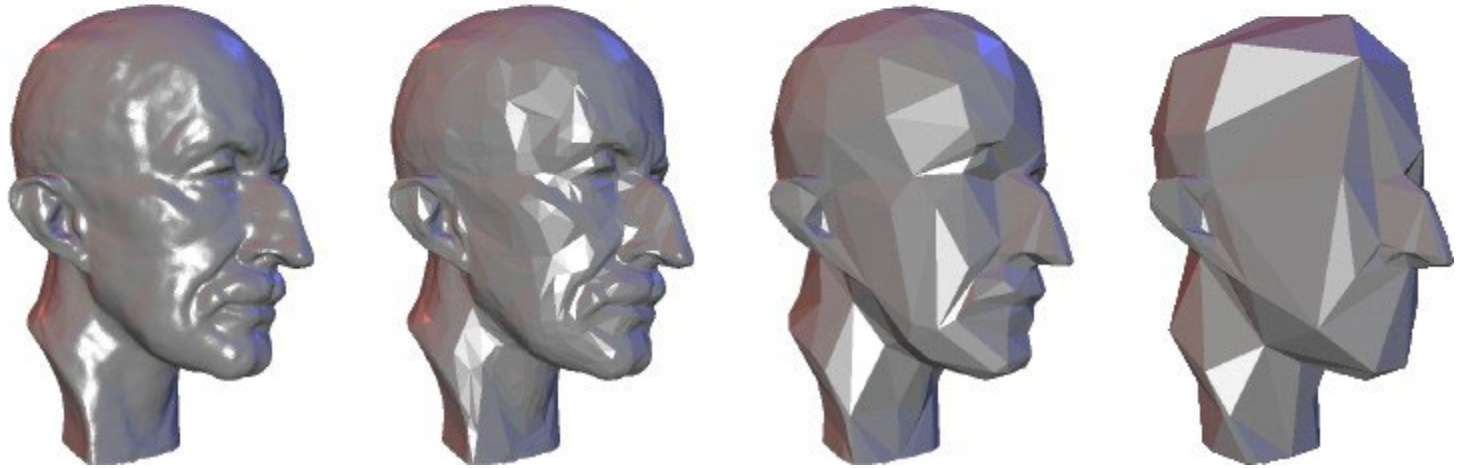
# Rendering
# Level of Detail

## CS 415: Game Development

Professor Eric Shaffer

# Level-of-Detail Rendering

- Construct multiple versions of mesh
  - Varying polygon count
- Multi-resolution hierarchies enable
  - efficient geometry processing
  - level-of-detail (LOD) rendering

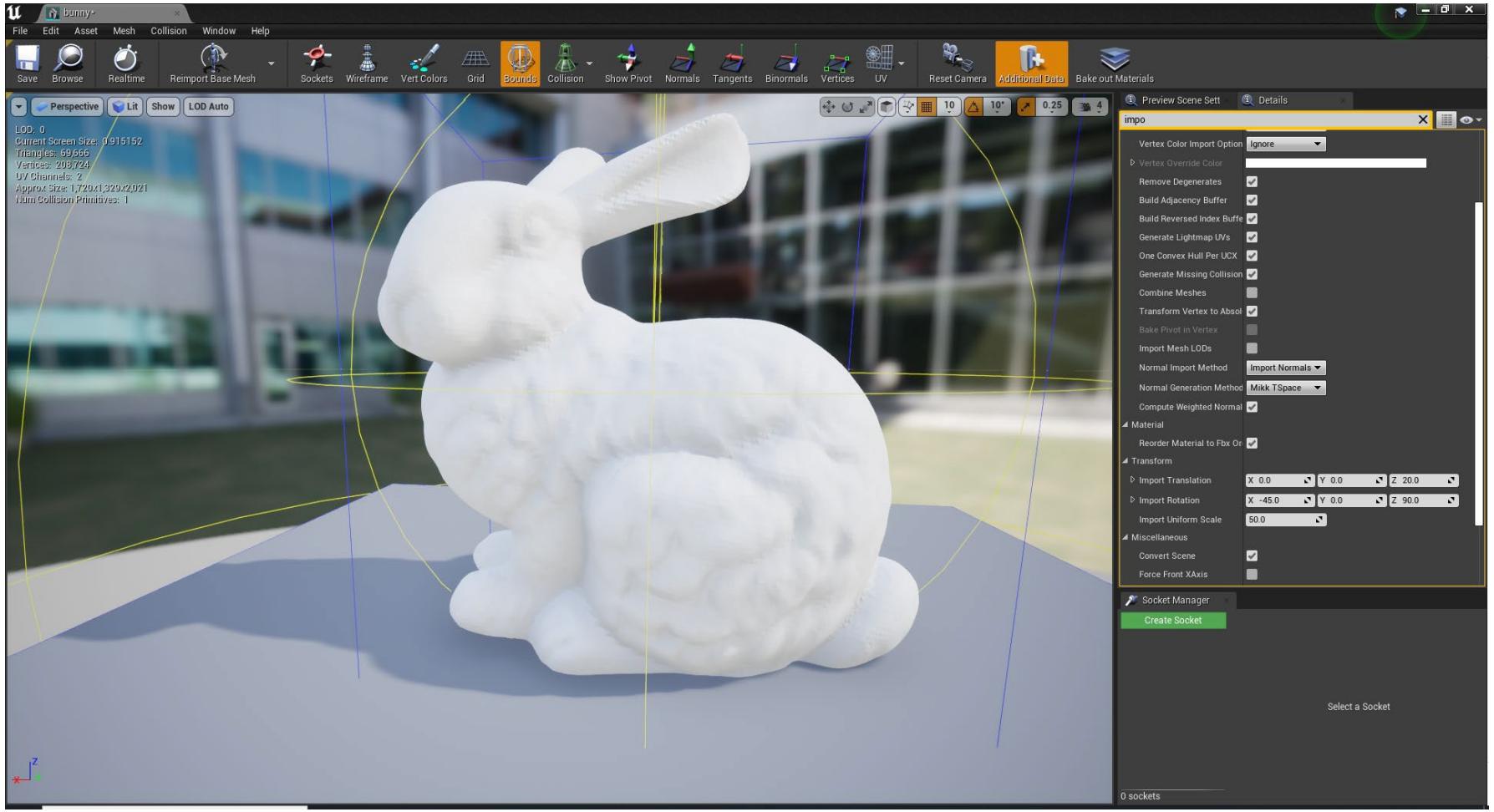


Pick which model to use based on approximate screen size…

Approx screen size → need to guess the width and height in pixels of the rendered mesh….

How could the engine make this guess?

# Unreal Engine 4 Mesh Editor




ILLINOIS

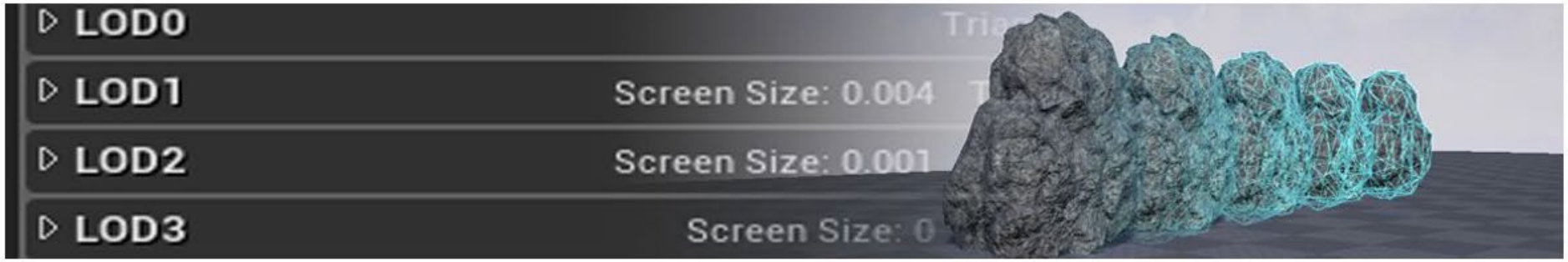
https://docs.unrealengine.com/4.27/en-US/WorkingWithContent/Types/StaticMeshes/HowTo/AutomaticLODGeneration/

# Generating LODs for a Static Mesh

## Setting Up Automatic LOD Generation

How To use the Automatic LOD Generation system in UE4.

Intermediate




ON THIS PAGE

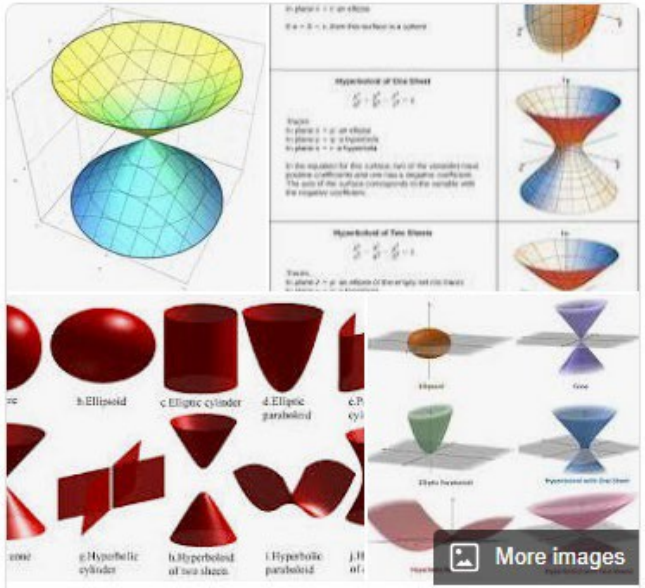


The Automatic LOD generation system allows you to automatically reduce the polygon count of your Static Meshes to create LODs with the Unreal Engine 4 (UE4) Editor. Automatic LOD generation uses what is called quadratic mesh simplification to help generate the LODs for Static Meshes. Quadratic mesh simplification works by calculating the amount of visual difference that collapsing an edge (by merging two vertices) would generate. It then picks the edge with the least amount of visual impact and collapses it. When this happens, the tool will pick the best place to put the newly merged vertex, removing any triangles that have also collapsed along with the edge. It will continue to collapse edges until it reaches the requested target number of triangles. In the following guide, we'll show you how-to setup and use the automatic LOD generation system in your UE4 projects.

That's wrong - should be quadric mesh simplification

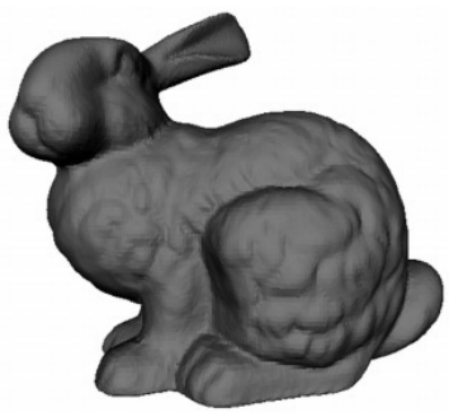
IS

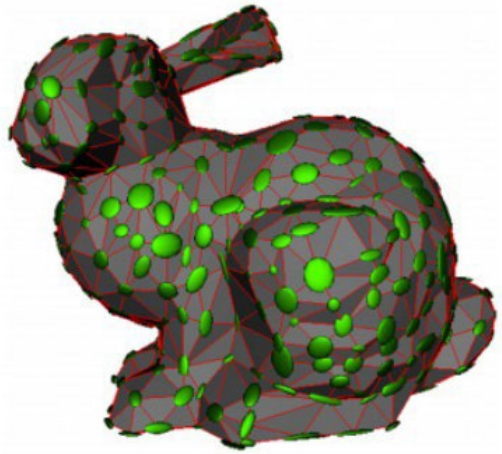
# What is a Quadric?




Quadric

In mathematics, a quadric or quadric surface, is a generalization of conic sections. It is a hypersurface in a-dimensional space, and it is defined as the zero set of an irreducible polynomial of degree two in D + 1 variables. Wikipedia



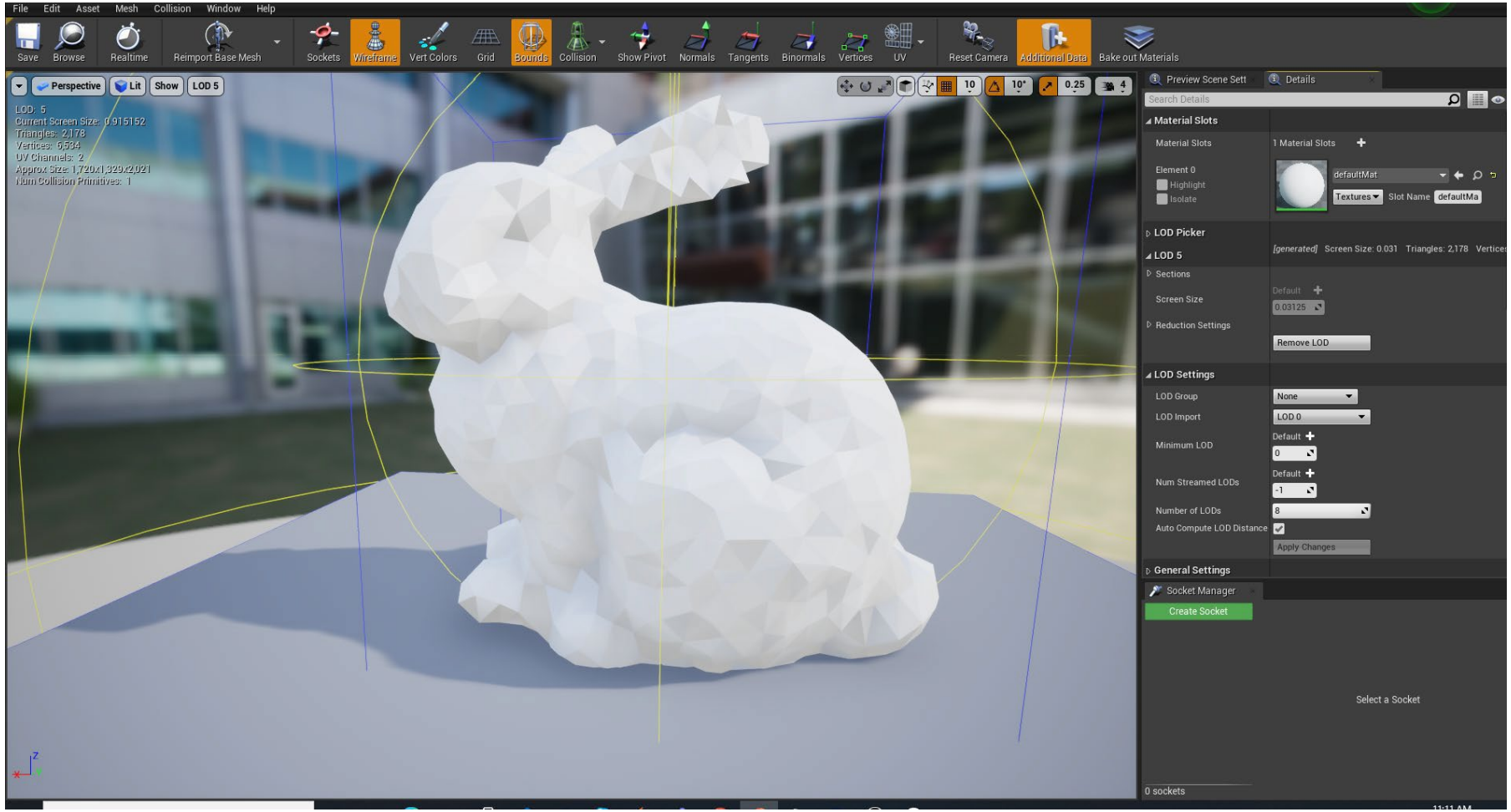


Error ellipsoids at each vertex are a type of measurement…

Shows how far away from the original surface the vertex is…

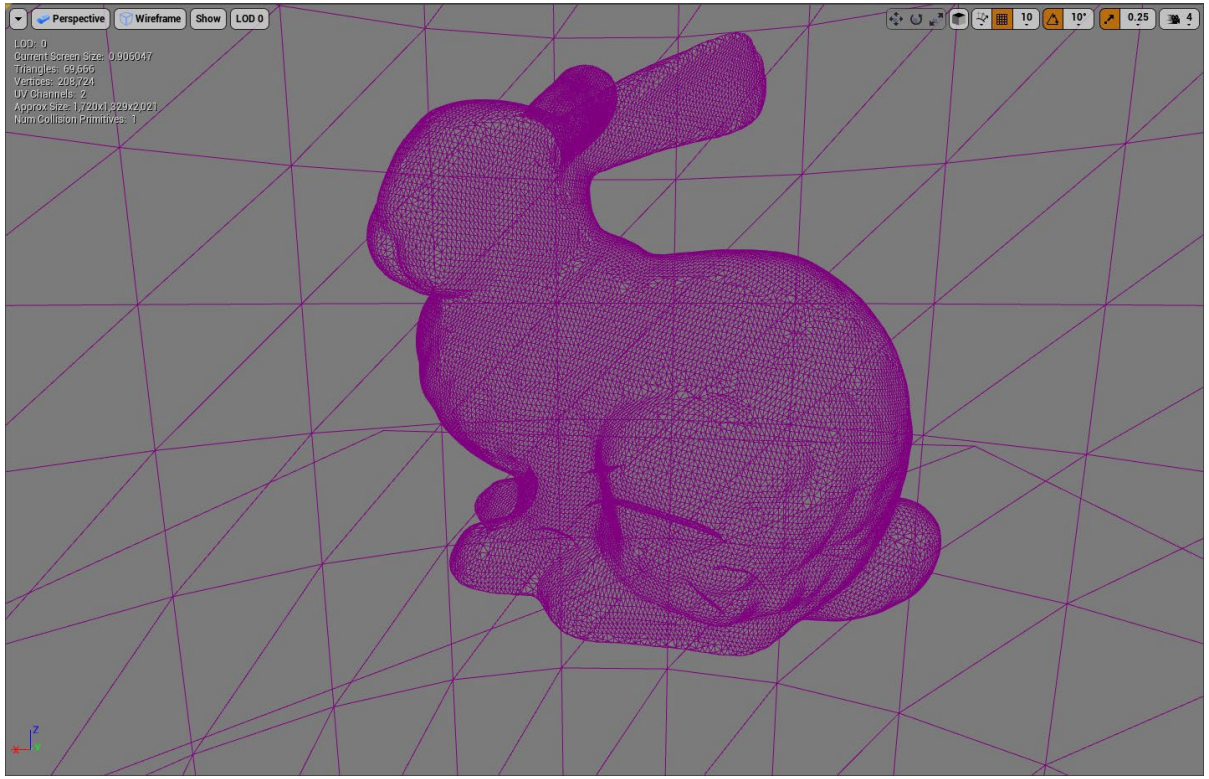
ILLINOIS

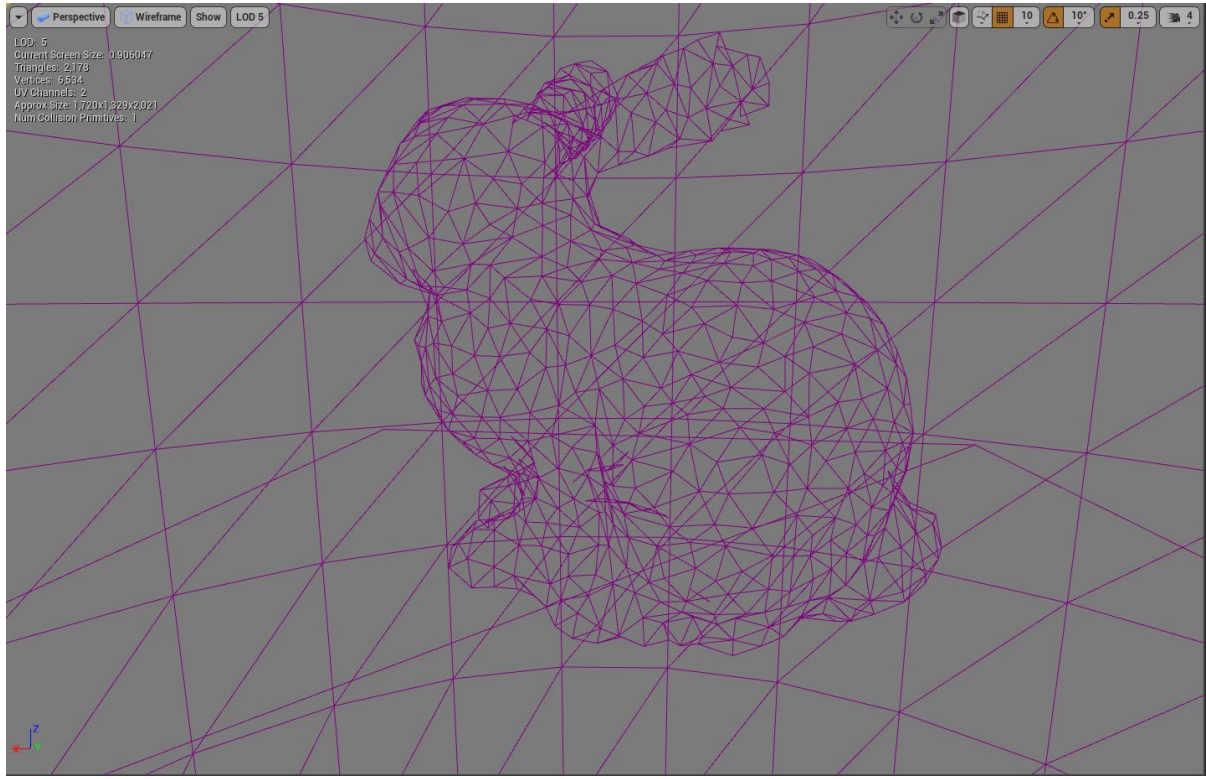
# Bunny LOD 5




ILLINOIS

# Comparison: LOD0 vs LOD5




LOD5 has about 3% of the number of polys as LOD0

ILLINOIS

# Rendering: LOD0 vs LOD5



ILLINOIS

# Error Quadrics

- Squared distance to plane

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

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

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

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

ILLINOIS

# 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

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

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

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

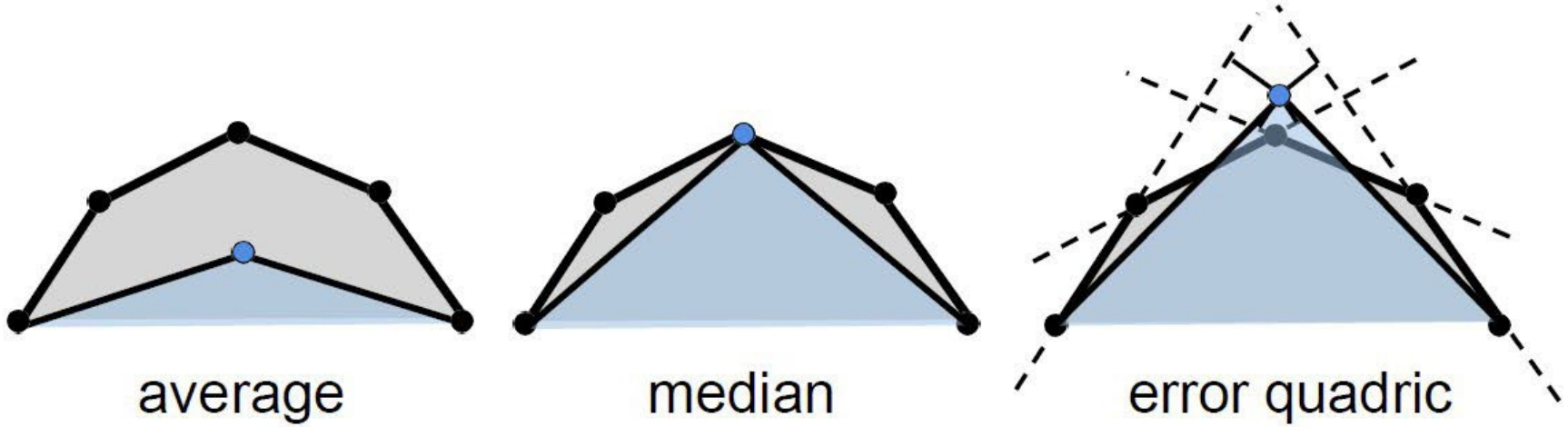
$$\nabla Q(\mathbf{v}) = 2\mathbf{A}\mathbf{v} + 2\mathbf{b}$$

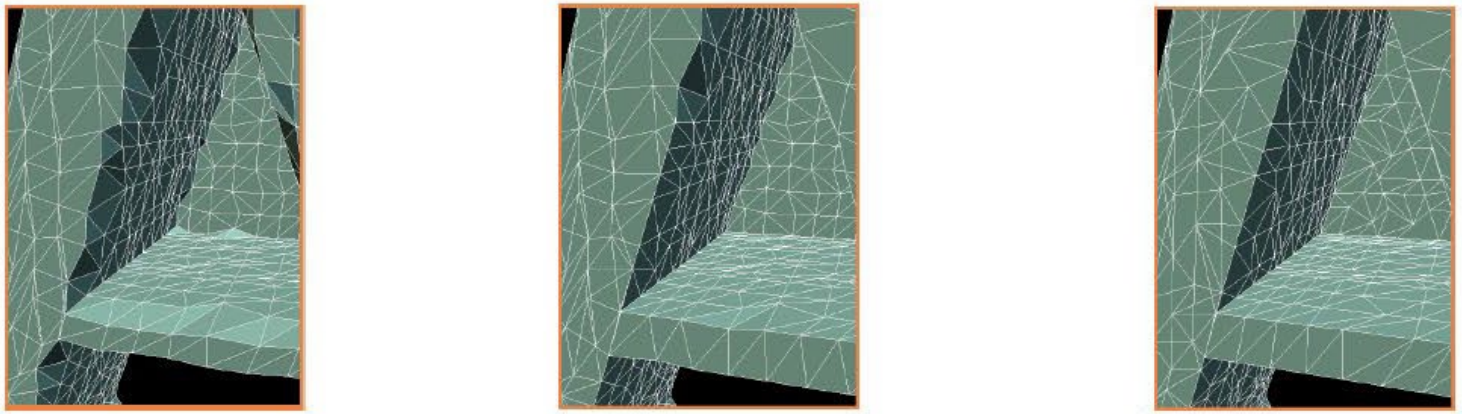
Solving for $\nabla Q(\mathbf{v}) = 0$, we find that the optimal position is

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

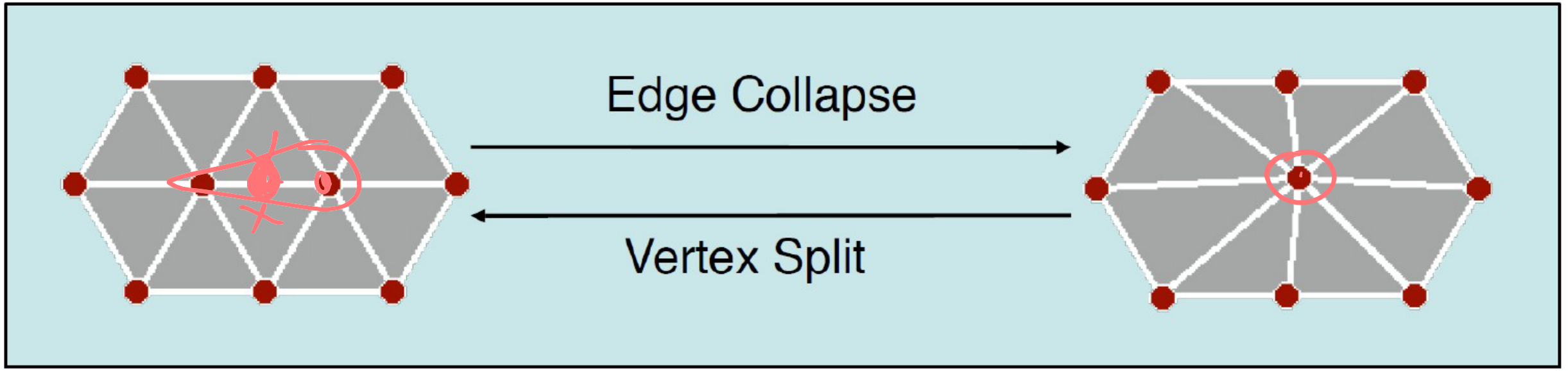
ILLINOIS

# Comparison of Vertex Placements



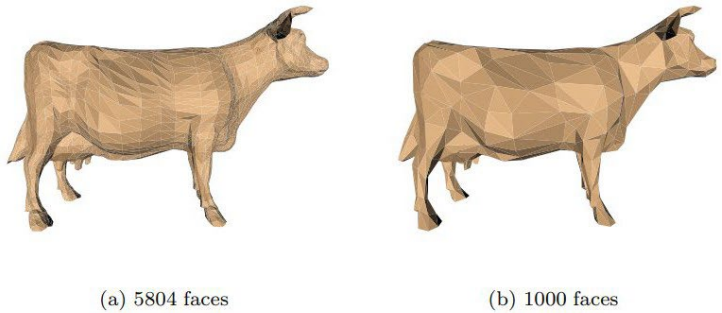



# Incremental Simplification: Edge Collapse




- Merge two adjacent triangles
- Define new vertex position

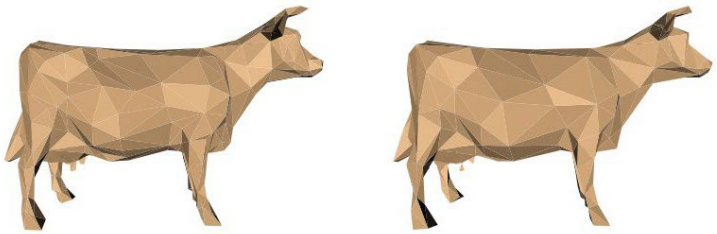
# Another Example.....



(a) 5804 faces



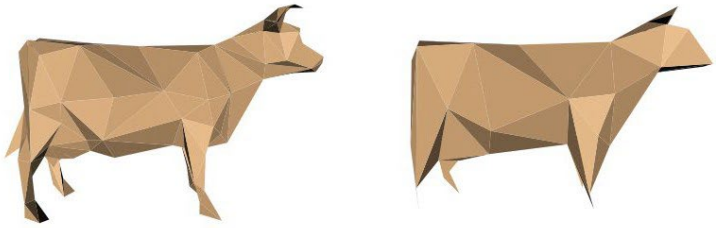
(b) 1000 faces



(c) 800 faces



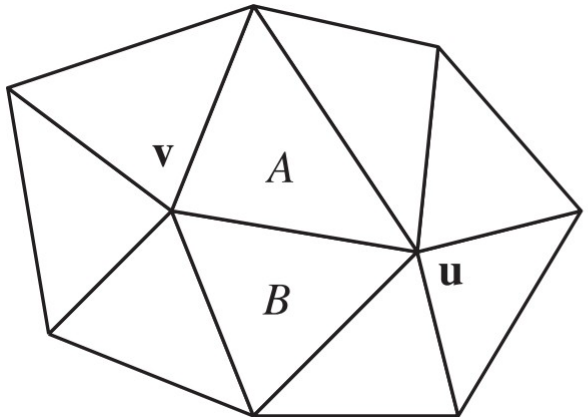
(d) 500 faces

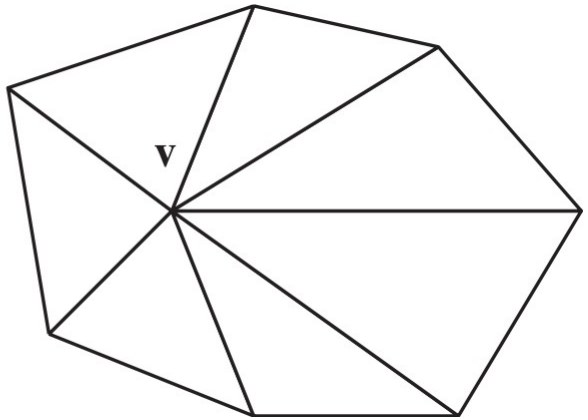


(e) 300 faces



(f) 100 faces

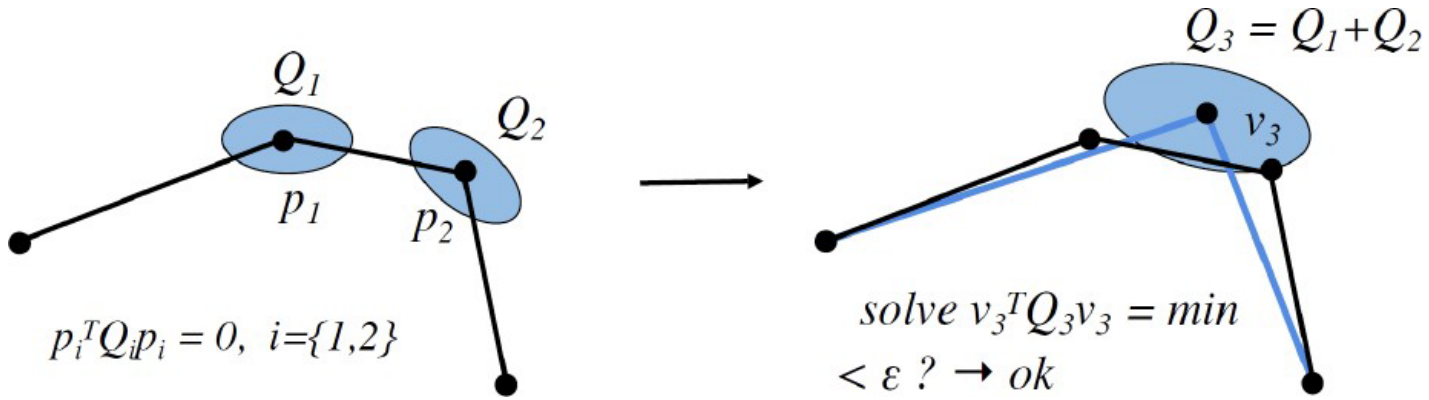



ILLINOIS

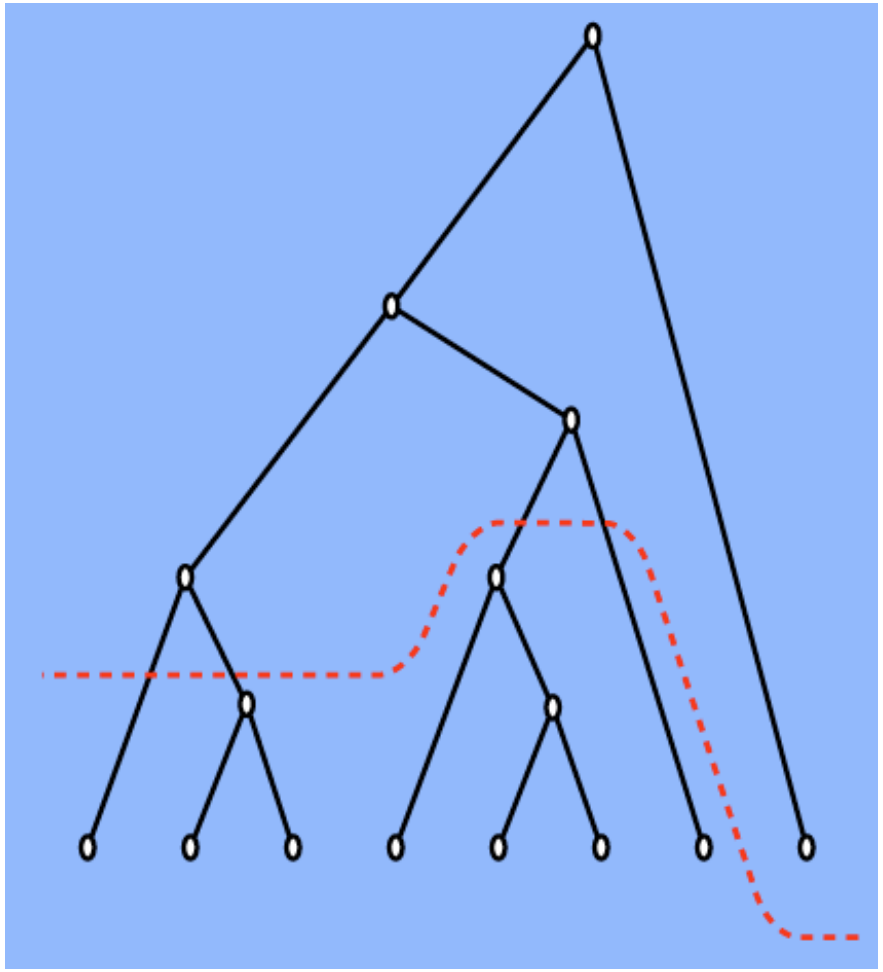
# Incremental Simplification Algorithm

1. Compute Quadric for each vertex
2. Create a priority queue of all possible edge collapses $pp_1 + pp_2$
   1. For each edge collapse compute $QQ_3 = QQ_1 + QQ_2$
   2. Compute new vertex $vv_3$
   3. Compute error $vv_3^{TT}QQ_3vv_3$
3. Choose collapse with least error…update quadrics and repeat




ILLINOIS

# Continuous LOD using Vertex Hierarchies

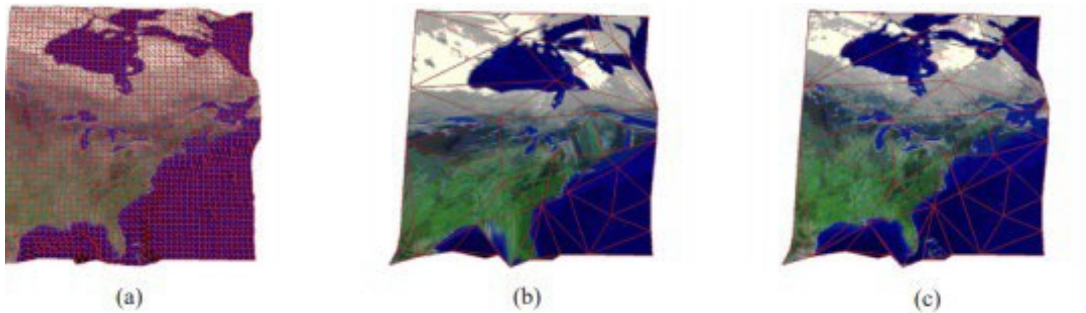


- Each original vertex in mesh is a leaf in this diagram
- An edge contraction makes vertices siblings and creates a parent
- A cut through the tree is a set of contractions applied to the mesh
- Could do CLOD (sort of) by using screen space metric to determine cut
- …adjust cut each frame…

ILLINOIS

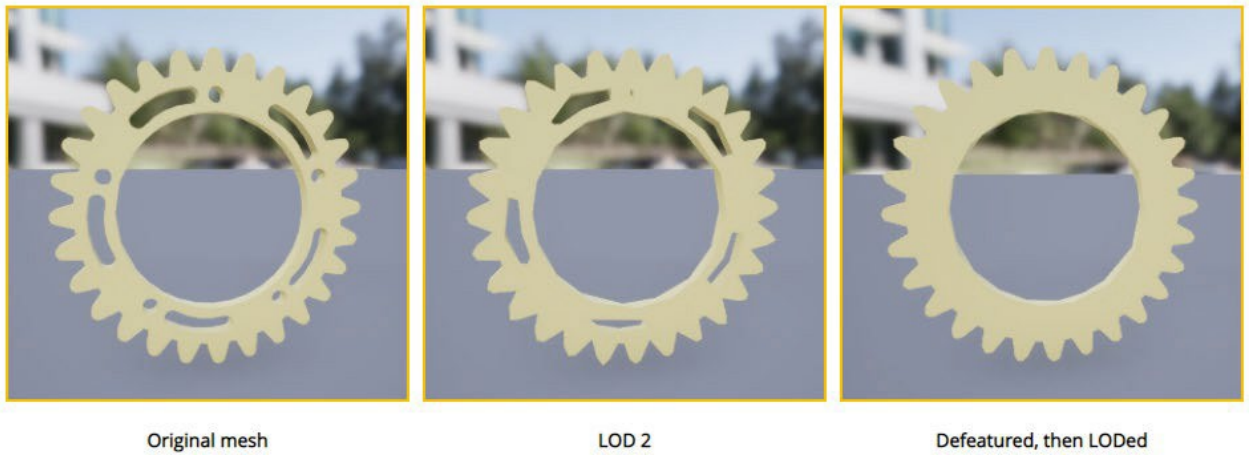
# In practice LOD gets more complicated

- 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




ILLINOIS

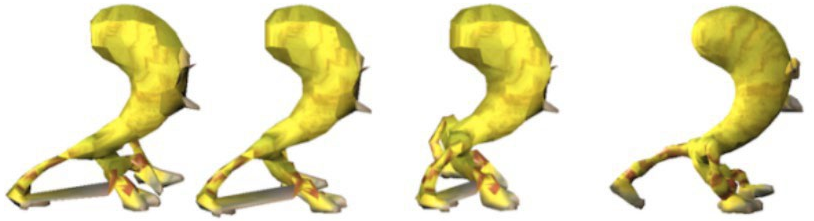
# Actually Used in Games…

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<br>Alex Hutchinson<br>Jenna Chalmers<br>Chaim Gingold<br>Stone Librande<br>Soren Johnson |
| **Programmer(s)** | Andrew Willmott |
| **Artist(s)** | Michael A. Khoury |
| **Composer(s)** | Brian Eno<br>Cliff Martinez |
| **Platform(s)** | Microsoft Windows[2]<br>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