CS 378: Computer Game Technology

LOD Meshes

Spring 2012

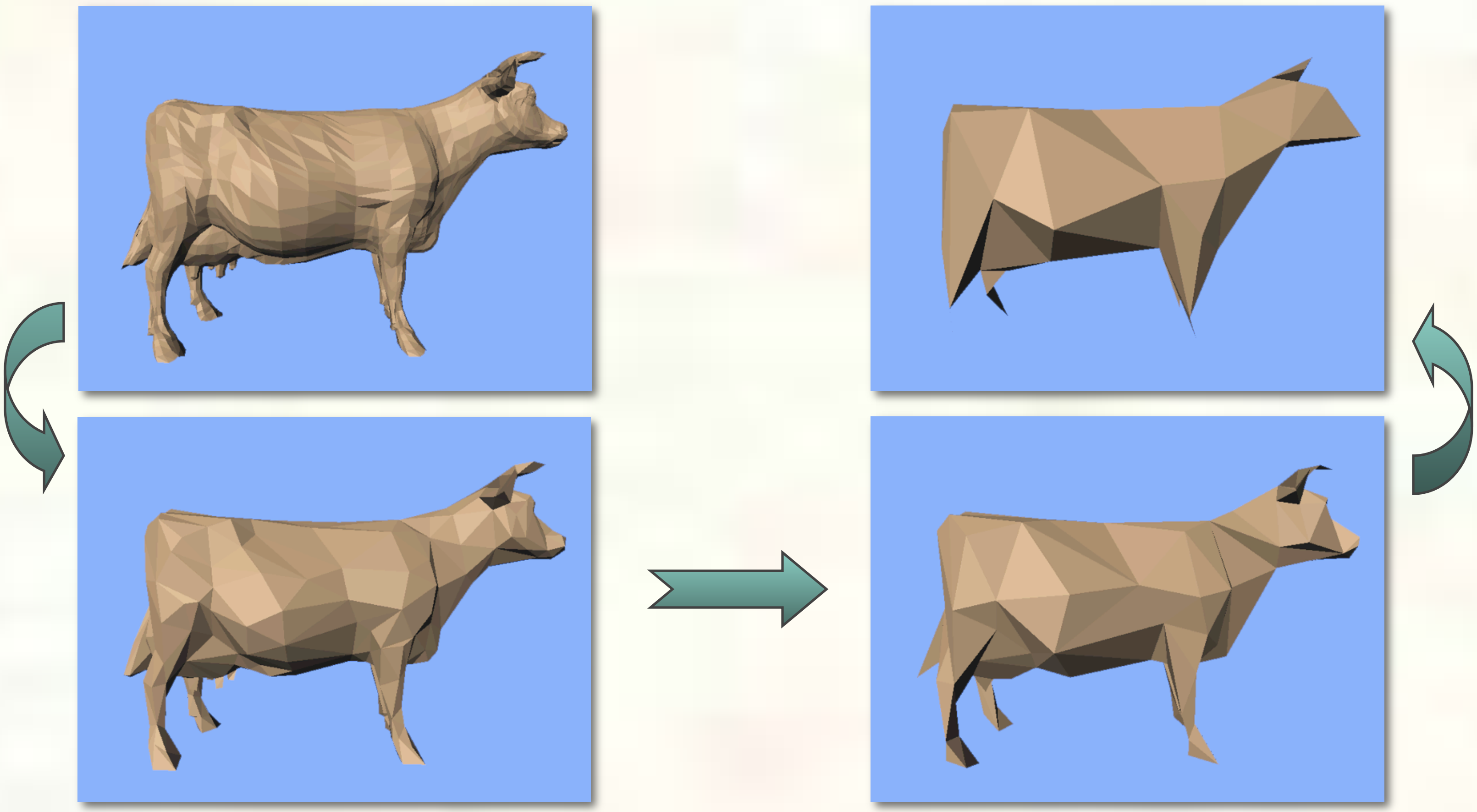
Today

- Level of Detail Overview
- Decimation Algorithms
- LOD Switching

Level of Detail (LOD)

- When an object is close to the viewer, we want a high resolution model
  - Maybe thousands of polygons, high-res textures
- When an object is a long way away, it maps to relatively few screen pixels, so we want a low resolution model
  - Maybe only a single polygon, or tens of polygons
- *Level of Detail* (LOD) techniques draw models at different resolutions depending on their relevance to the viewer
  - Covers both the generation of the models and how they are displayed
  - Deep topic, this is just an overview

LOD Models (I)



LOD Models (II)

Standard LOD

- The current standard for LOD in games is to use a finite set of models
  - Typically aim for models with n, n/2, n/4, … polygons
  - Models may be hand generated or come from automatic *decimation* of the *base mesh* (highest resolution)
- Models are switched based on the distance from the object to the viewer, or projected screen size
  - Better metric is *visual importance*, but it is harder to implement
    - For example, objects that the viewer is likely to be looking at have higher resolution, objects at periphery have lower resolution

Mesh Decimation

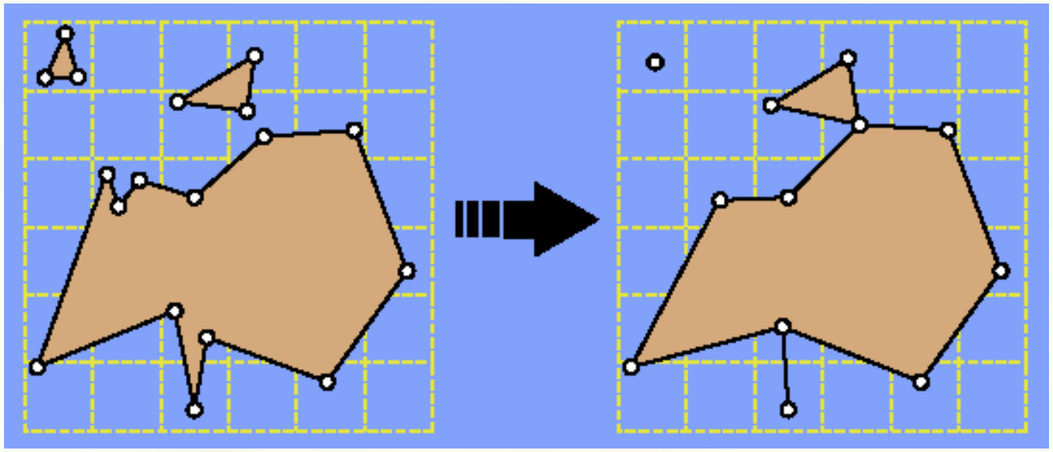
- Take a high resolution *base mesh* and approximate it while retaining its visual appearance
- Desirable properties:
  - Fast (although not real-time)
  - Generates “good” approximations in some sense
  - Handles a wide variety of input meshes
  - Allows for *geomorphs* - geometric blending from one mesh to another
- Two essential questions:
  - What operations are used to reduce the mesh complexity?
  - How is “good” measured and used to guide the approximation?

Mesh Operations

- Operations seek to eliminate triangles while maintaining appearance. Key questions:
  - What does the operation do?
  - Can it connect previously unconnected parts of the mesh?
    - Does it change the mesh topology?
  - How much does it assume about the mesh structure?
    - Must the mesh be *manifold* (locally isomorphic to a disc)?
    - Can it handle, or does it produce, degenerate meshes?
  - Can it be construed as a morph?
    - Can it be animated smoothly?

Vertex Clustering

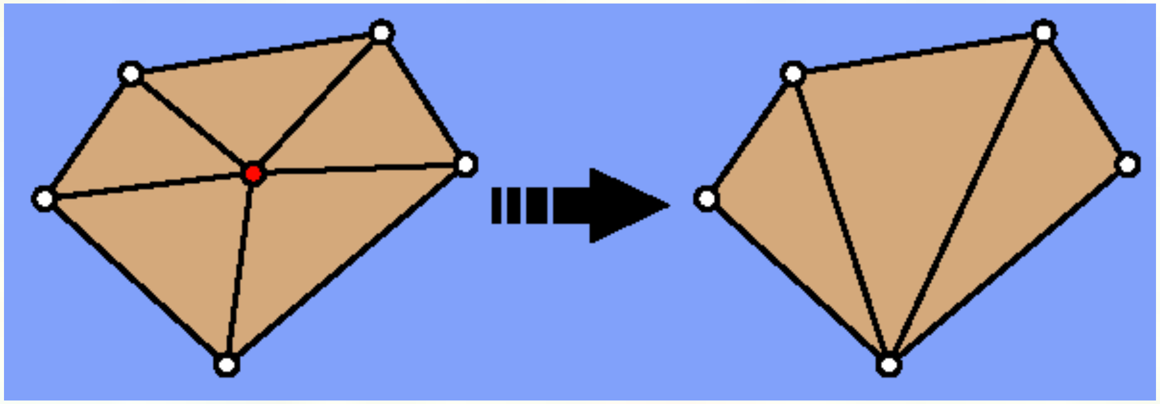
- Partition space into cells
  - grids [Rossignac-Borrel], spheres [Low-Tan], octrees, ...
- Merge all vertices within the same cell
  - triangles with multiple corners in one cell will degenerate



Slide courtesy Michael Garland, http://graphics.cs.uiuc.edu/~garland

Vertex Decimation

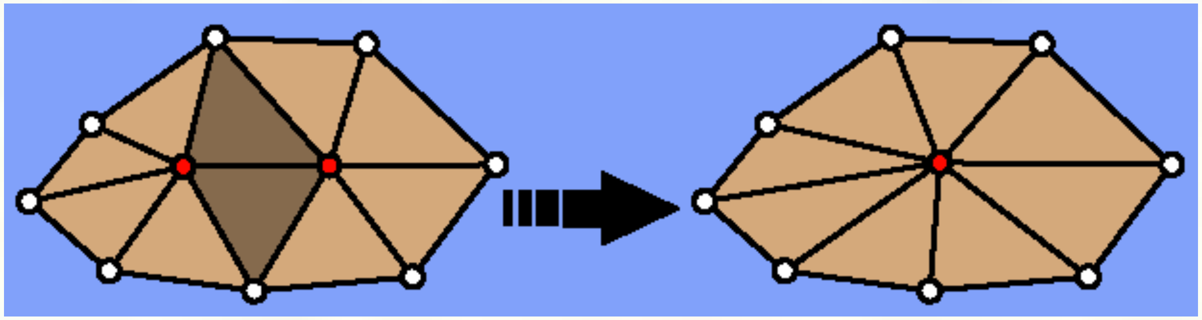
- Starting with original model, iteratively
  - rank vertices according to their importance
  - select unimportant vertex, remove it, retriangulate hole
- A fairly common technique
  - Schroeder *et al*, Soucy *et al*, Klein *et al*, Ciampalini *et al*



Slide courtesy Michael Garland, http://graphics.cs.uiuc.edu/~garland

Iterative Contraction

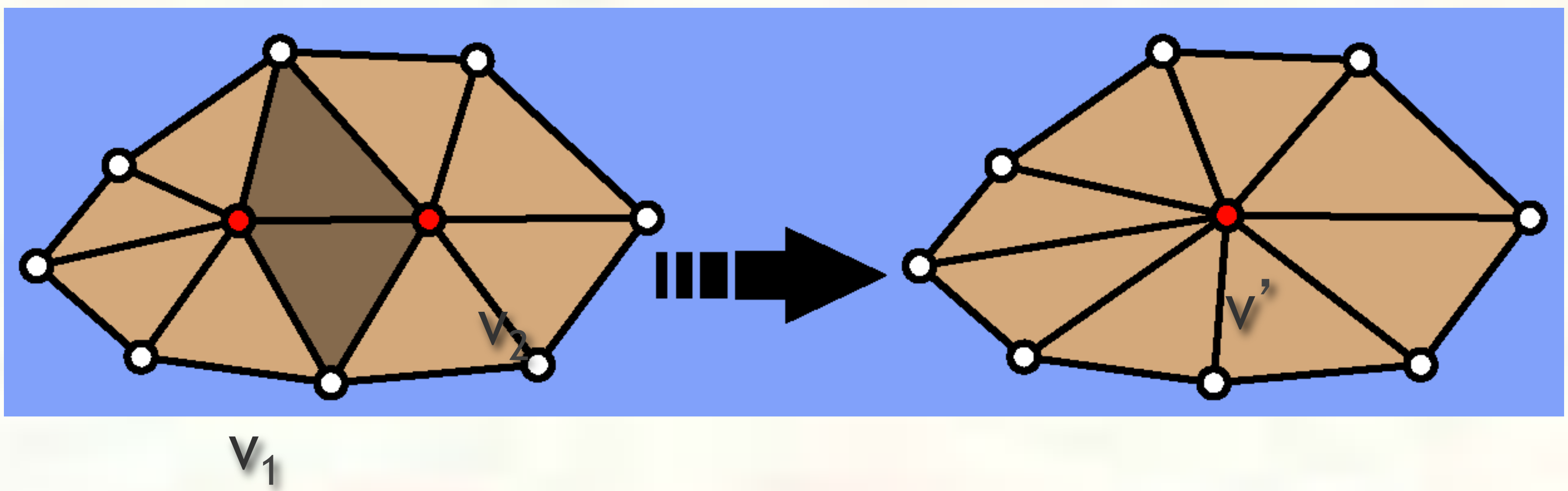
- Contraction can operate on any set of vertices
  - edges (or vertex pairs) are most common, faces also used
- Starting with the original model, iteratively
  - rank all edges with some cost metric
  - contract minimum cost edge
  - update edge costs



Slide courtesy Michael Garland, http://graphics.cs.uiuc.edu/~garland

Edge Contraction

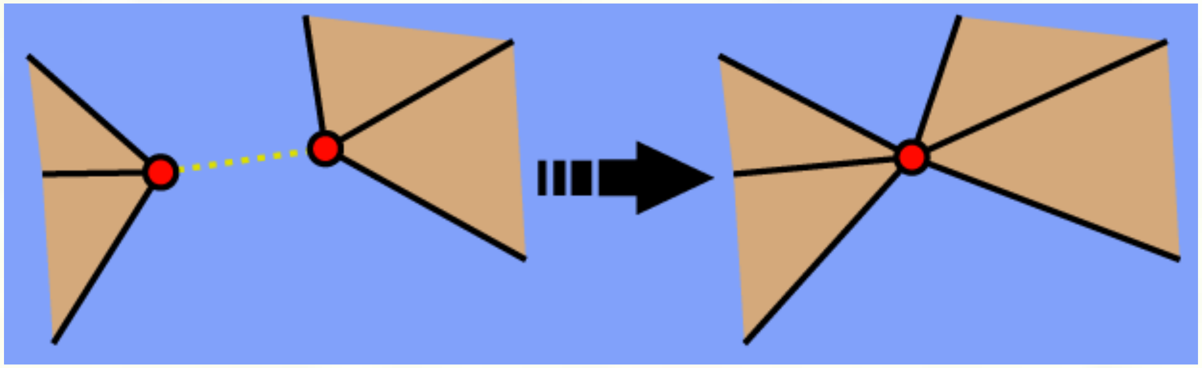
- Single edge contraction $(v_1,v_2) \rightarrow v'$ is performed by
  - moving v_1 and v_2 to position v'
  - replacing all occurrences of v_2 with v_1
  - removing v_2 and all degenerate triangles



Slide courtesy Michael Garland, http://graphics.cs.uiuc.edu/~garland

Vertex Pair Contraction

- Can also easily contract any pair of vertices
  - fundamental operation is exactly the same
  - joins previously unconnected areas
  - can be used to achieve topological simplification



Slide courtesy Michael Garland, http://graphics.cs.uiuc.edu/~garland

Operations to Algorithms

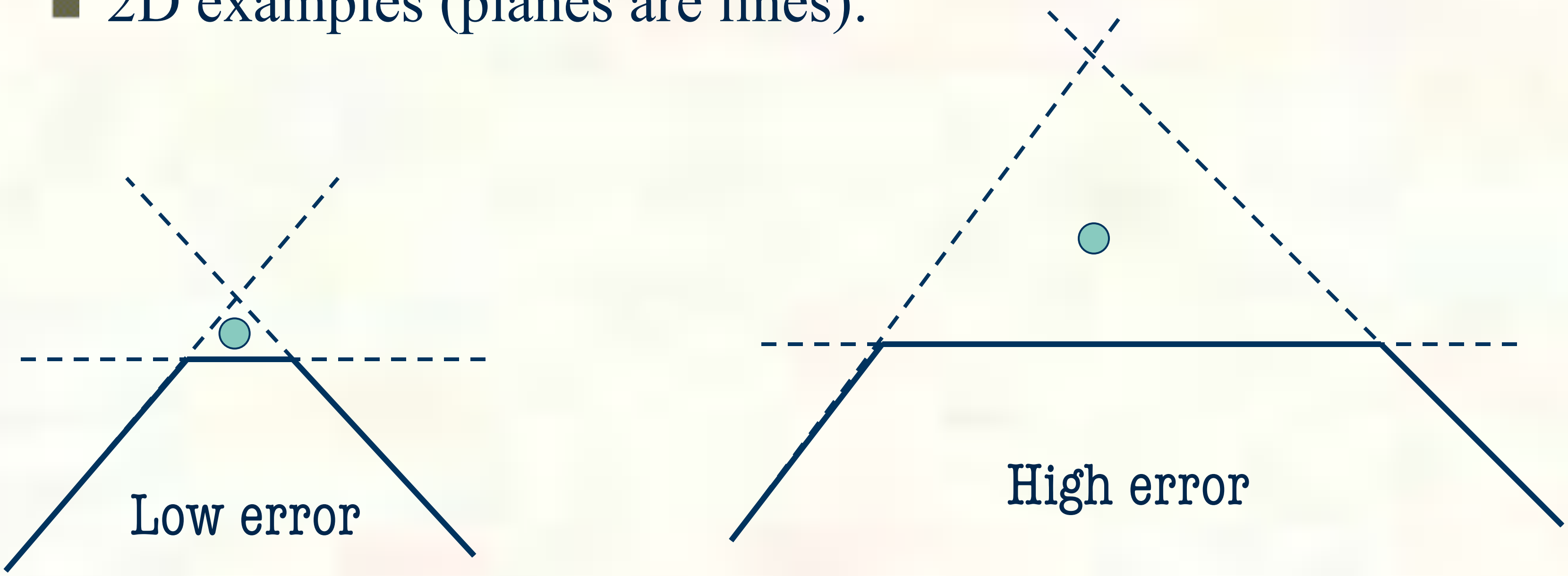
- A single operation doesn't reduce a mesh!
- Most common approach is a greedy algorithm built on edge contractions (*iterative edge contractions*):
  - Rank each possible edge contraction according to how much error it would introduce
  - Contract edge that introduces the least error
  - Repeat until done
- Does NOT produce optimal meshes
  - An optimal mesh for a given target number of triangles is the one with the lowest error with respect to the original mesh
  - Finding the optimal mesh is NP-hard (intractable)

Error Metrics

- The error metric measures the error introduced by contracting an edge
  - Should be low for edges whose removal leaves mesh nearly unchanged
    - What is an example of an edge that can be removed without introducing any error?
- Issues:
  - How well does it measure changes in *appearance*?
  - How easy is it to compute?
    - Subtle point: Error should be measured with respect to original mesh, which can pose problems
  - Can it handle color and other non-geometric attributes?

Measuring Error With Planes

- An edge contraction moves two vertices to a single new location
- Measure how far this location is from the planes of the original faces
- 2D examples (planes are lines):



Which Planes?

- Each vertex has a (conceptual) set of planes
  - Error ≡ sum of squared distances to planes in set
  - Each plane given by normal, $\boldsymbol{n}_i$, and distance to origin, $\boldsymbol{d}_i$

$$\text{Error}(\boldsymbol{v}) = \sum_i \left(\boldsymbol{n}_i^T \boldsymbol{v} + d_i\right)^2$$

- Initialize with planes of incident faces
  - Consequently, all initial errors are 0
- When contracting pair, use plane set union
  - planes(v') = planes(v_1) ∪ planes(v_2)

The Quadric Error Metric

- Given a plane, we can define a quadric Q

$$Q = \left(\boldsymbol{A}, \boldsymbol{b}, c\right) = \left(\boldsymbol{n}\boldsymbol{n}^T, d\boldsymbol{n}, d^2\right)$$

measure squared distance to the plane as

$$Q(\boldsymbol{v}) = \boldsymbol{v}^T \boldsymbol{A}\boldsymbol{v} + 2\boldsymbol{b}^T\boldsymbol{v} + c$$

$$Q(\boldsymbol{v}) = \begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} a^2 & ab & ac \\ ab & b^2 & bc \\ ac & bc & c^2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + 2\begin{bmatrix} ad & bd & cd \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + d^2$$

The Quadric Error Metric

- Sum of quadrics represents set of planes

$$\sum_i \left(\boldsymbol{n}_{\boldsymbol{i}}^{\boldsymbol{T}} \boldsymbol{v} + d_i\right)^2 = \sum_i Q_i(\boldsymbol{v}) = \left(\sum_i Q_i\right)(\boldsymbol{v})$$

- Each vertex has an associated quadric
  - Error(v_i) = $Q_i\ (v_i)$
  - Sum quadrics when contracting $(v_i, v_j) \rightarrow v'$
  - Error introduced by contraction is $Q(v')$

$$Q = Q_i + Q_j = \left(\boldsymbol{A}_i + \boldsymbol{A}_j, \boldsymbol{b}_i + \boldsymbol{b}_j, c_i + c_j\right)$$

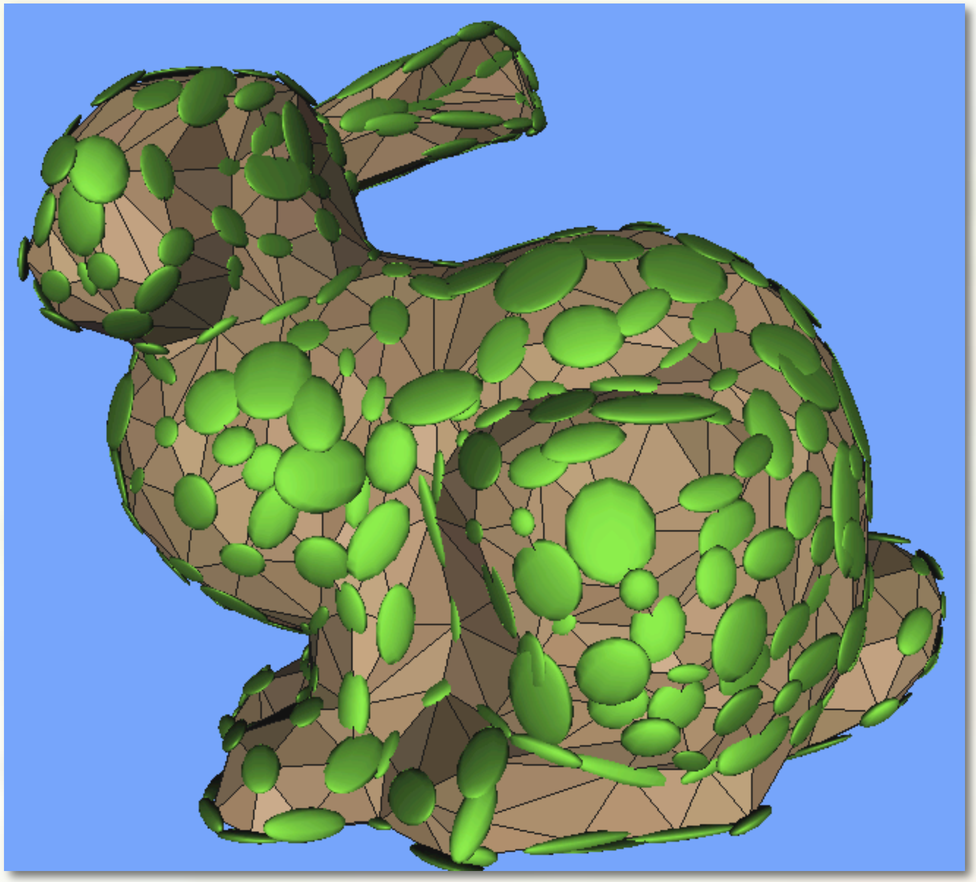
Where Does v' Belong?

- Choose the location for v' that will minimize the error introduced

$$\nabla Q(\boldsymbol{v}') = 0 \Rightarrow \boldsymbol{v}' = -\boldsymbol{A}^{-1}\boldsymbol{b}$$

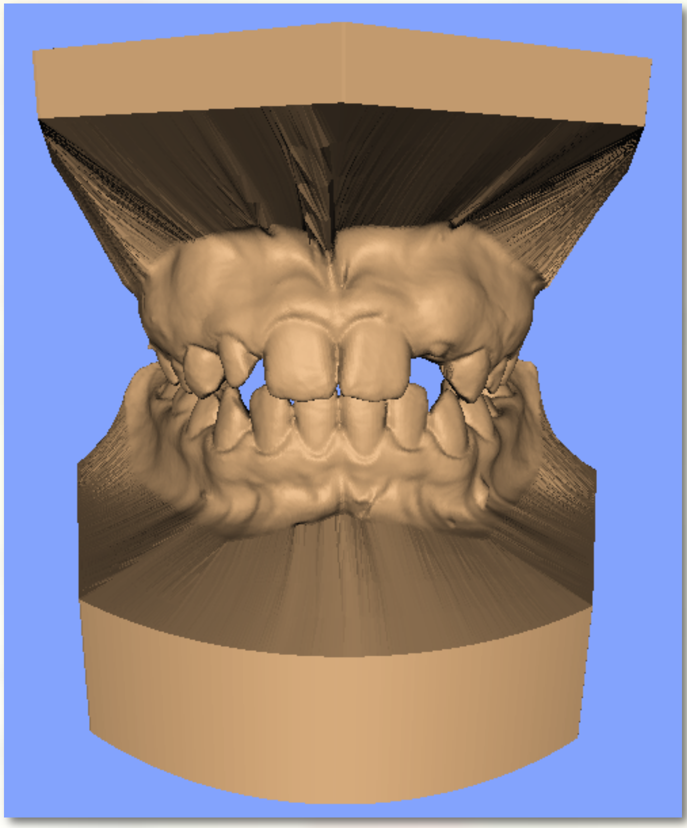
- Alternative is to use fixed placement:
  - Select v_1 or v_2
  - Fixed placement is faster but lower quality
  - But it also gives smaller progressive meshes (more later)
  - Fallback to fixed placement if A is non-invertible
    - When is A non-invertible (hard!)?

Visualizing Quadrics in 3-D

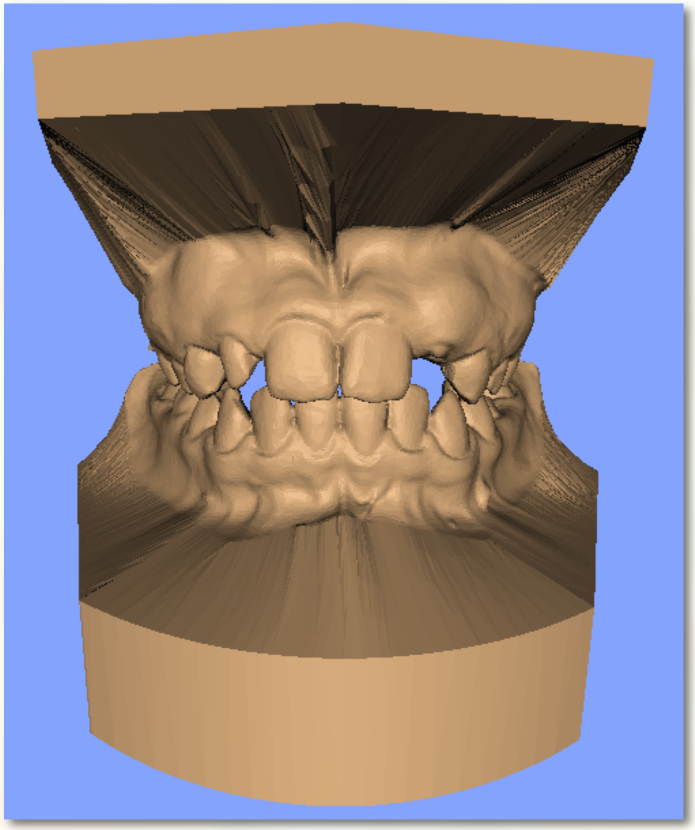


- Quadric isosurfaces
  - Contain all the vertex locations that are within a given distance of the face planes
  - Are ellipsoids (maybe degenerate)
  - Centered around vertices
  - Characterize shape
  - Stretch in least-curved directions

Sample Model: Dental Mold



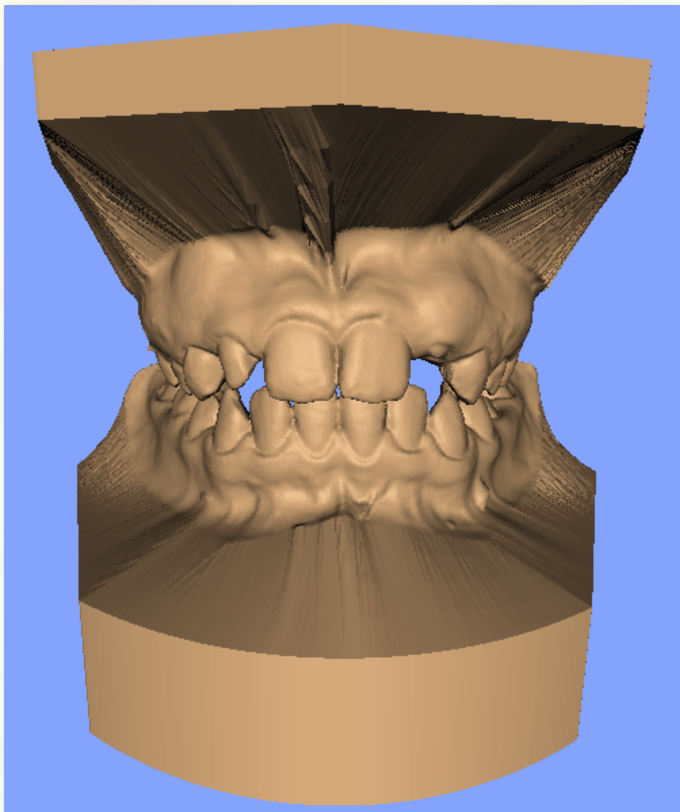
50 sec



424,376 faces

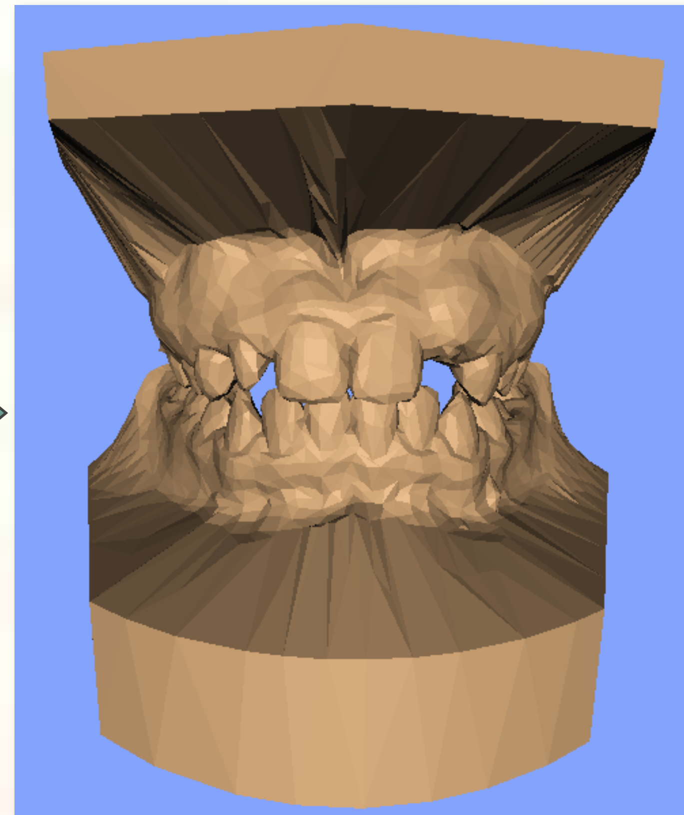
60,000 faces

Sample Model: Dental Mold



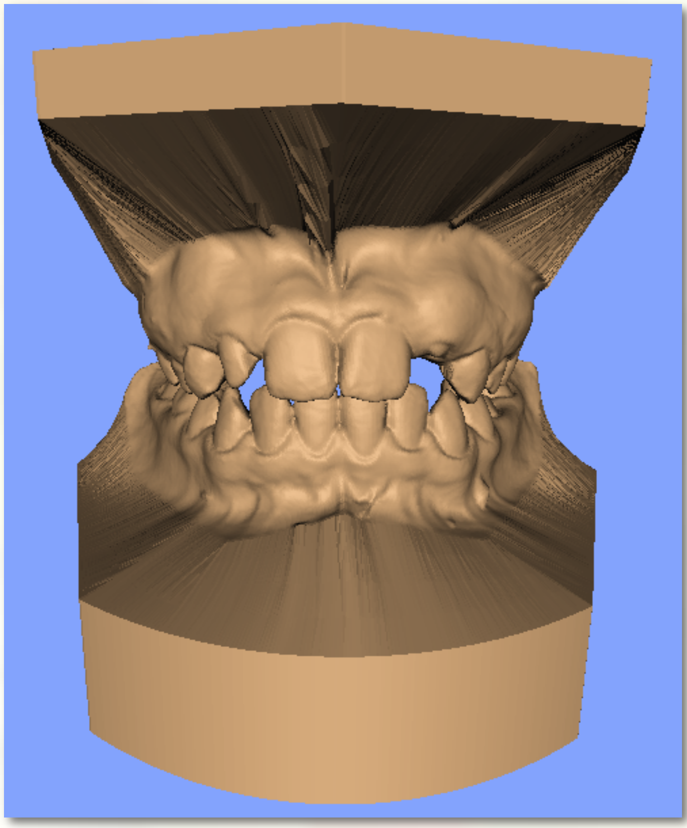
424,376 faces

55 sec



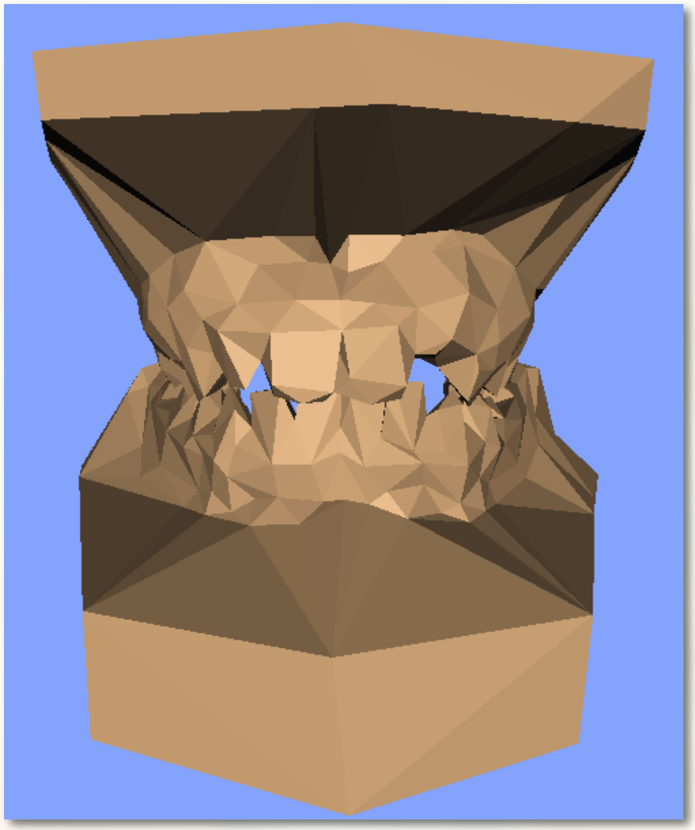
8000 faces

Sample Model: Dental Mold



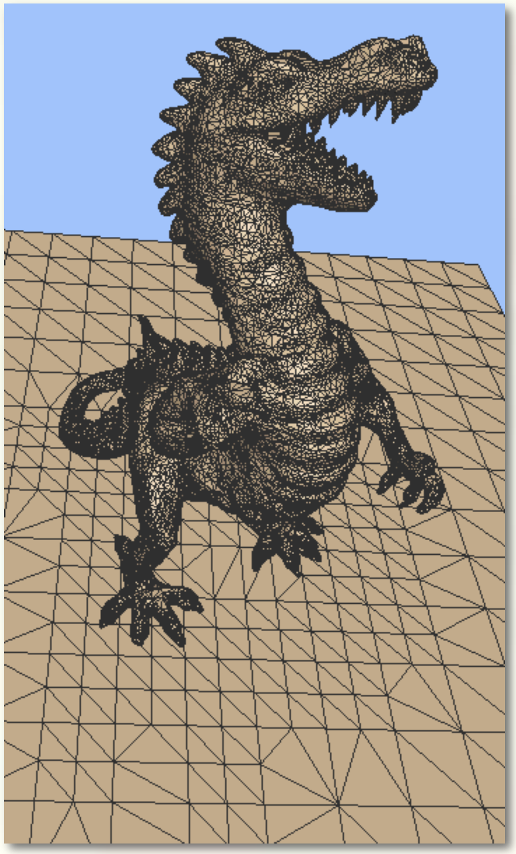
424,376 faces

56 sec



1000 faces

Must Also Consider Attributes

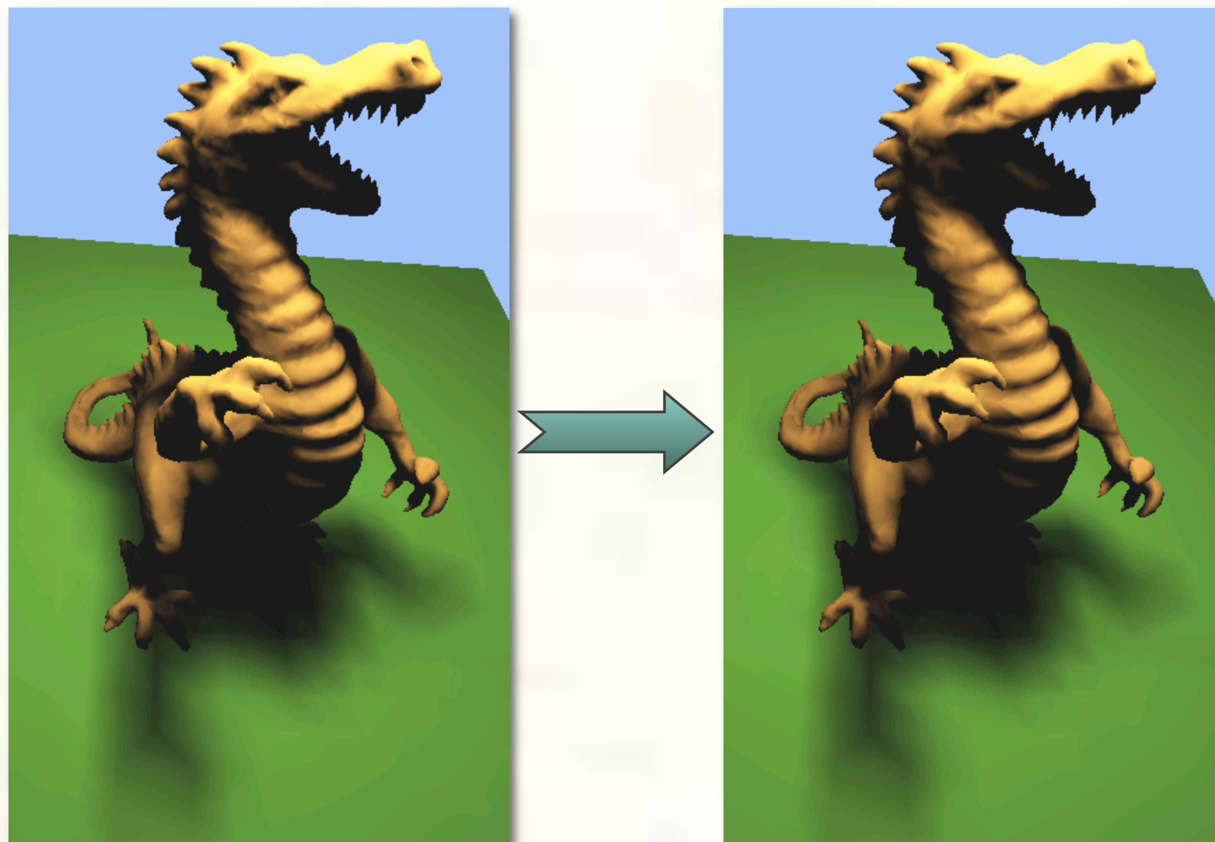


Mesh for solution

Radiosity solution

Must Also Consider Attributes

- Add extra terms to plane equations for color information (or texture coords)
- Makes matrices and vectors bigger, but doesn’t change overall approach



50,761 faces

10,000 faces

LOD Switching

- The biggest issue with switching is *popping*, the sudden change in visual appearance as the models are swapped
  - Particularly poor if the object is right at the switching distance – may flicker badly
- There are several options:
  - Leave the popping but try to avoid flicker
  - Show a blended combination of two models, based on the distance between the switching points
    - Image blend – render two models at alpha 0.5 each
    - Geometric blend – define a way to geometrically morph from one model to the next, and blend the models
  - Have more models that are so similar that the popping is not evident

Hysteresis Thresholds

- The aim is to avoid rapid popping if the object straddles the threshold for switching
- Define two thresholds for each switch, distance as the metric
  - One determines when to improve the quality of the model
  - The other determines when to reduce it
  - The reduction threshold should be at a greater distance than the increase threshold
  - If the object is between the thresholds, use the same LOD as on the previous frame

Hysteresis Illustration

- Say there is an object that repeatedly jumps from one position to a nearer one and then back again
  - How far must it jump to exhibit popping?
- One way to think about it: If you're on one level, you can only ride the arrows to the next level