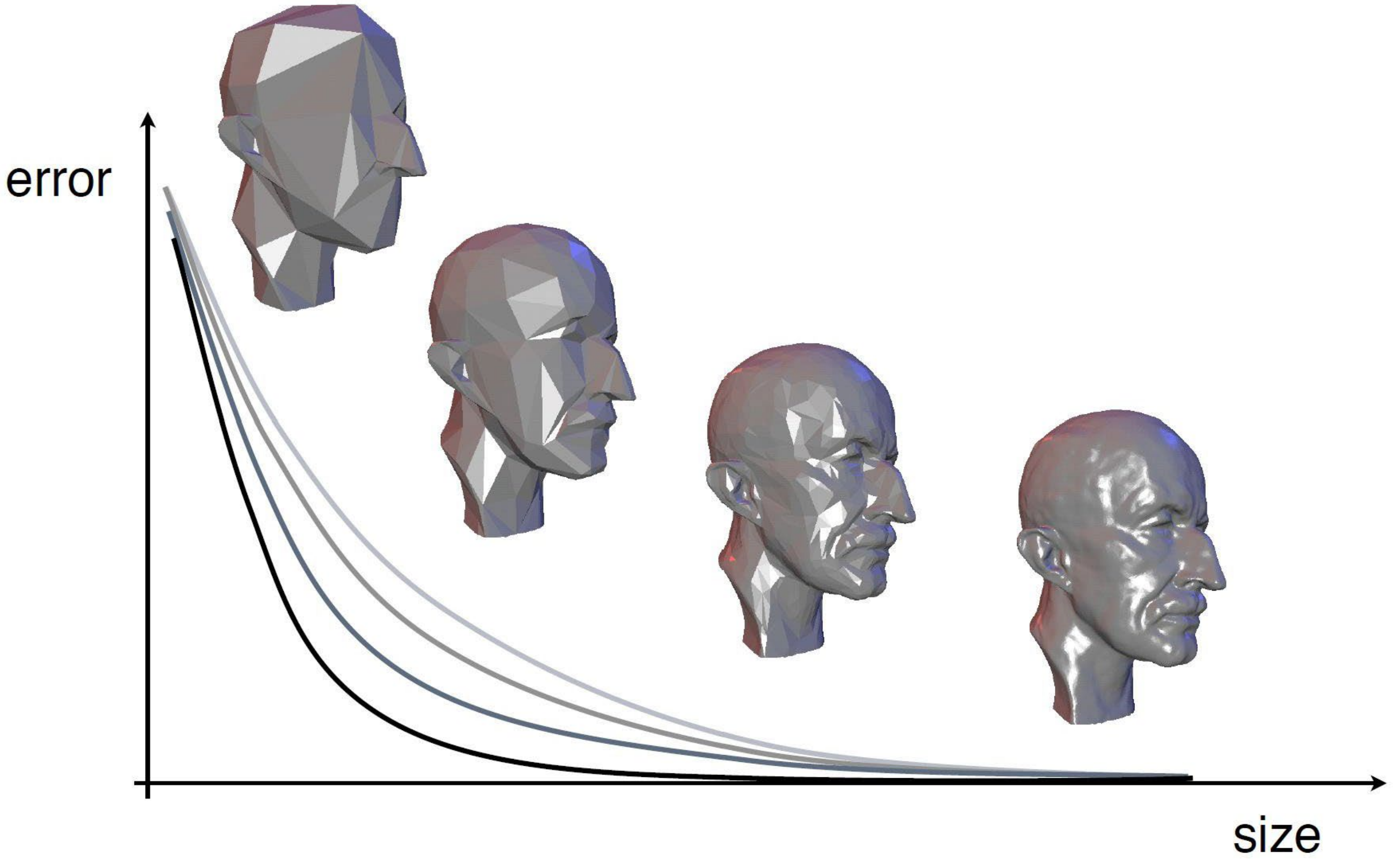
Rendering
Level of Detail: the Math

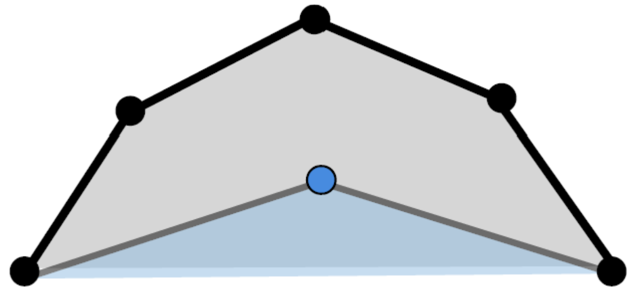
CS 415: Game Development

Professor Eric Shaffer

Mesh Simplification: Polygon Count vs Quality



Here ***error*** means some metric that measures how far away from the original surface the low poly surface is



…measuring the difference between two surfaces is complicated and there are many different ways people attempt to do it

Squared Distance From a Point to a Plane

$$p = (x, y, z, 1)^T, \quad q = (a, b, c, d)^T$$

$$dist(q, p)^2 = (q^T p)^2 = p^T (qq^T) p$$

Using implicit form of a plane equation ax+by+cz+d=0

Squared Distance: Matrix Form

$$p = (x, y, z, 1)^T, \quad q = (a, b, c, d)^T$$

$$dist(q, p)^2 = (q^T p)^2 = p^T (qq^T) p = p^T Q_q p$$

$$Q_q = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

Error Quadrics

$$p = (x, y, z, 1)^T, \quad q = (a, b, c, d)^T$$

$$dist(q, p)^2 = (q^T p)^2 = p^T (q q^T) p = p^T Q_q p$$

$$Q_q = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

What's the distance from p = (1,3,4) to plane z=0?

What's the distance from p = (1,3,2) to plane x+y+z=1?

Error Quadrics

- Sum distances to vertex' planes

$$\sum_i dist(q_i, p)^2 = \sum_i p^T Q_{q_i} p = p^T \left(\sum_i Q_{q_i} \right) p =: p^T Q_p p$$

You can compute the sum of squared distances from p to N planes using a single 4x4 matrix

You simply sum up the N matrices Q_{q_i} component-wise and use it as shown here.

Imagine we cluster a bunch of vertices together and replace them with a single vertex

How can we use this formula to place this new vertex?

ILLINOIS

Vertex Placement

$$\mathbf{Q} = \begin{bmatrix} \boxed{\mathbf{A}} & \boxed{\mathbf{b}} \\ \boxed{\mathbf{b}^\mathsf{T}} & \boxed{c} \end{bmatrix}$$

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

Point v that minimizes $Q(v)$ occurs when

$$\partial Q/\partial x = \partial Q/\partial y = \partial Q/\partial z = 0.$$

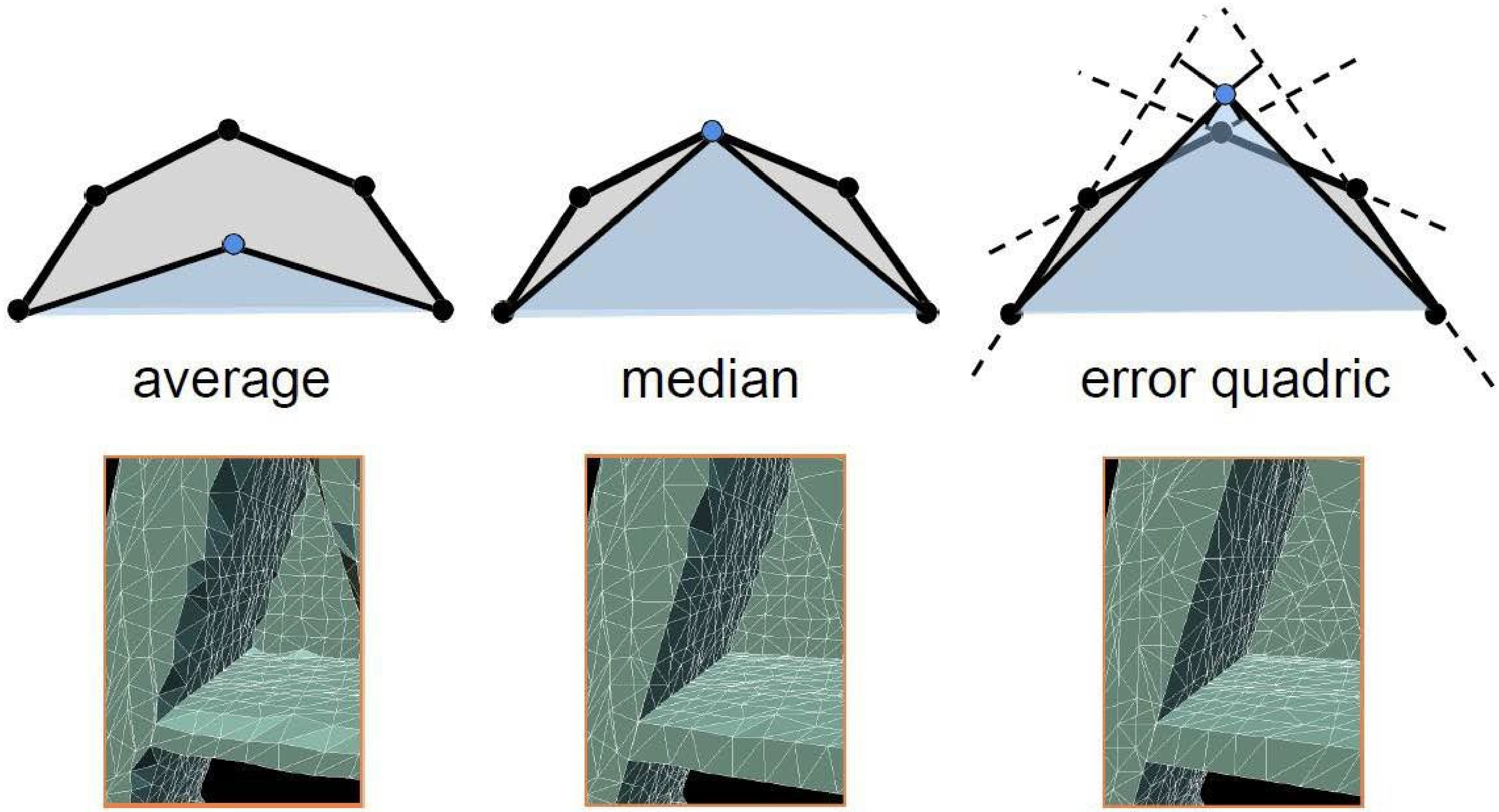
The gradient of $Q(v)$ is

$$\nabla Q(\mathbf{v}) = 2\mathbf{A}\mathbf{v} + 2\mathbf{b}$$

Solving for $\nabla Q(\mathbf{v}) = 0$, we find that the optimal position is

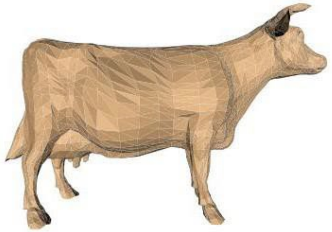
$$\bar{\mathbf{v}} = -\mathbf{A}^{-1}\mathbf{b}$$

Comparison of Vertex Placements

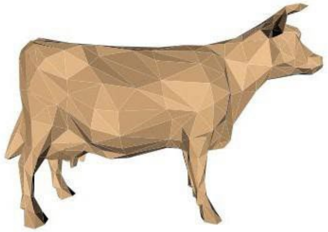


ILLINOIS

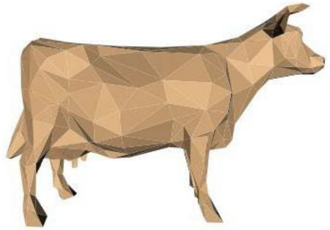
Another Example.....



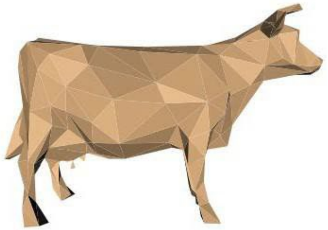
(a) 5804 faces



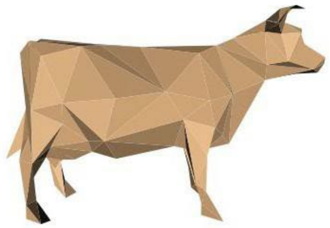
(b) 1000 faces



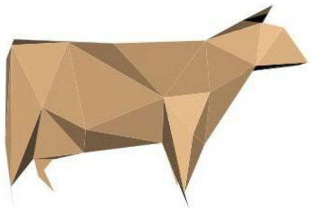
(c) 800 faces



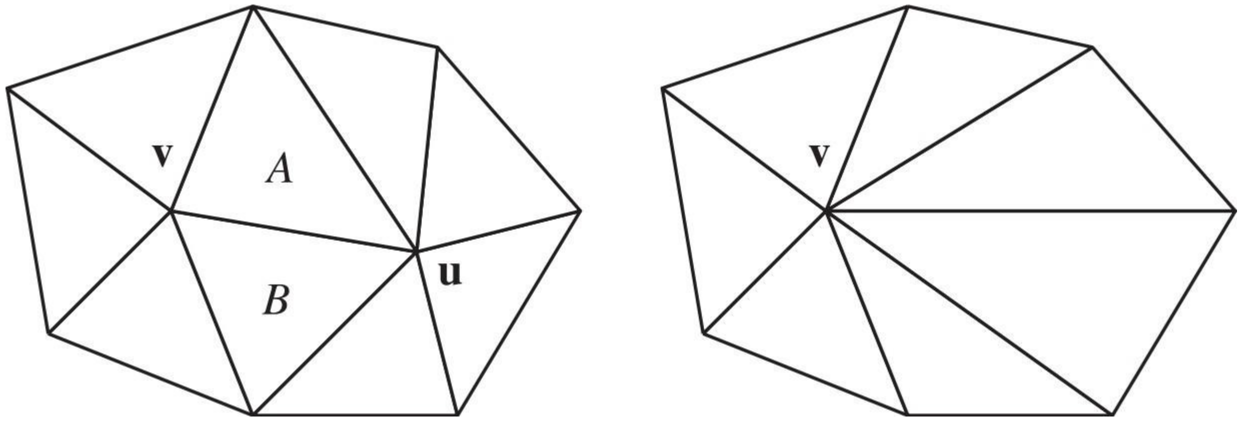
(d) 500 faces



(e) 300 faces

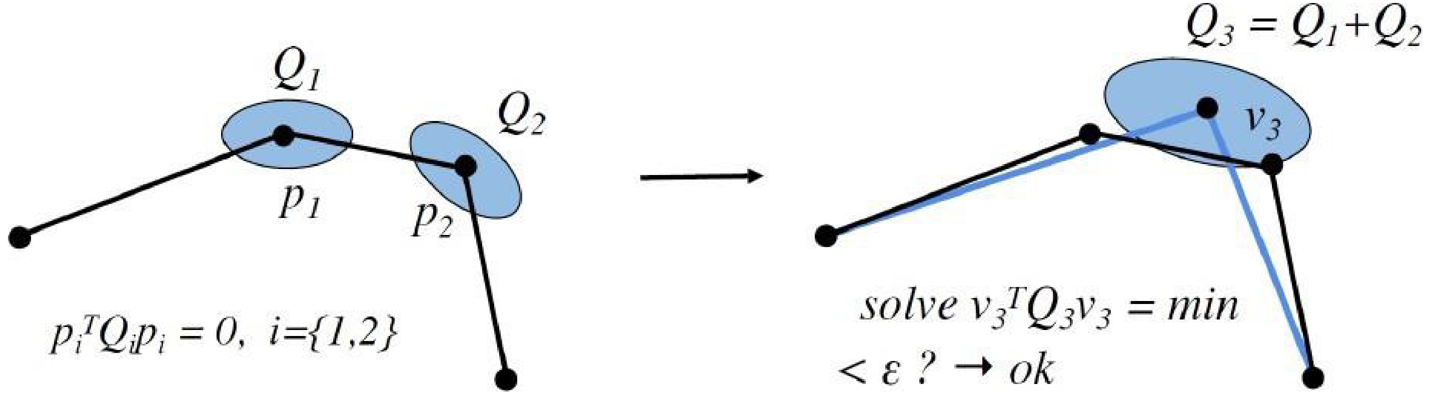


(f) 100 faces

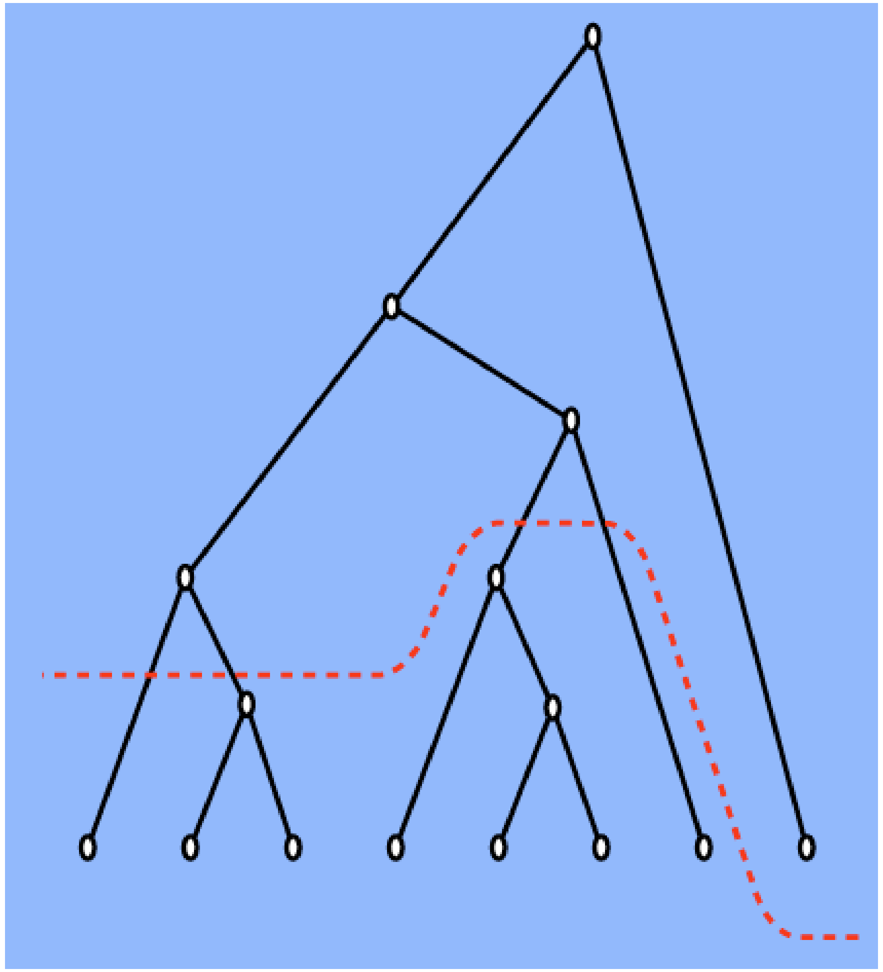


Incremental Simplification Algorithm

1. Compute Quadric for each vertex
2. Create a priority queue of all possible edge collapses $p_1 + p_2$
   1. For each edge collapse compute $Q_3 = Q_1 + Q_2$
   2. Compute new vertex v_3
   3. Compute error – how?
3. Choose collapse with least error…update quadrics and repeat



Continuous LOD using Vertex Hierarchies

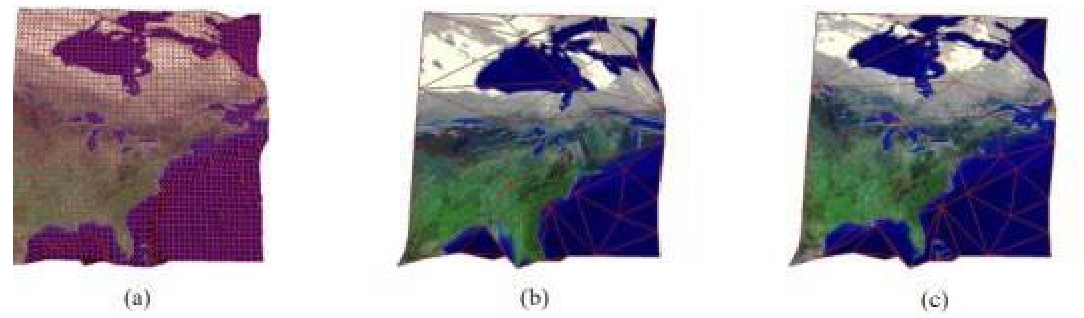


- Each original vertex in mesh is a leaf in this diagram
- An edge contraction makes vertices siblings and creates a parent
- A cut through the tree is a set of contractions applied to the mesh
- Could do CLOD (sort of) by using screen space metric to determine cut
- …adjust cut each frame…

ILLINOIS

In practice LOD gets more complicated

- Error metric should incorporate color and texture information
  - …don’t want to merge discontinuous parts of texture if possible



Geometry & texture: A 3,872 face model (a) reduced to 53 faces without (b) and with (c) updating texture coordinates.

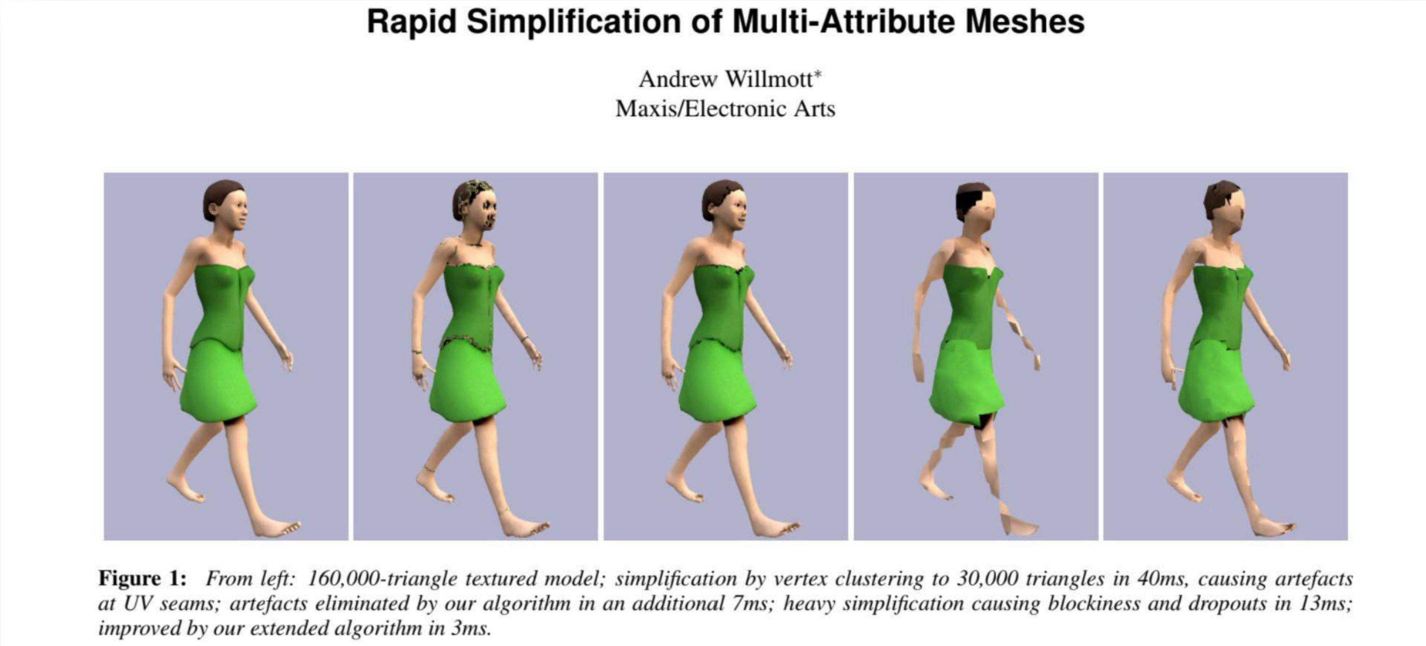
- Features may need to be preserved or removed…requires artist input

Original mesh LOD 2 Defeatured, then LODed

ILLINOIS

Actually Used in Games…

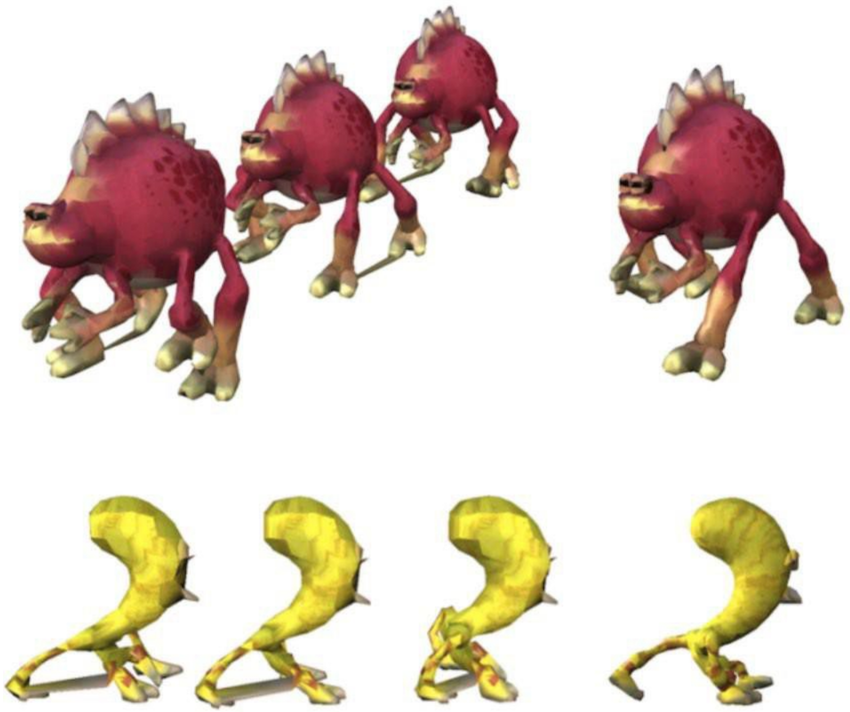
Can anyone name the game?



Rapid Simplification of Multi-Attribute Meshes

Andrew Willmott*
Maxis/Electronic Arts

Figure 1: *From left: 160,000-triangle textured model; simplification by vertex clustering to 30,000 triangles in 40ms, causing artefacts at UV seams; artefacts eliminated by our algorithm in an additional 7ms; heavy simplification causing blockiness and dropouts in 13ms; improved by our extended algorithm in 3ms.*



ILLINOIS

Spore

Spore

Developer(s)	Maxis
Publisher(s)	Electronic Arts
Designer(s)	Will Wright Alex Hutchinson Jenna Chalmers Chaim Gingold Stone Librande Soren Johnson
Programmer(s)	Andrew Willmott
Artist(s)	Michael A. Khoury
Composer(s)	Brian Eno Cliff Martinez
Platform(s)	Microsoft Windows[2] Mac OS X[2]
Release	September 5, 2008[1][2]
Genre(s)	God game, life simulation, real-time strategy
Mode(s)	Single-player

Abstract

We present a rapid simplification algorithm for meshes with multiple vertex attributes, targeted at rendering acceleration for real-time applications. Such meshes potentially feature normals, tangents, one or more texture coordinate sets, and animation information, such as blend weights and indices. Simplification algorithms in the literature typically focus on position-based meshes only, though extensions to handle surface attributes have been explored for those techniques based on iterative edge contraction. We show how to achieve the same goal for the faster class of algorithms based on vertex clustering, despite the comparative lack of connectivity information available. In particular, we show how to handle attribute discontinuities, preserve thin features, and avoid animation-unfriendly contractions, all issues which prevent the base algorithm from being used in a production situation.

Our application area is the generation of multiple levels of detail for player-created meshes at runtime, while the main game process continues to run. As such the robustness of the simplification algorithm employed is key; ours has been run successfully on many millions of such models, with no preprocessing required. The algorithm is of application anywhere rapid mesh simplification of standard textured and animated models is desired.

From a university to Electronic Arts

ILLINOIS