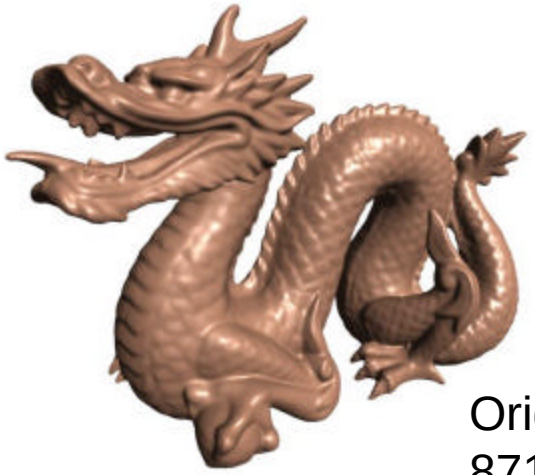
Original Buddha
1,087,716 triangles

OoCS
204,750 triangles

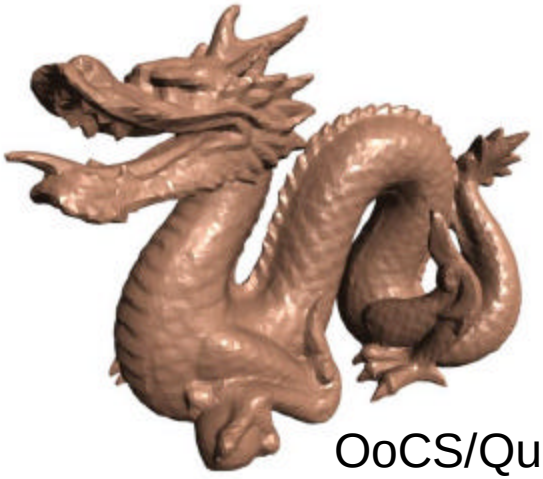


OoCS
62,354 triangles

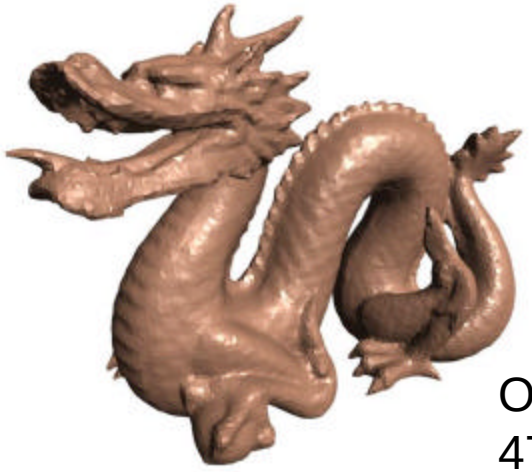
Simplification



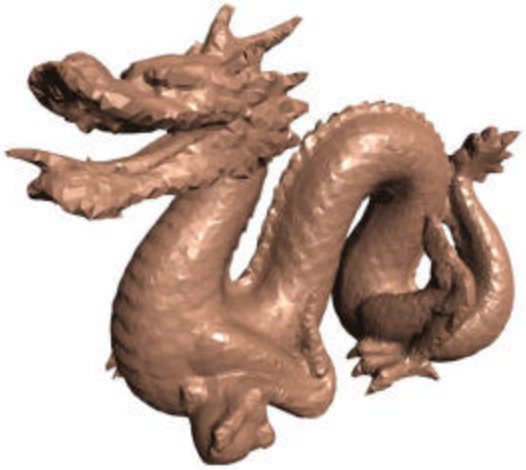
Original dragon
871,306 triangles



OoCS/Quadrics
47,228 triangles

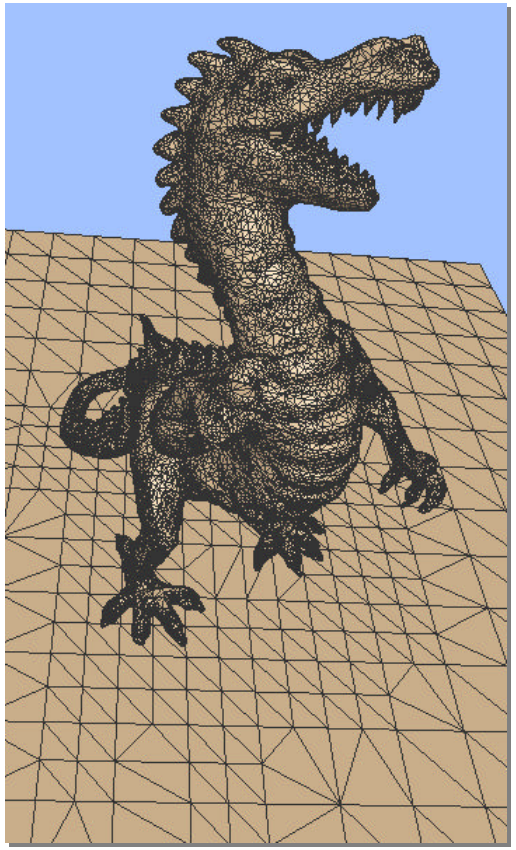


OoCS/Vertex mean
47,228 triangles



OoCS/Vertex grading
47,228 triangles

Simplification



Mesh for solution

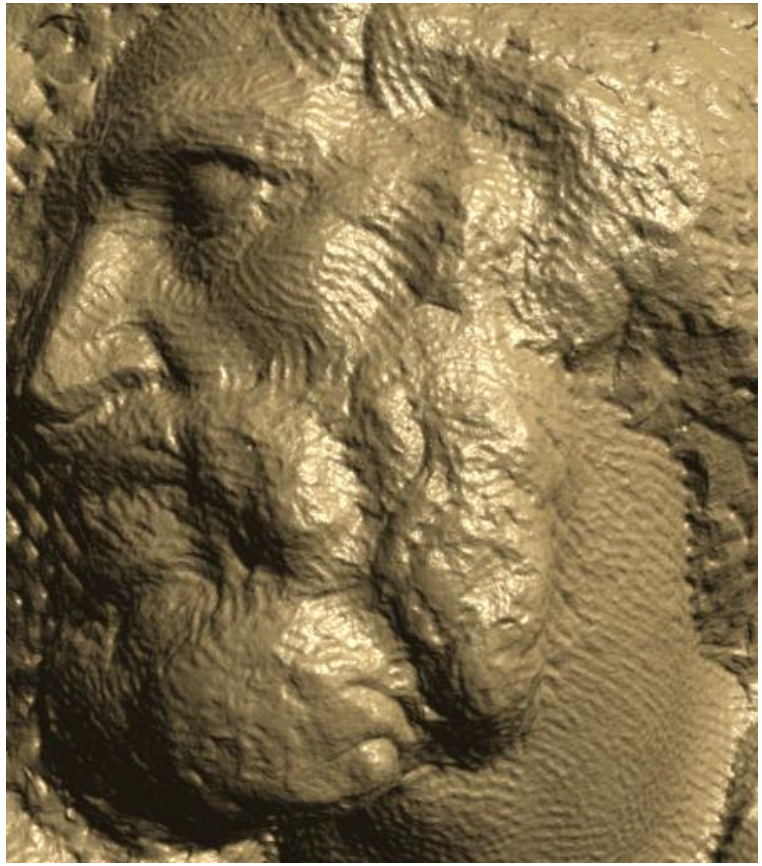
Radiosity solution

Simplification

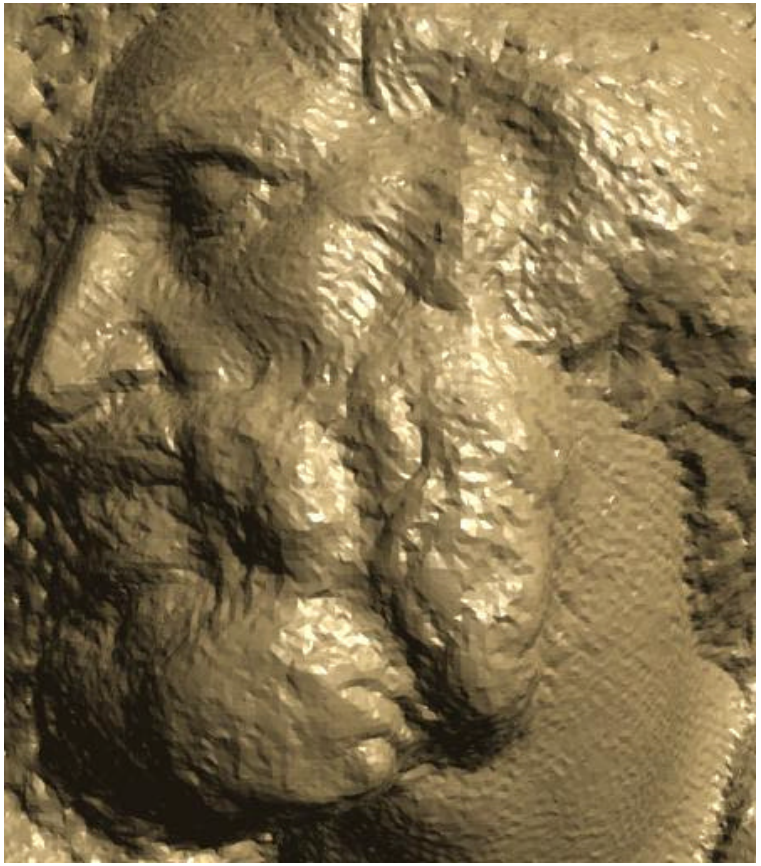
50,761 faces

10,000 faces

Simplification



Original statue
386,488,573 triangles



OoCS
3,122,226 triangles

References

- Francine Evans, Steven Skiena *...,* Optimizing Triangle Strips for Fast Rendering*. IEEE Visualization '96*
- Totally Tessellated - ThinkQuest winner - great site, instruction, information. http://library.advanced.org/16661/
- Tessellations Tutorials - Math Forum site http://forum.swarthmore.edu/sum95/suzanne/tess.intro.html - site for construction of tessellations. http://forum.swarthmore.edu/sum95/suzanne/links.html - great list of tessellation links
- Michael Garland Eric Shaffer: A Multiphase Approach to Efficient Surface Simplification in Proceedings of IEEE VIS 2002, pages 117-124. October 2002