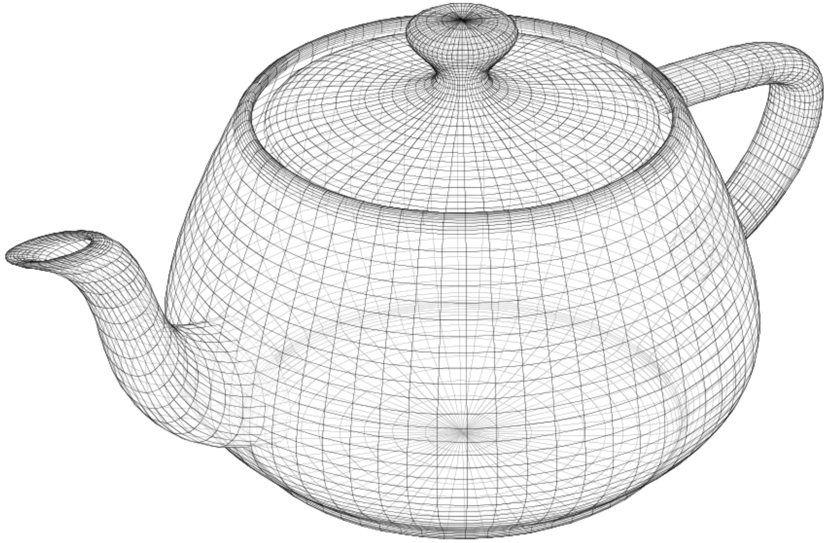


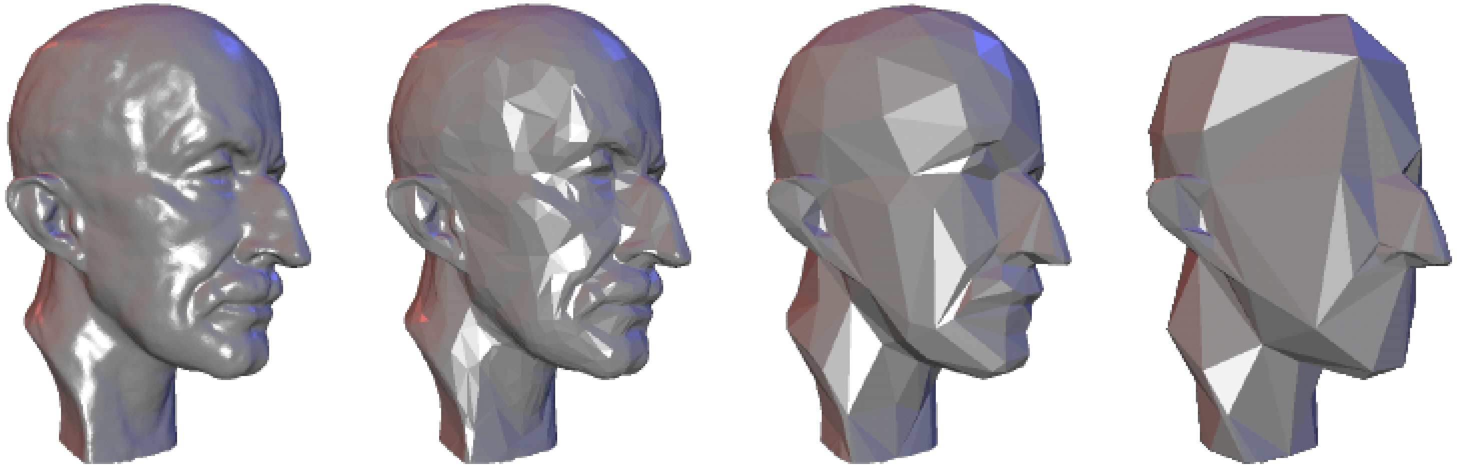
Geometric Design:

Mesh Simplification
Level-of-Detail Rendering

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

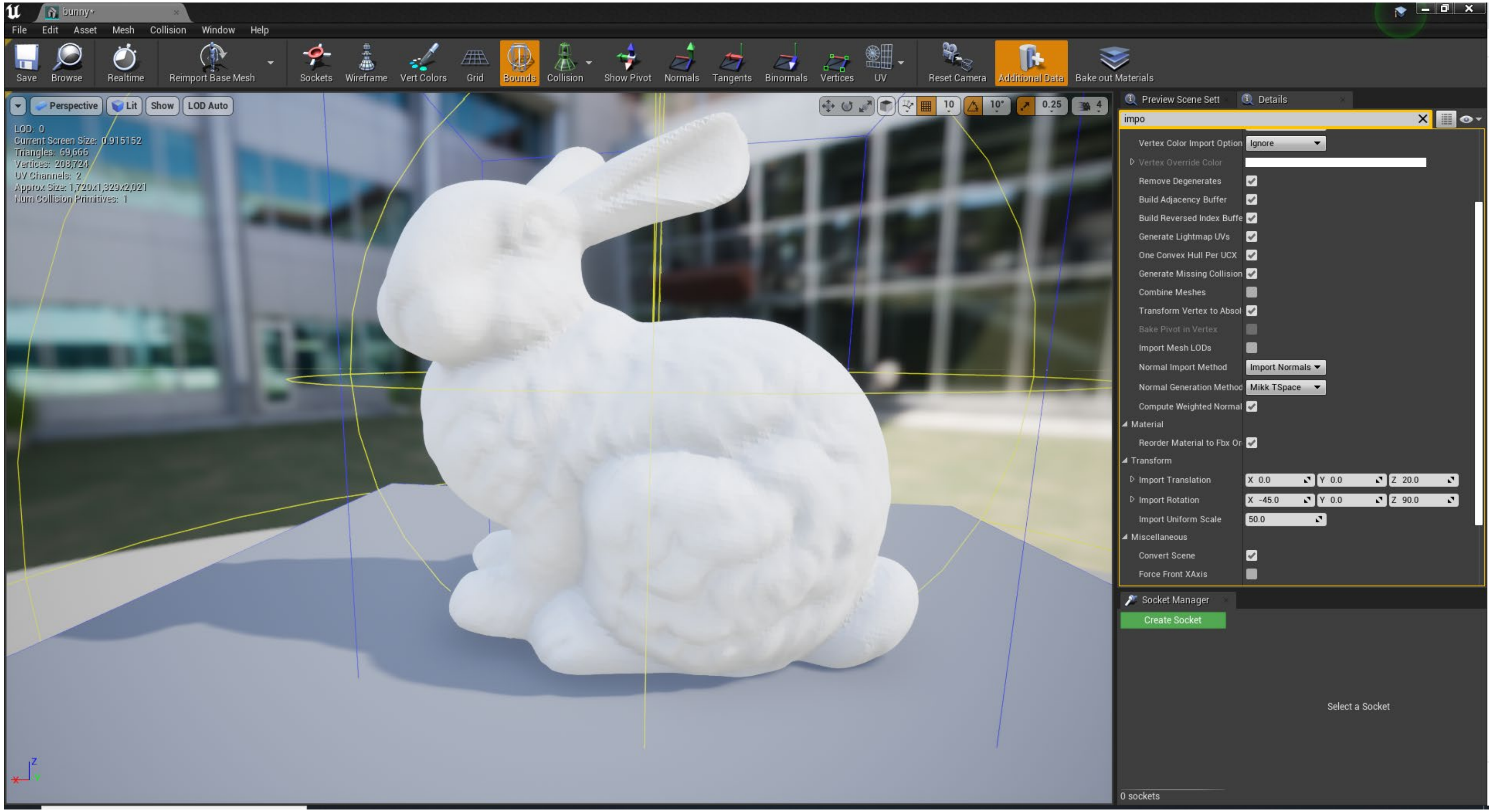


Pick which model to use based on approximate screen size…

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



LOD: 0
Current Screen Size: 0.915152
Triangles: 69,666
Vertices: 208,724
UV Channels: 2
Approx Size: 1,720x1,329x2,021
Num Collision Primitives: 1

impo

Vertex Color Import Option: Ignore
Vertex Override Color
Remove Degenerates ☑
Build Adjacency Buffer ☑
Build Reversed Index Buffe ☑
Generate Lightmap UVs ☑
One Convex Hull Per UCX ☑
Generate Missing Collision ☑
Combine Meshes □
Transform Vertex to Absol ☑
Bake Pivot in Vertex □
Import Mesh LODs □
Normal Import Method: Import Normals
Normal Generation Method: Mikk TSpace
Compute Weighted Normal ☑

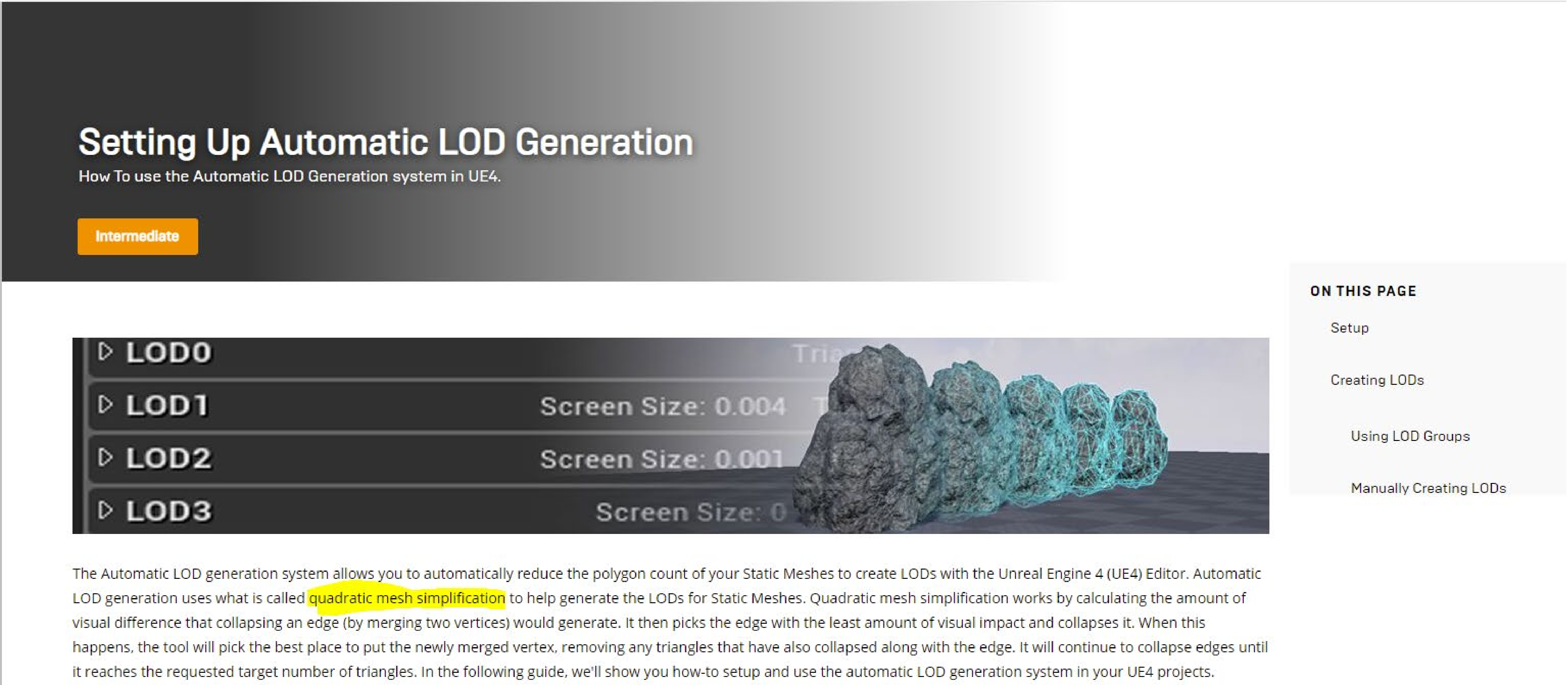
Material
Reorder Material to Fbx Or ☑

Transform
Import Translation: X 0.0 Y 0.0 Z 20.0
Import Rotation: X -45.0 Y 0.0 Z 90.0
Import Uniform Scale: 50.0

Miscellaneous
Convert Scene ☑
Force Front XAxis □

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

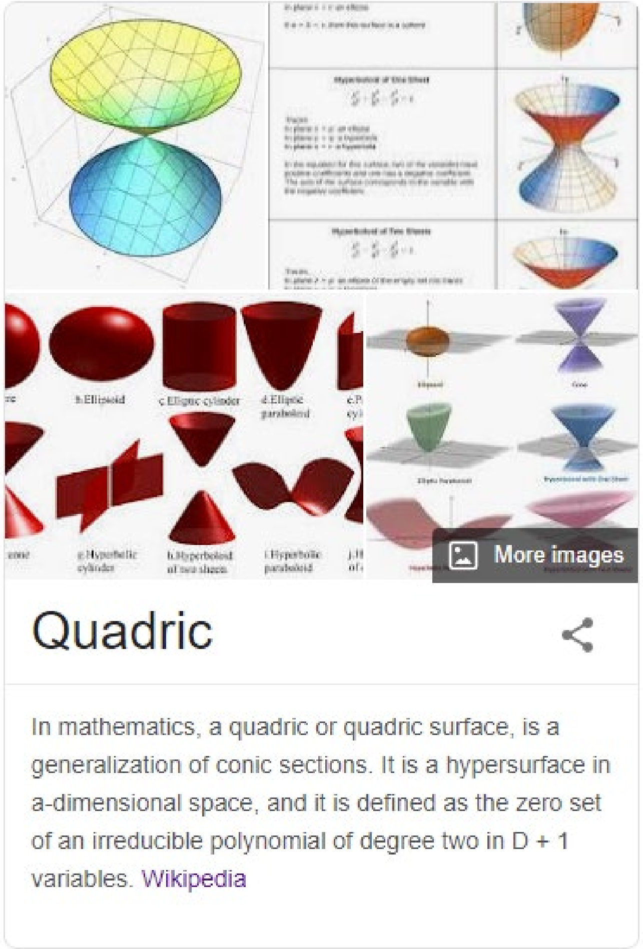
ON THIS PAGE

- Setup
- Creating LODs
  - Using LOD Groups
  - Manually Creating LODs

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.

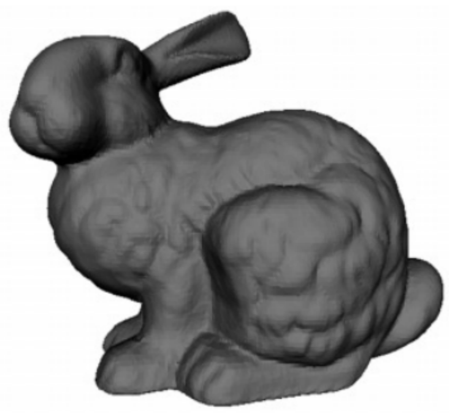
That’s wrong - should be quadric mesh simplification

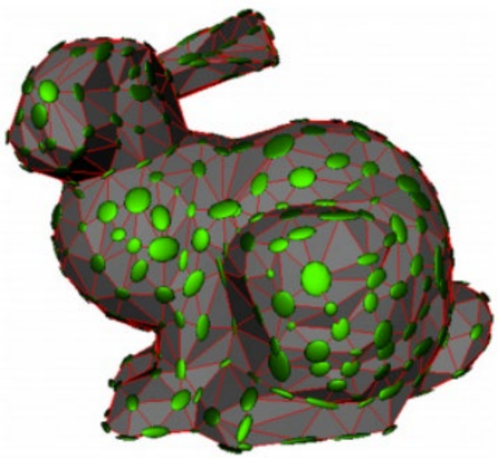
What is a Quadric?



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

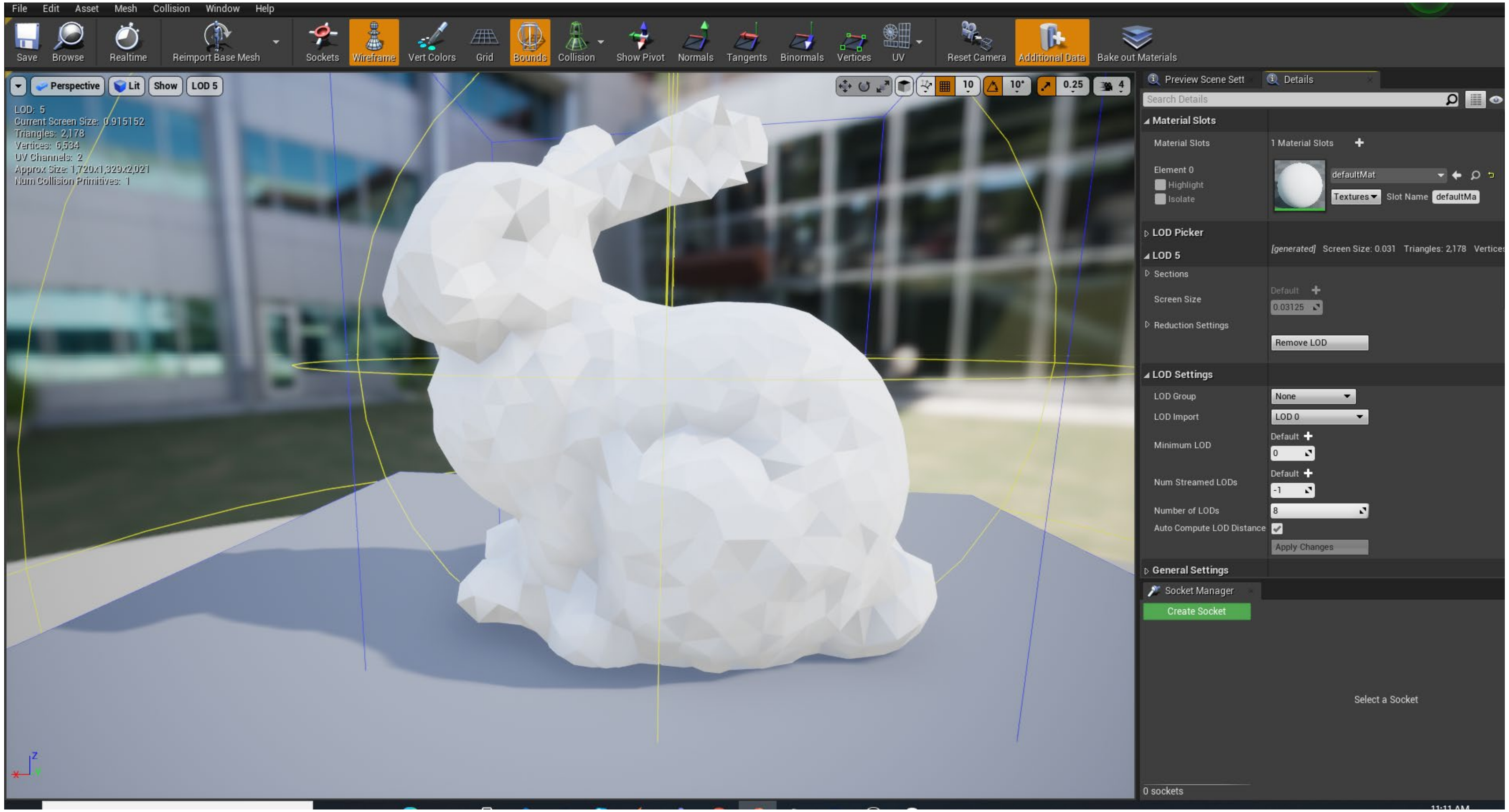




Error ellipsoids at each vertex are a type of measurement…

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

Bunny LOD 5

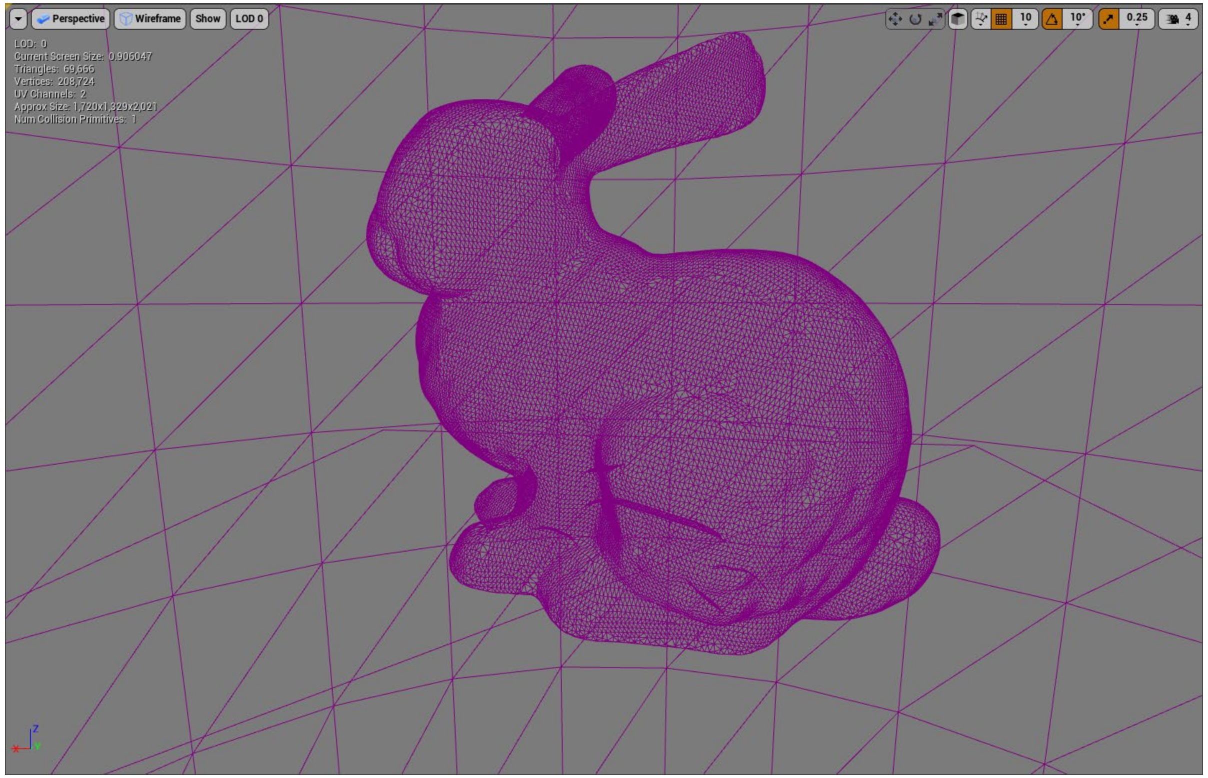


LOD: 5
Current Screen Size: 0.915152
Triangles: 2,178
Vertices: 6,534
UV Channels: 2
Approx Size: 1,720x1,329x2,021
Num Collision Primitives: 1

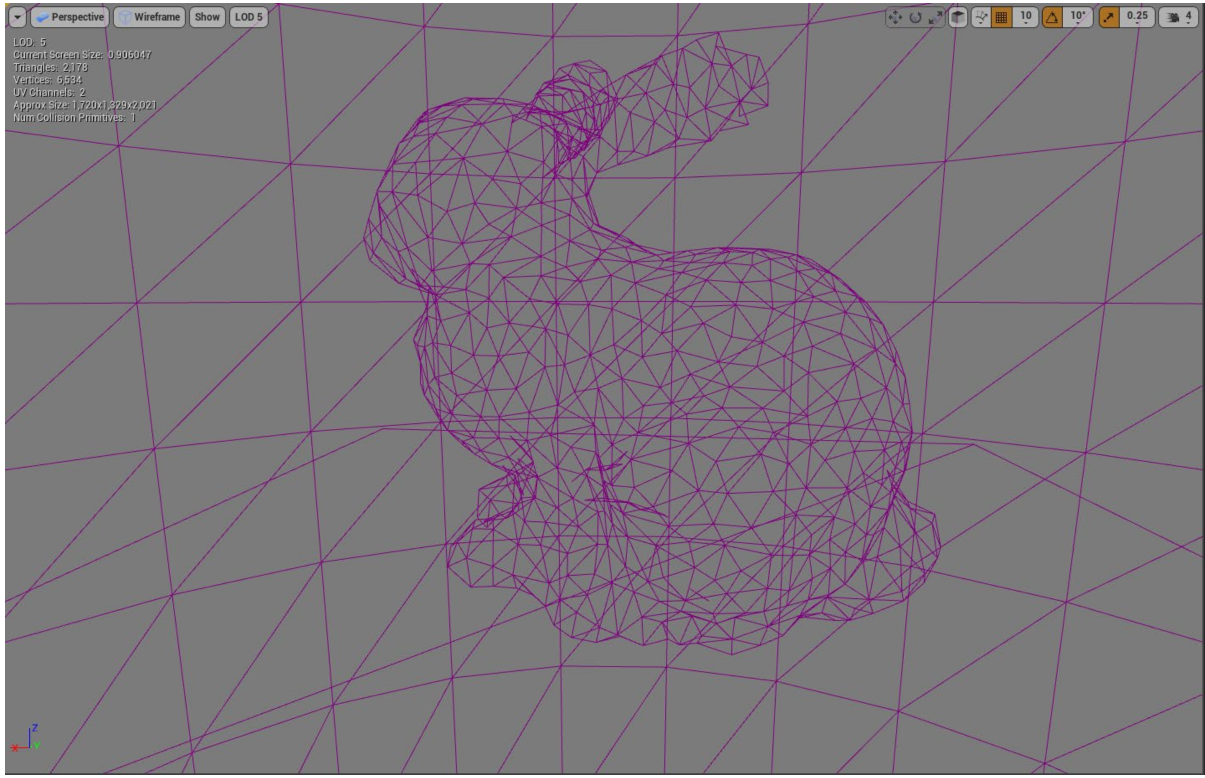
Material Slots: 1 Material Slots
Element 0: defaultMat
Slot Name: defaultMa
[generated] Screen Size: 0.031 Triangles: 2,178 Vertices
Screen Size: 0.03125
LOD Group: None
LOD Import: LOD 0
Minimum LOD: 0
Num Streamed LODs: -1
Number of LODs: 8
Auto Compute LOD Distance: ☑

Select a Socket
0 sockets

Comparison: LOD0 vs LOD5



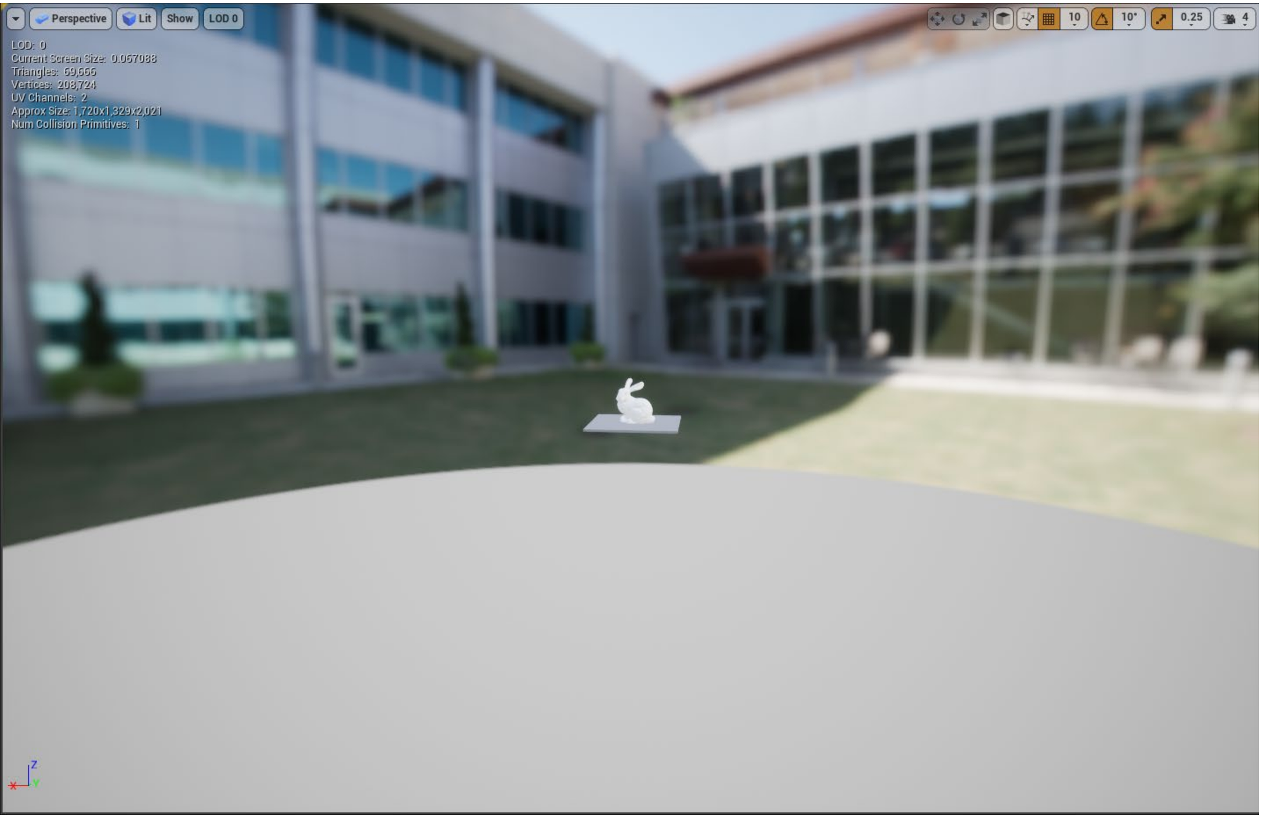
```
LOD:  0
Current Screen Size:  0.906047
Triangles:  69,666
Vertices:  208,724
UV Channels:  2
Approx Size: 1,720x1,329x2,021
Num Collision Primitives:  1
```



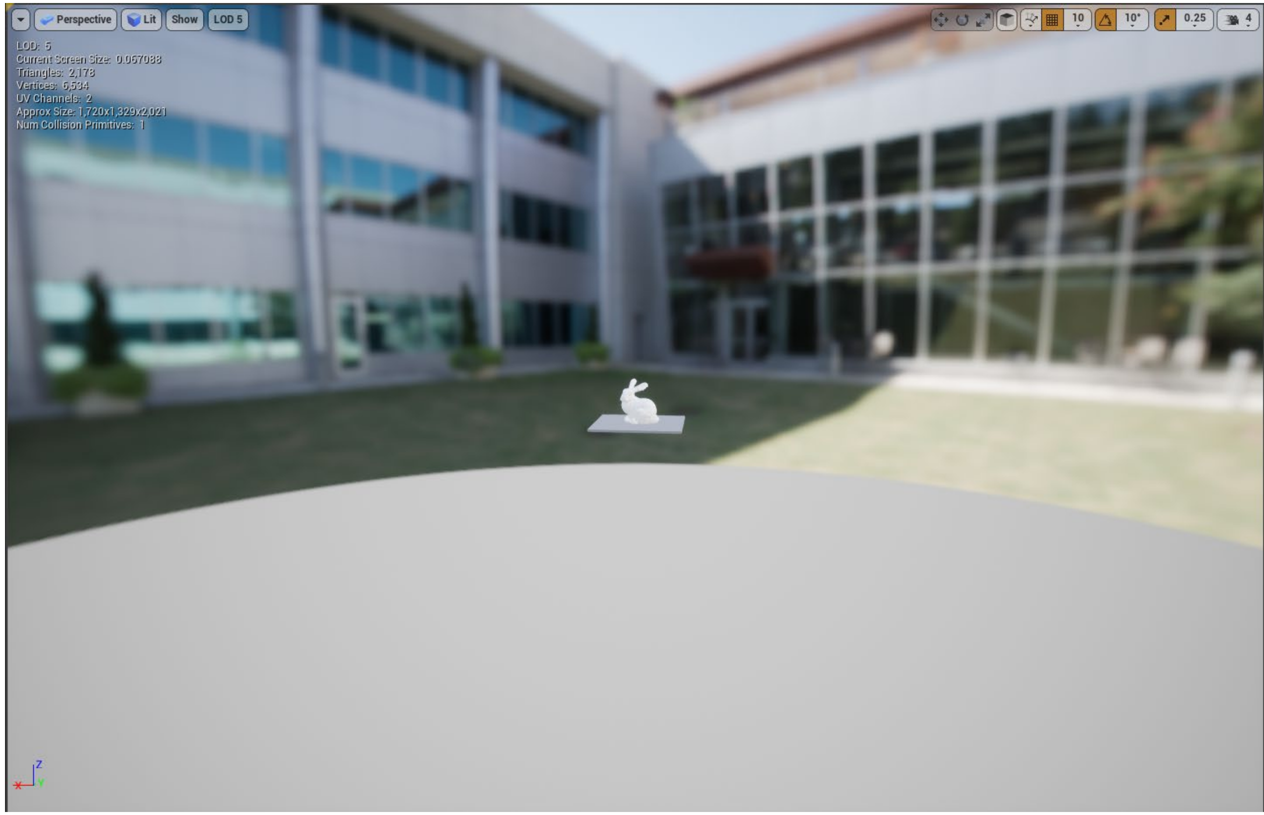
```
LOD:  5
Current Screen Size:  0.906047
Triangles:  2,178
Vertices:  6,534
UV Channels:  2
Approx Size: 1,720x1,329x2,021
Num Collision Primitives:  1
```

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

Rendering: LOD0 vs LOD5



```
LOD:  0
Current Screen Size:  0.067088
Triangles:  69,666
Vertices:  208,724
UV Channels:  2
Approx Size: 1,720x1,329x2,021
Num Collision Primitives:  1
```



```
LOD:  5
Current Screen Size:  0.067088
Triangles:  2,178
Vertices:  6,534
UV Channels:  2
Approx Size: 1,720x1,329x2,021
Num Collision Primitives:  1
```

Error Quadrics

- Squared distance to plane

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

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

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

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

Error Quadrics

- Sum distances to vertex' planes

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

- Point that minimizes the error

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} p^* = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

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.

Vertex Placement

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

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

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

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

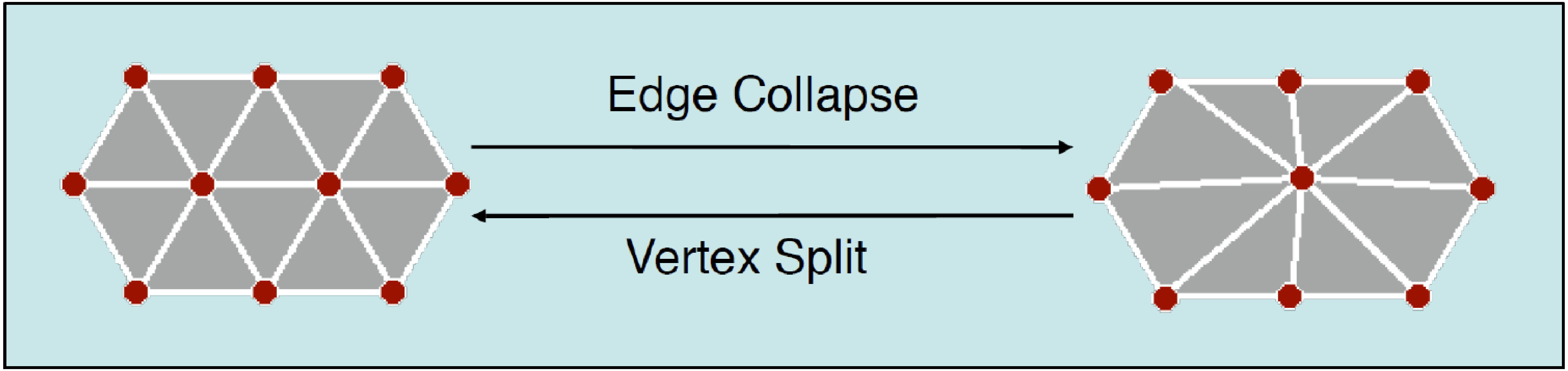
The gradient of $Q(v)$ is

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

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

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

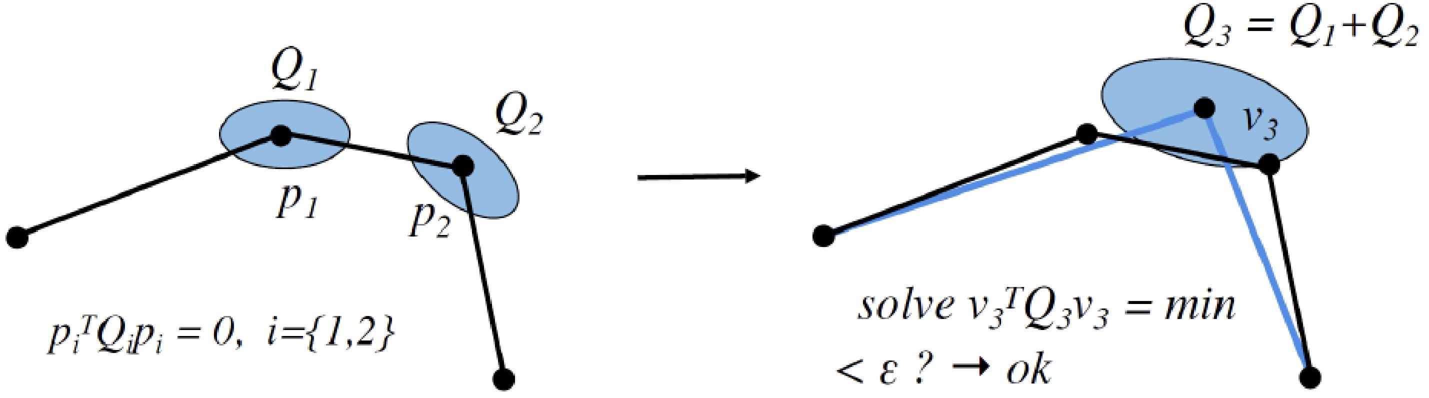
Incremental Simplification: Edge Collapse



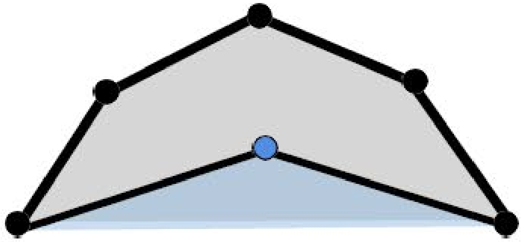
- Merge two adjacent triangles
- Define new vertex position

Incremental Simplification Algorithm

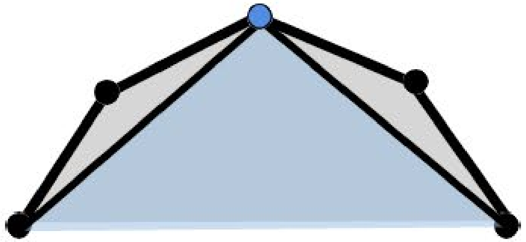
1. Compute Quadric for each vertex
2. Create a priority queue of all possible edge collapses $p_1 + p_2$
    1. For each edge collapse compute $Q_3 = Q_1 + Q_2$
    2. Compute new vertex v_3
    3. Compute error $v_3^T Q_3 v_3$
3. Choose collapse with least error…update quadrics and repeat



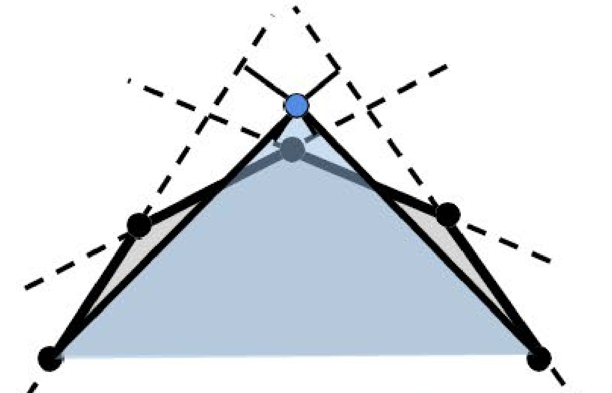
Comparison of Vertex Placements



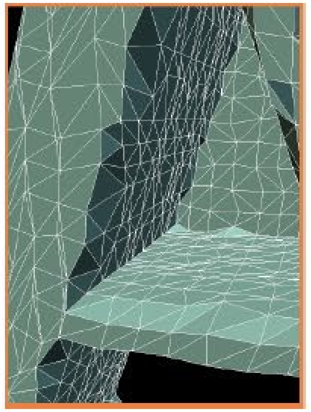
average

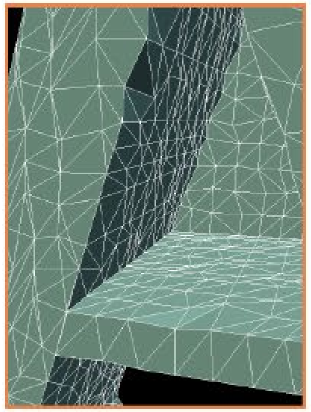


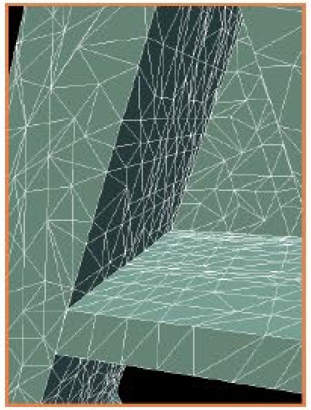
median



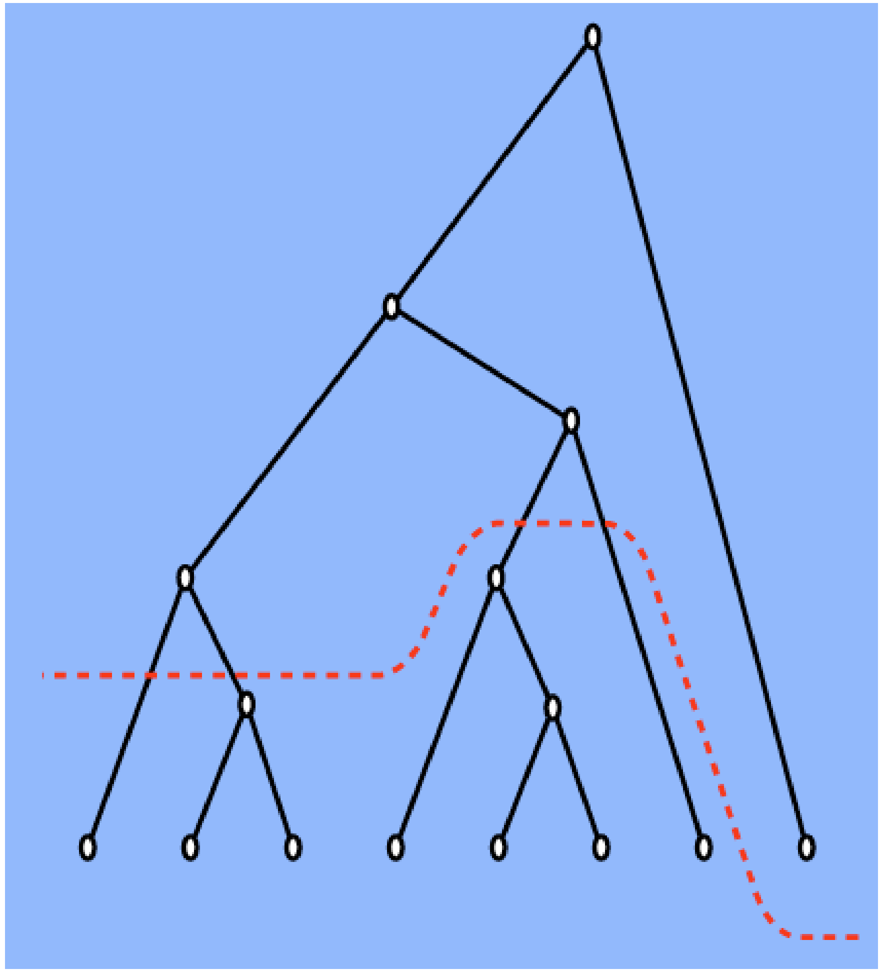
error quadric







Continuous LOD using Vertex Hierarchies



Each original vertex in mesh is a leaf in this diagram

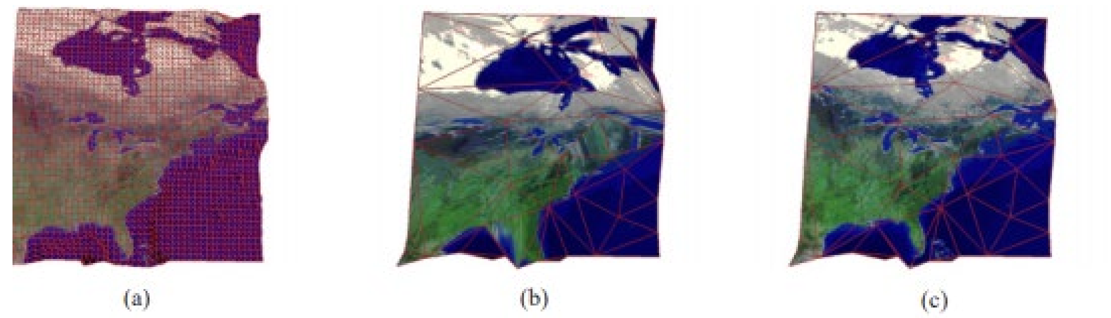
An edge contraction makes vertices siblings and creates a parent

A cut through the tree represents 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…

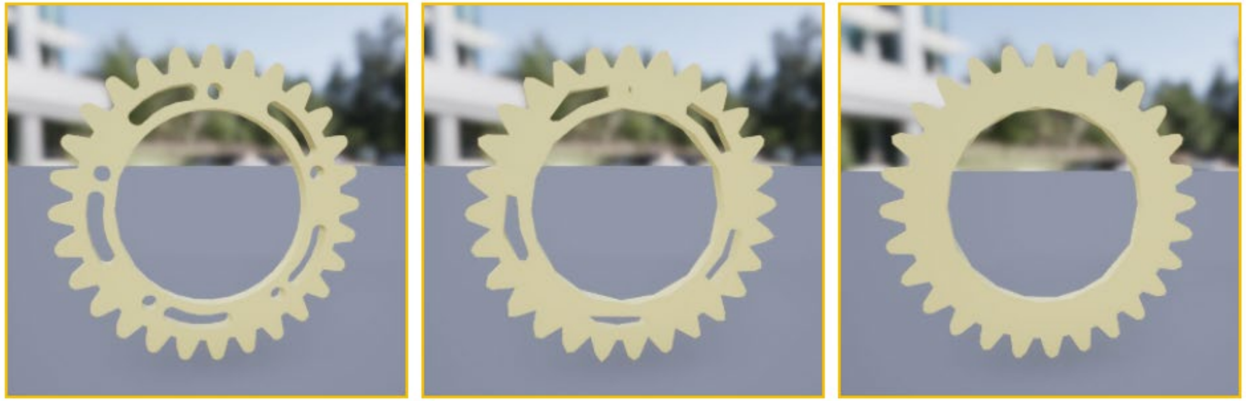
In practice LOD gets more complicated

- 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