3D Model Simplification
(SENG 463 - Game Programming)

Dr.Çağatay ÜNDEĞER

Research and Innovation Director

SimBT Inc.

e-mail :

cagatay.undeger@simbt.com.tr

cagatay@undeger.com

Outline

- Introduction to Simplification
- Simplification Algorithm Selection Criteria
- Kinds of Simplification Algorithms
- Brief Description of Simplification Algorithms

Polygonal Models

- Currently dominate interactive computer graphics because of
  - Their mathematical simplicity and
  - Their simple, regular rendering algorithms that embed well in hardware.

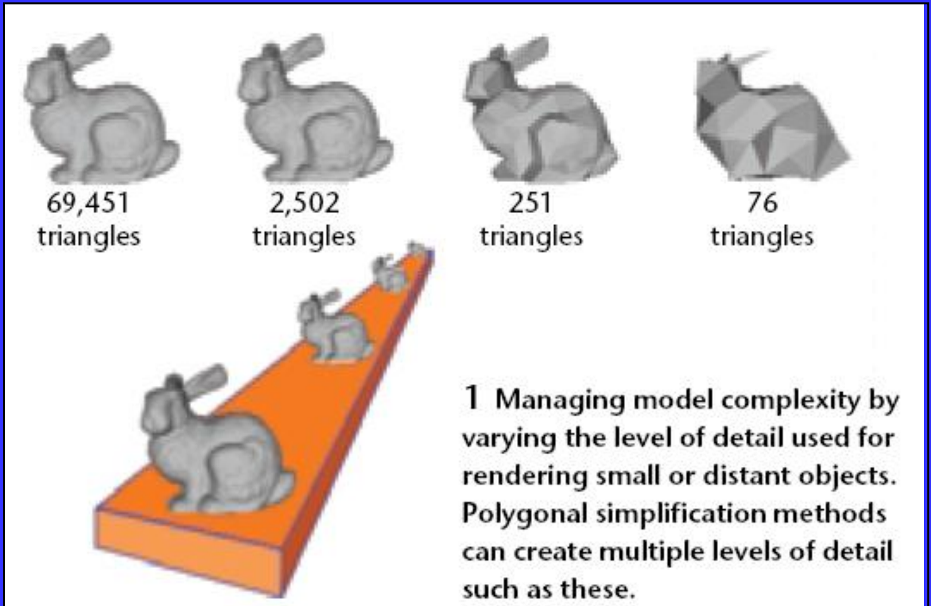
Difficulties of Polygonal Models

- Complexity of polygonal models are measured by the number of polygons / triangles

- Complexity of required models seems to grow faster than the ability of our graphics hardware to render them interactively.

Hand-Crafted Simplification

- Developers prepare Level of Detail (LOD) models manually (hand-crafted).
- Switch among models using distance to camera
- In Unity, you can use these LOD models using LOD Group component.



1 Managing model complexity by varying the level of detail used for rendering small or distant objects. Polygonal simplification methods can create multiple levels of detail such as these.

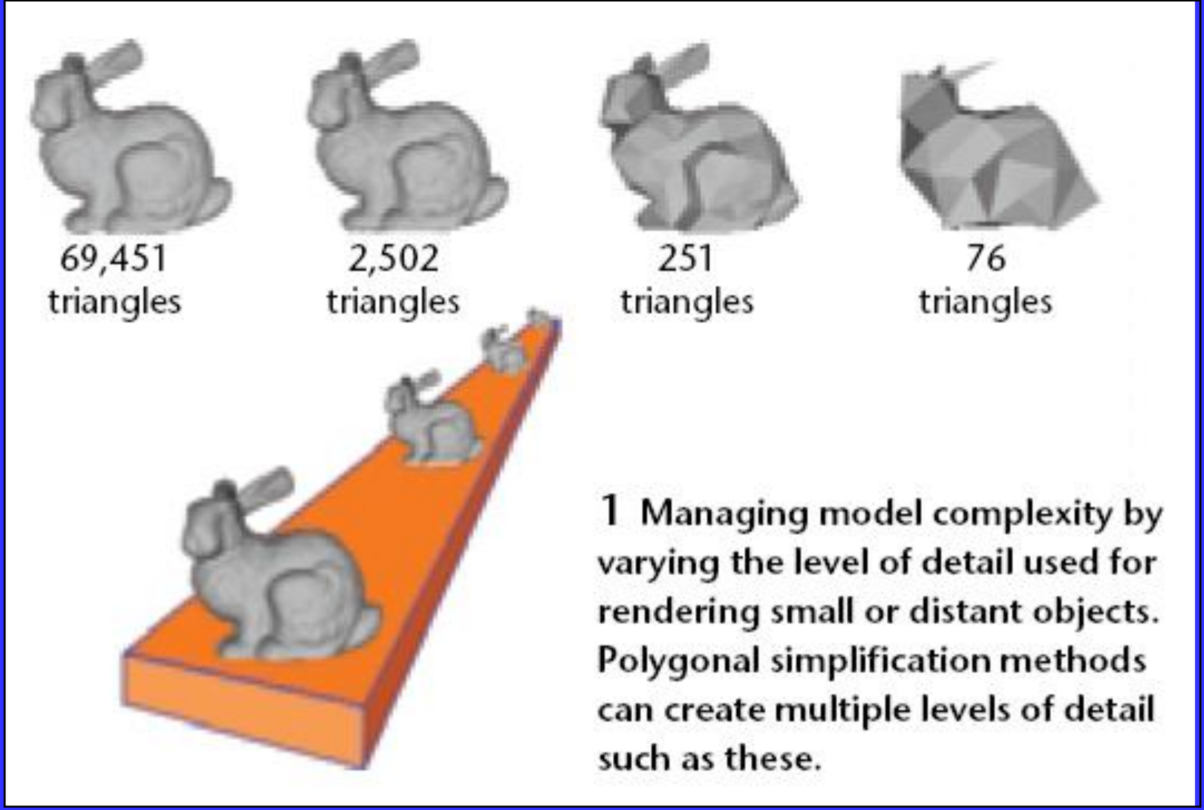
Polygonal Simplification

- Hand-crafted multi-resolution airplane models mostly used with flight simulators in the past
- To guarantee a constant frame rate.

- In this course, we will deal with automated simplification of these models.

Automated Polygonal Simplification

- Offers one solution for developers to deal with these complex models.



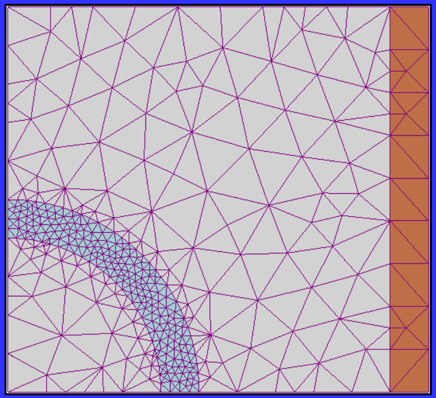
1 Managing model complexity by varying the level of detail used for rendering small or distant objects. Polygonal simplification methods can create multiple levels of detail such as these.

Determining Appropriate Algorithm

- Need to answer several questions:
  - Why do I need to simplify models?
  - What do my models seem like?
  - What is important to me the most?

Why do I Need to Simplify?

- Trying to eliminate redundant geometry?
  – e.g. Volumetric iso-surfaces generated by the marching cubes algorithm tile the model’s flat regions with many small, coplanar triangles?
  – After subdivision of a model for increasing detail, a simplification algorithm is required to remove unnecessary geometry?



Why do I Need to Simplify?

- Trying to reduce model size?
  - Geometric compression?
  - Creating downloadable models for a Web site?
  - Minimizing storage requirements?

Why do I Need to Simplify?

- Trying to improve runtime performance?
  - Generating levels of detail (LODs) of the objects in a scene?
  - Simplifying the polygonal scene being rendered,
    - By representing distant objects with a lower LOD and nearby objects with a higher LOD?

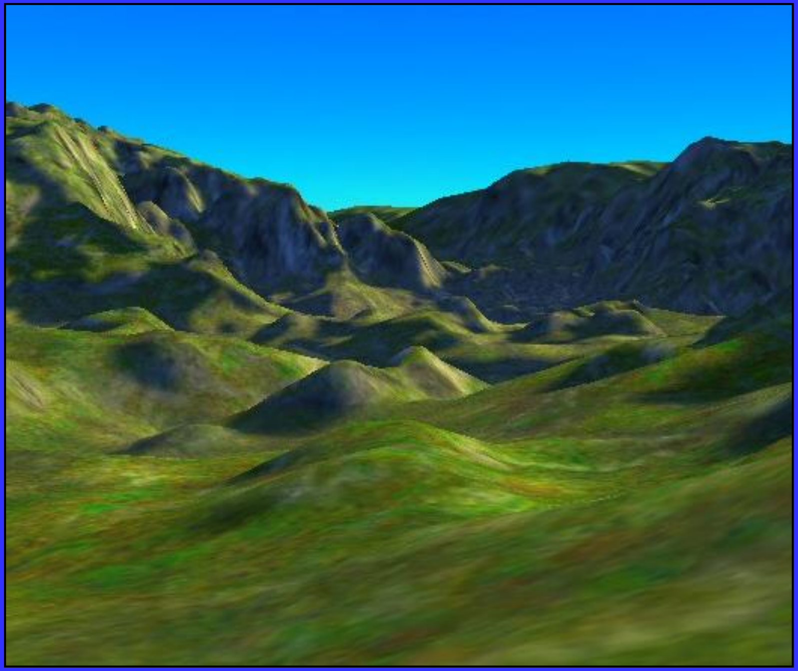
What do My Models Seem Like?

- No algorithm today exists that simplifying all models sucessully.

- Organic forms with smooth curves.
- Mechanical objects with sharp corners, flat faces, and regular curves.
- Precomputed colors, lighting or texture that must be considered.

What do My Models Seem Like?

- A few large, high-complexity, individual objects (e.g. terrain data, medical data).
- Multiple objects of moderate complexity.
- Large assemblies of many small objects.

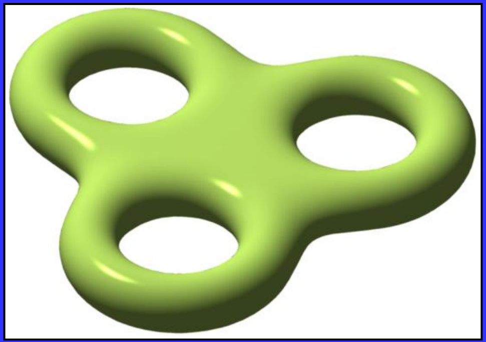


What is Important to Me?

- Preserve accuracy in the simplified models?
  - Control the *Hausdorff distance* of the simplified vertices / surface to the original
  - Bound the volumetric deviation of the simplified mesh from the original...

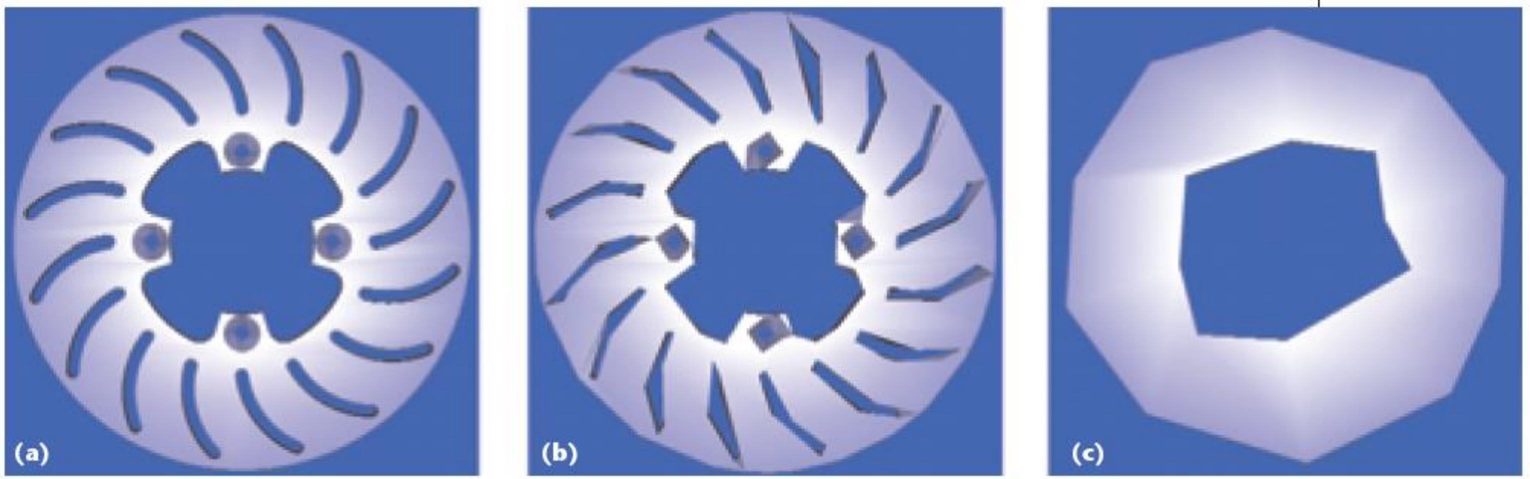
- Just want high visual fidelity?
  - Perception is more difficult to quantify than geometry

- Preserve model's topological genus?



What is Important to Me?

- If run-time performance is crucial,
  - You may need an algorithm capable of drastic simplification.
- A drastic simplification may require:
  - View-depended simplification
  - Topology-reduction simplification



4 Preserving genus limits drastic simplification. The original model of a brake rotor with (a) 4,736 triangles and 21 holes is simplified with a topology-preserving algorithm using (b) 1,006 triangles and 21 holes and a topology-modifying algorithm with (c) 46 triangles and one hole. Model courtesy of the Alpha_1 Project, University of Utah.

Kinds of Algorithms

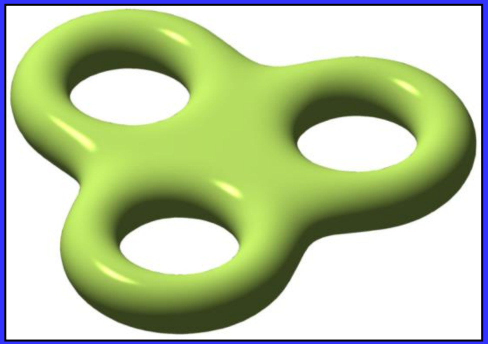
- Many approaches, each with strengths and weaknesses.
  - Treatment of mesh topology
  - Simplification mechanism
  - Static, dynamic, view-depended

Treatment of Mesh Topology

- Provides an important distinction
- Topology: connected polygonal mesh structure
- Genus: number of holes in the mesh surface


Genus of zero

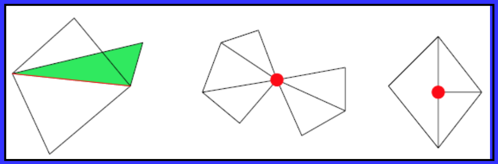

Genus of one

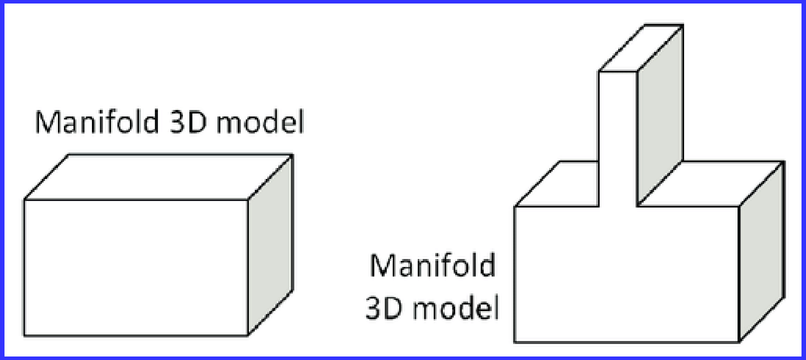


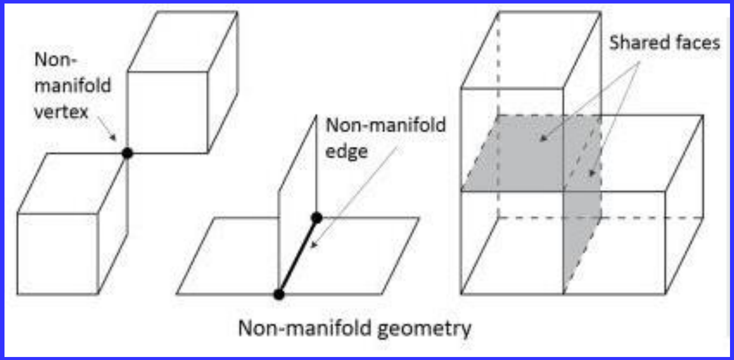
Genus of three

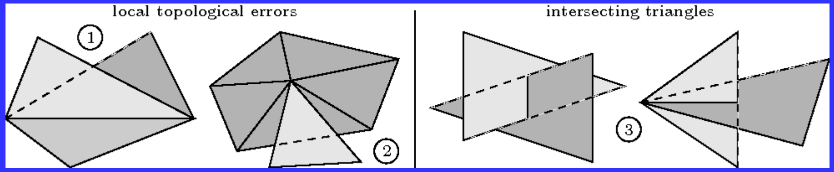
Treatment of Mesh Topology

- In a 3D manifold mesh:
  - Exactly two triangles share a single edge, and
  - Every triangle shares its edges with exactly three neighboring triangles.
  - They are continuous. They wrap without any end or beginning
  - Separate regions shall not connected through a single vertex





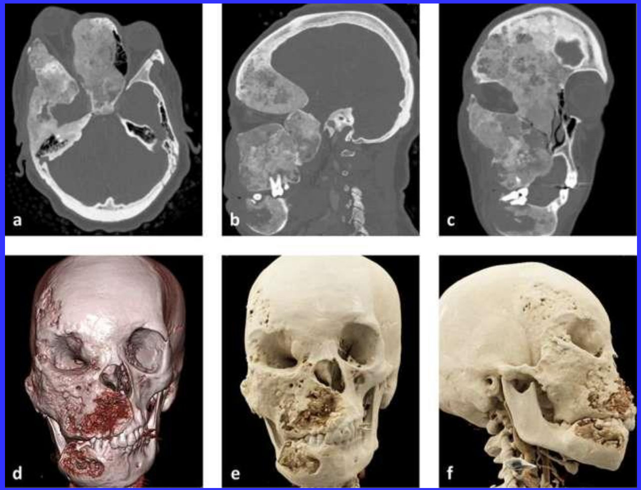


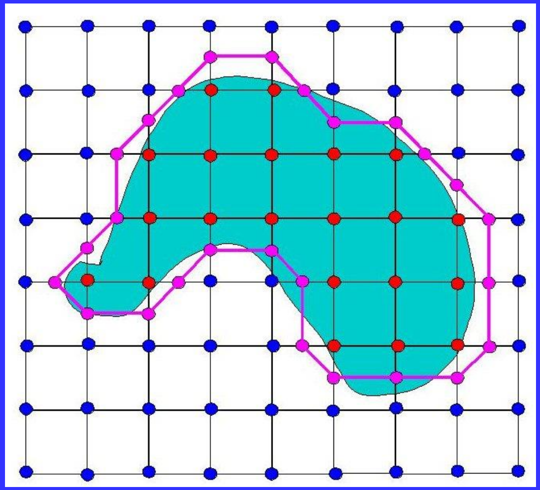
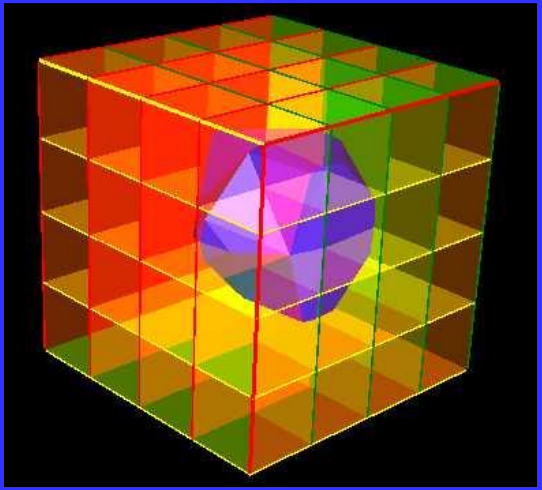


Treatment of Mesh Topology

- Manifold meshes result in well-behaved models.
- Any simplification algorithm can successfully operate on any manifold object.

- Some algorithms (e.g. marching-cubes) guarantee manifold output.



Treatment of Mesh Topology

- A *topology-preserving* simplification algorithm:
  - Preserves manifold connectivity at every step.
  - Fidelity tends to be relatively good.
  - Can’t be simplified drastically.
  - Requires a mesh with manifold topology.

- *A topology tolerant* simplification algorithm:
  - Ignores regions in the mesh with nonmanifold local topology,
  - Leaves those regions unsimplified.

Treatment of Mesh Topology

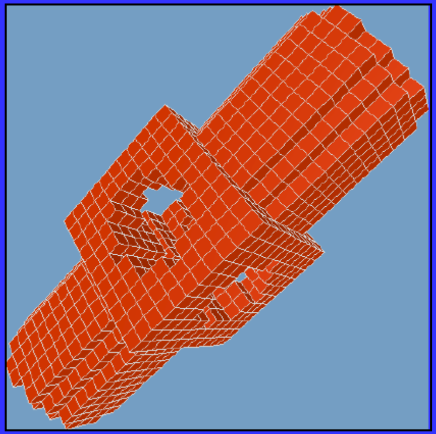
- *A topology-modifying* algorithm:
  - Don’t necessarily preserve manifold topology.
  - May close up holes in the model and aggregate separate objects into assemblies
  - Permitting drastic simplification beyond the scope of topology-preserving schemes.
  - Often provides poor visual fidelity.
  - Good at real-time visualization of complex scenes.

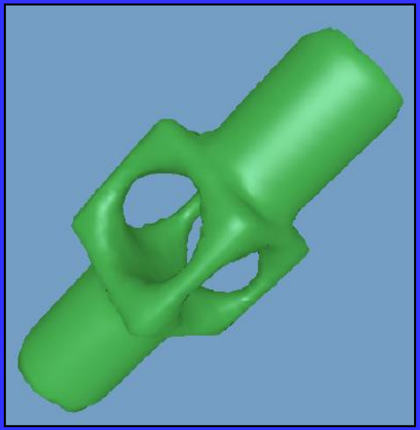
Simplification Mechanism

- Four basic polygon removal mechanisms:
  - Sampling,
  - Decimation,
  - Vertex merging,
  - Adaptive subdivision

Sampling

- Sample the initial model's geometry:
  - With points on the model's surface or
  - With voxels superimposed on the model in a 3D grid.
- More elaborate and difficult to code approaches.
- Trouble achieving high fidelity.
- Work best on smooth organic forms with no sharp corners.



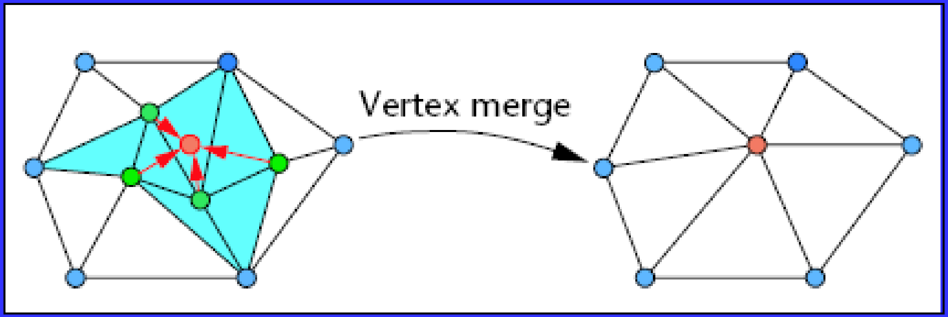


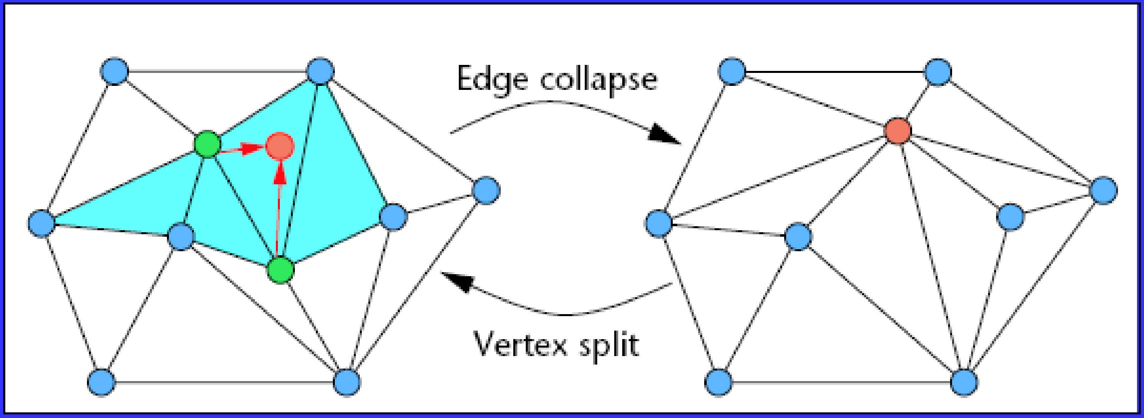
Decimation

- Iteratively remove vertices or faces from the mesh, re-triangulating the resulting hole after each step.
- Relatively simple to code and can be very fast.
- Use strictly local changes that tend to preserve the genus.
- This restricts drastic simplification.

Vertex Merge

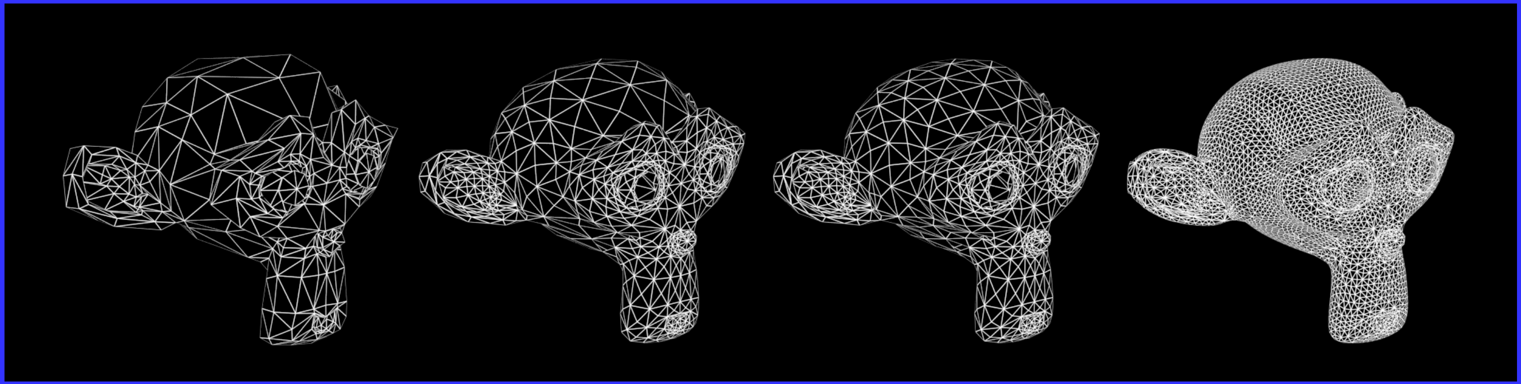
- Collapses two or more vertices of a triangulated model into a single vertex,
  - Which in turn can be merged with other vertices.





Adaptive Subdivision

- Finds a simple base mesh that closely approximate the initial model by recursively subdividing the mesh.
- Requires creating a base model that captures the original model's important features, which can be tricky.
- Preserve the surface topology, which may limit their capacity for drastic simplification.



Static, Dynamic, and View-Dependent Simplification

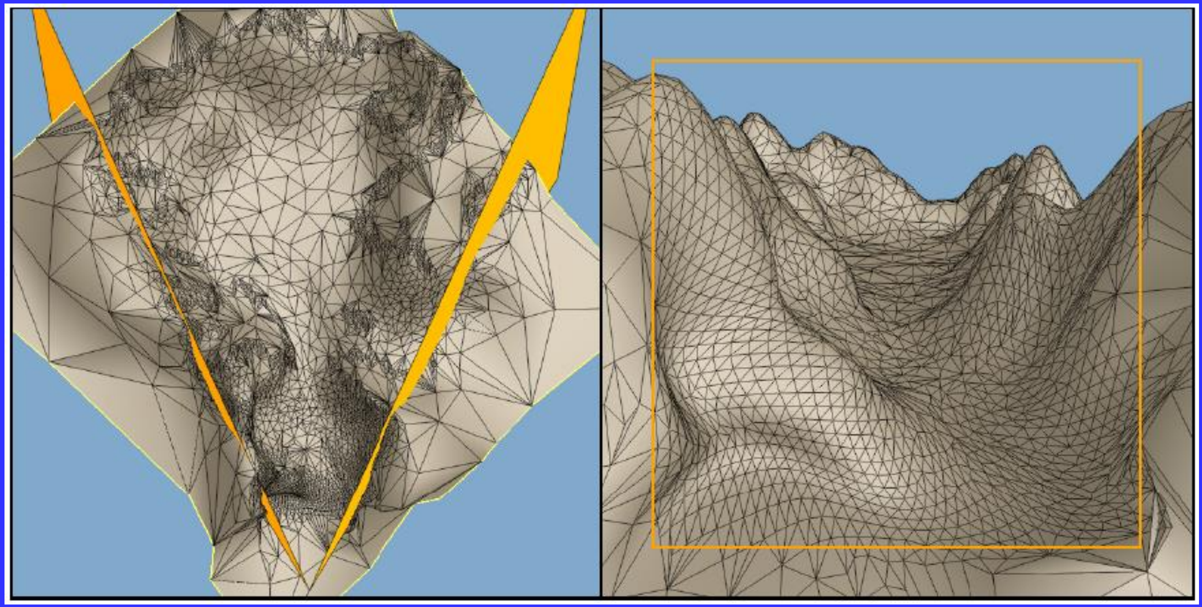
- Traditional static approach:
  - Polygonal simplification creates several discrete versions of each object in a preprocess, each at a different LOD.
  - Decoupling simplification and rendering makes this the simplest model to code.
  - Each LOD can be converted during preprocessing to triangle strips and compiled as a separate display list.

Static, Dynamic, and View-Dependent Simplification

- Dynamic approach:
  - Creates a data structure, which encodes a continuous spectrum of detail.
  - Desired LOD can be extracted from this structure at run-time.
  - Advantage: uses no more polygons than necessary.

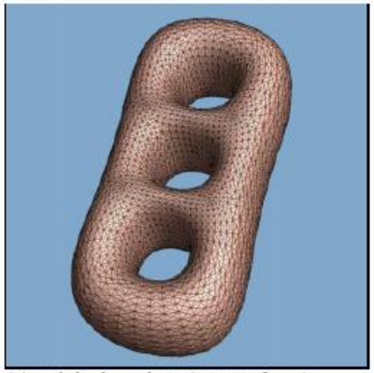
Static, Dynamic, and View-Dependent Simplification

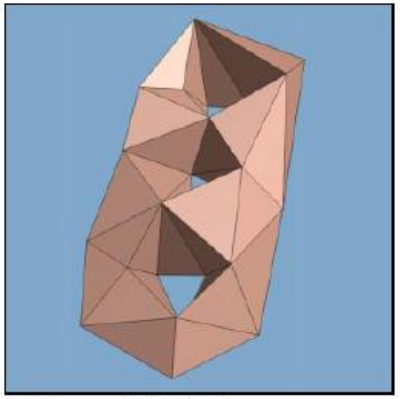
- *View-dependent simplification:*
  - Extends dynamic simplification by using view-dependent criteria
  - Selects the most appropriate LOD for the current view.



Brief Description of Some Simplification Algorithms

- Triangle Mesh Decimation
- Vertex Clustering
- Multiresolution Analysis of Arbitrary Meshes
- Voxel-Based Object Simplification
- Simplification Envelopes
- Appearance-Preserving Simplification
- Quadric-Error Metrics
- Image-Driven Simplification
- Progressive Meshes
- Real-Time Optimally Adapting Meshes (ROAM)


(a) Original mesh M (11,776 faces)


(e) Base mesh (70 faces)

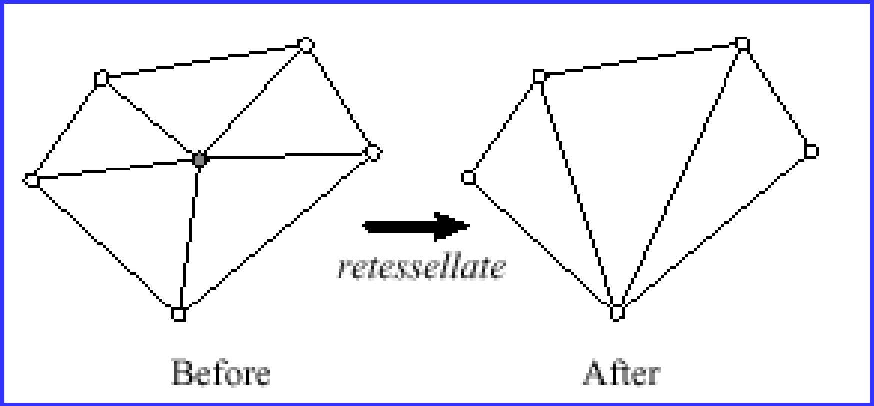
Triangle Mesh Decimation

- Simplify general polygonal models.
- Term *decimation* for iterative removal of vertices.
- Designed to operate on the output of the marching cubes algorithm.
- Since coplanar regions divided into many more polygons than necessary.

By Schroeder, Zarge, and Lorenson

Triangle Mesh Decimation

- Multiple passes over all the vertices:
  - If the vertex can be removed without violating the local topology, and
  - If resulting surface would lie within a user-specified distance of the unsimplified geometry,
  - Deletes vertex and all its associated triangles.
  - Hole in the mesh is re-triangulated.



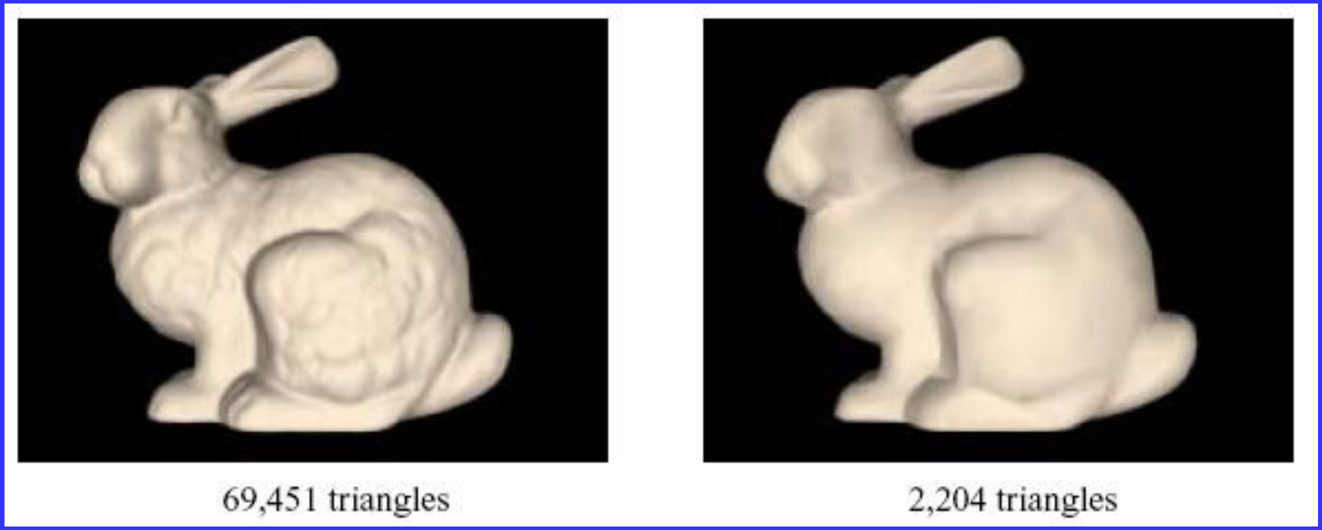
Triangle Mesh Decimation

- Vertices of simplified model are a subset of the original model’s vertices.
- Convenient for reusing normals and texture coordinates at the vertices.
- Limit the fidelity of the simplifications.
- Quite fast and topology tolerant.

- Available as part of the Visualization Tool Kit at http://www.kitware.com/vtk.html

Appearance-Preserving Simplification

- Takes the error-bounding approach with respect to the rendered view.
- Provides the best guarantees on fidelity of any simplification algorithm.



By Cohen, Olano, and Manocha

Appearance-Preserving Simplification

- Fidelity is expressed in terms of maximum screenspace deviation.
- Rendered image deviate from the original’s appearance by no more than a user-specified number of pixels.

Appearance-Preserving Simplification

Figure 6: Armadillo model: 249,924 triangles

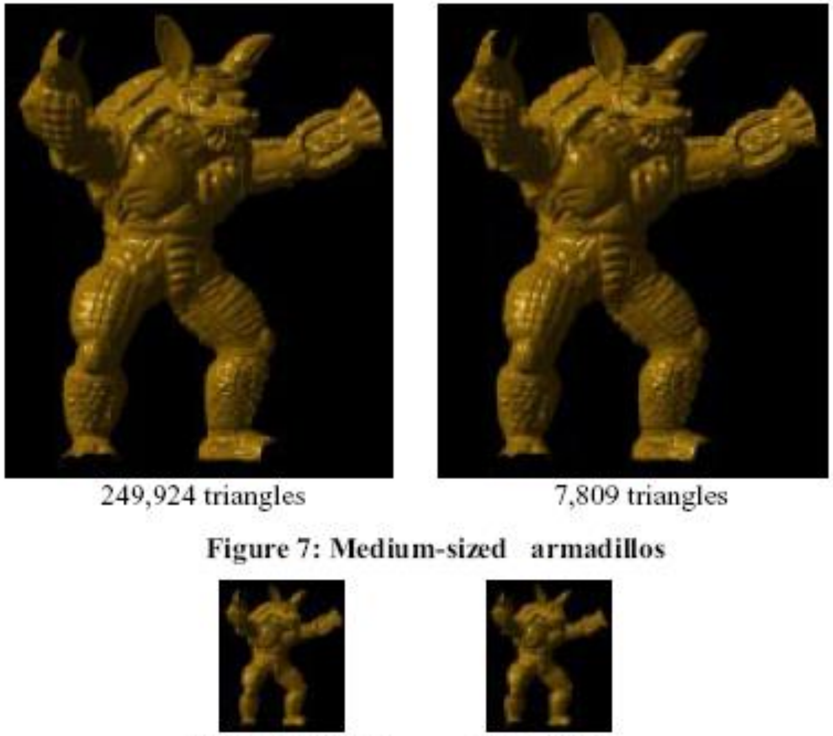


Figure 7: Medium-sized armadillos

Figure 8: Small-sized armadillos

Level of Detail In Unity