CS 563 Advanced Topics in Computer Graphics

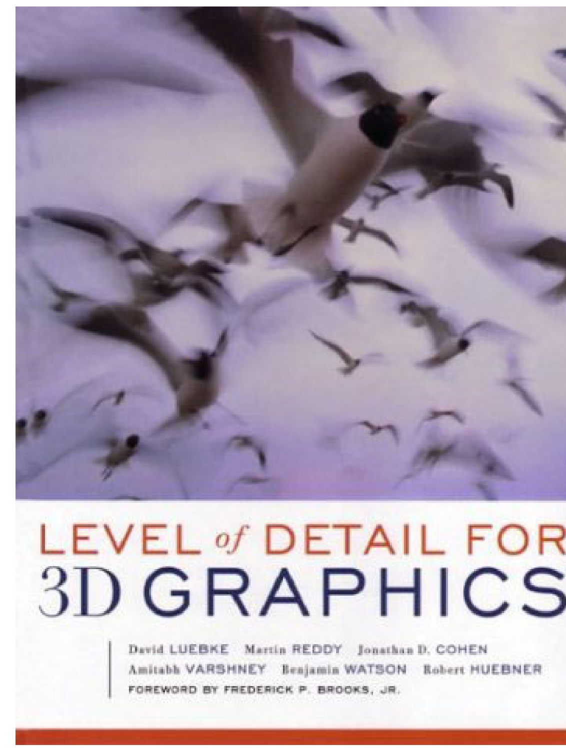
Level Of Detail

Cliff Lindsay

[the ubiquitous Stanford bunny]

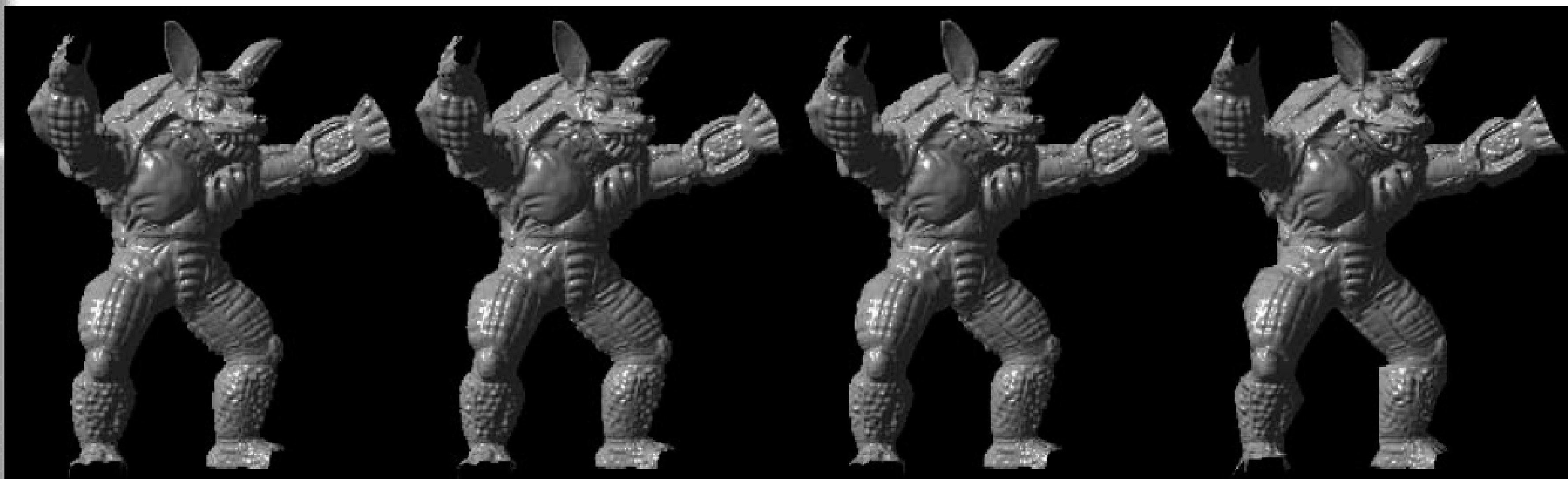
Topics Covered:

- L.O.D. Basics
  - Generating
  - Selection
  - Switching
- Case Study
  - Terrain Rendering: ROAM



[Above is a image of the definitive guide for L.O.D. from which this presentations borrows heavily]

Example 1



[Image courtesy of L.O.D For 3D Graphics, pg 81]

L.O.D. Basic Ideas

Main Idea: L.O.D. is concerned with using simplified versions of an object as an objects details become less visible.

Highlights:

1. Generation
   - Local Geometry operators
   - Topological Operators
   - Global Geometry operators
2. Selection
   - Factors (distance, size, priority, etc.)
3. Switching
4. Error Measurement

Example 2

Original Model 250,000 tris

Orig Vs. 7809 triangles

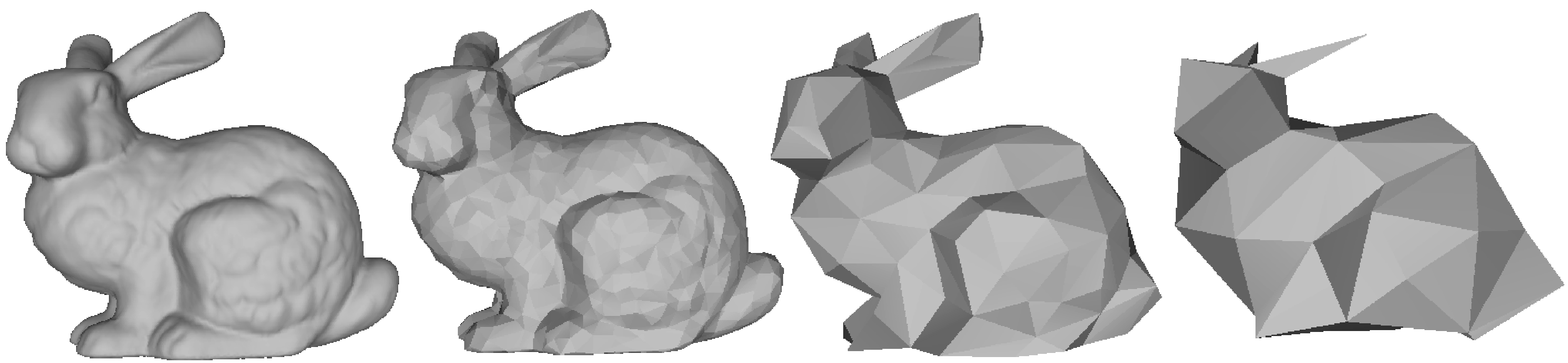
Vs. 975 tris

[images courtesy Cohen 99]

Generation – *Where do we get Simplified models?*

Techniques :

- *Local* Simplification
- *Topological* Simplification
- *Global* Simplification



69,451 polys   2,502 polys   251 polys   76 polys

[Image courtesy of LOD for 3D Graphics, pg. 5]

Some Assumptions

Assumptions For This presentation

(not a general assumption)

- Talk about geometry is a set of **Triangles** (no quads, or odd Shaped geometry, Edges = 3, Vertices = 3)
- No holes for local operators
- Some cases require a user prescribed fidelity
- May or may not have an inverse operation

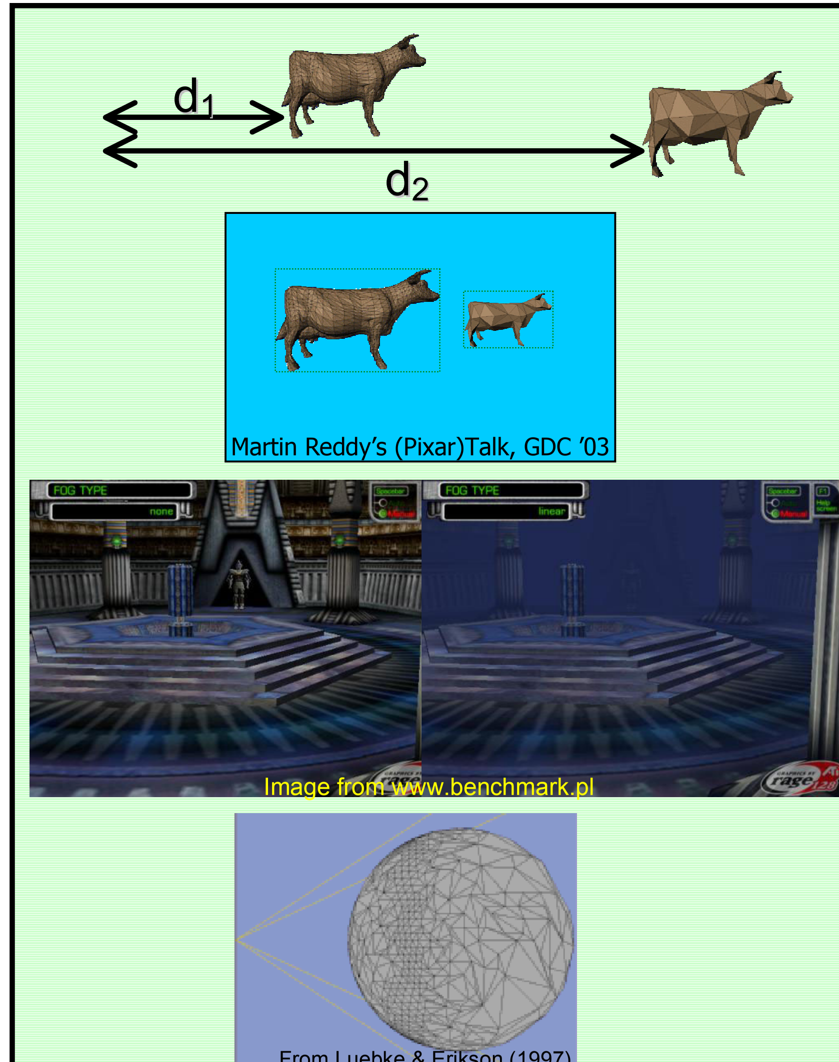
Static Vs. Dynamic L.O.D.

Static	Dynamic
Preprocess	Partial preprocess, partial runtime
Discrete Meshes	Create continuous meshes during r.t.
Minimal Run-time Processing (selection, switching)	More run-time computation
Retained mode rendering	Immediate mode rendering
Selection Criteria Bounded (view independent)	Can incorporate view-dependent tessellation
Visible Artifacts (cracks, popping)	Smooth Blending

Situation that call for a reduction in detail

When to use L.O.D:

- Distance
- Size
- Environment
- Perceptual



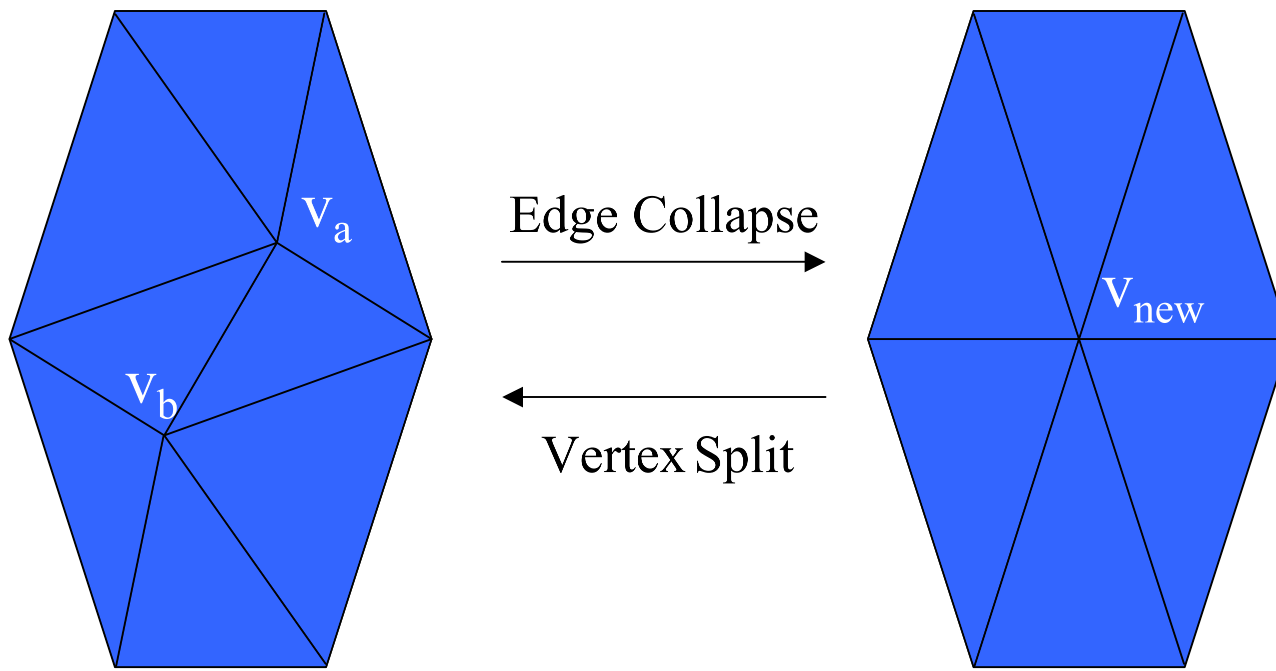
Various Local Operator Techniques

- Edge Collapse
- Vertex Pair Collapse
- Vertex Removal
- Triangle Collapse

Local Operators: Edge Collapse

Edge Collapse

<u>Technique</u>: Collapse edge to a single vertex. Inverse operation splits a vertex and creates an edge



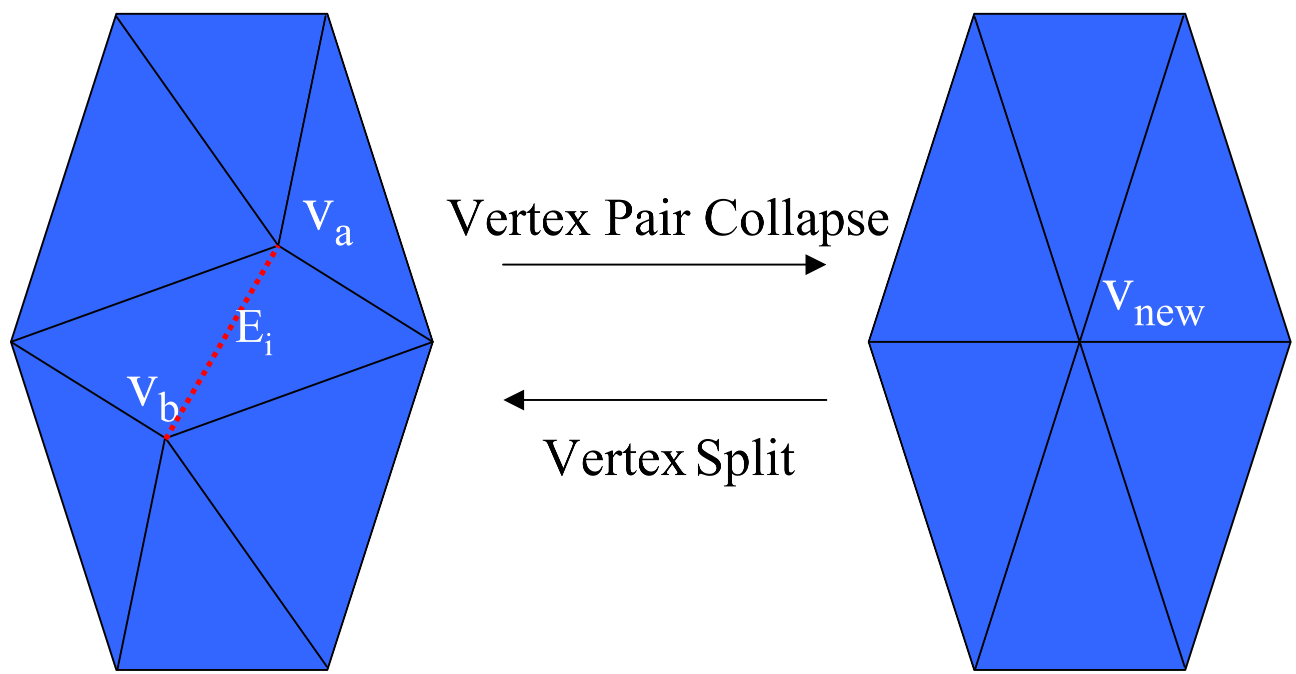
[Image courtesy of LOD for 3D Graphics, pg. 21]

Local Operators: Vertex Pair Collapse

Vertex Pair Collapse (virtual edge collapse)

Technique: Collapse two unconnected vertices

- Edge E_i is considered a “virtual edge”

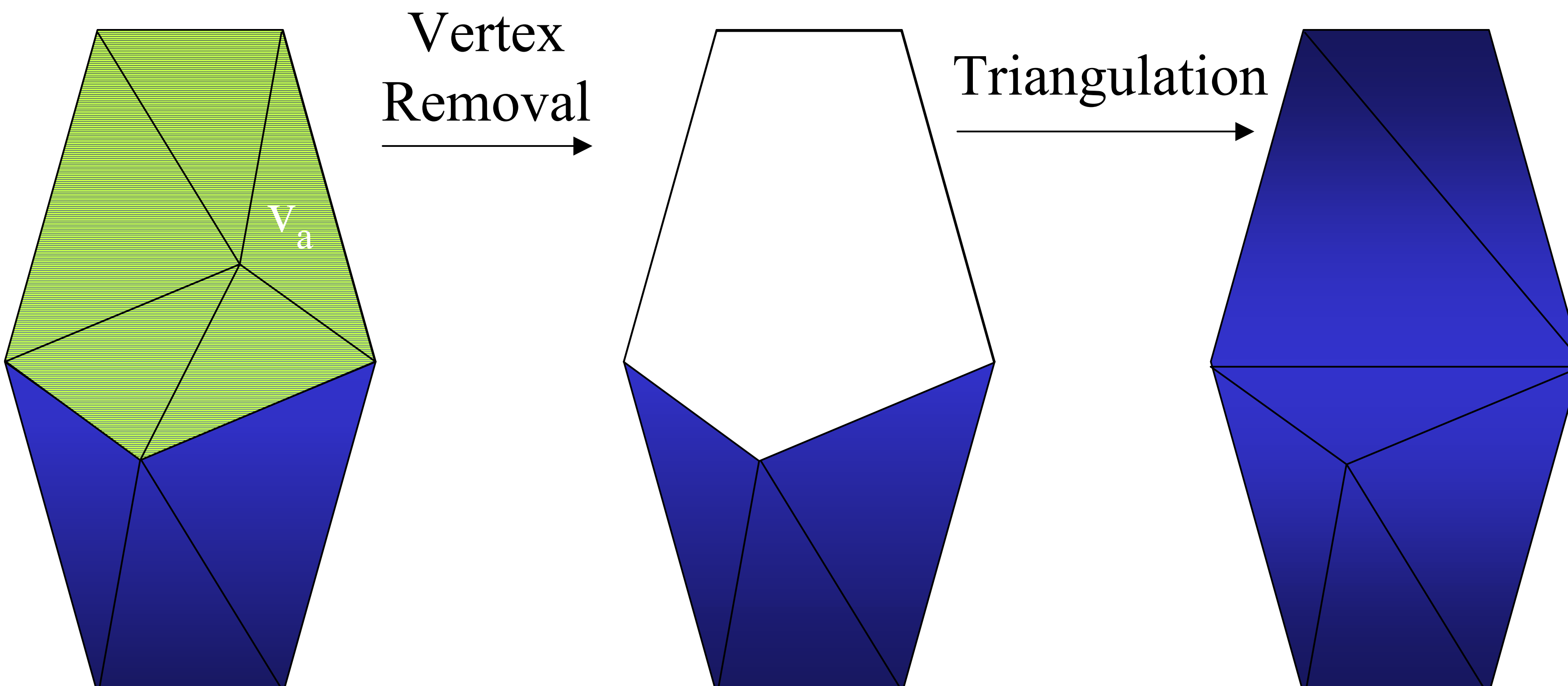


[Image courtesy of LOD for 3D Graphics, pg. 23]

Local Operators: Vertex Removal

Vertex Removal

<u>Technique</u>: The vertex removal operator removes a vertex, along with its adjacent edges and triangles, and triangulates the resulting hole



[image courtesy of LOD for 3D Graphics, pg. 26]

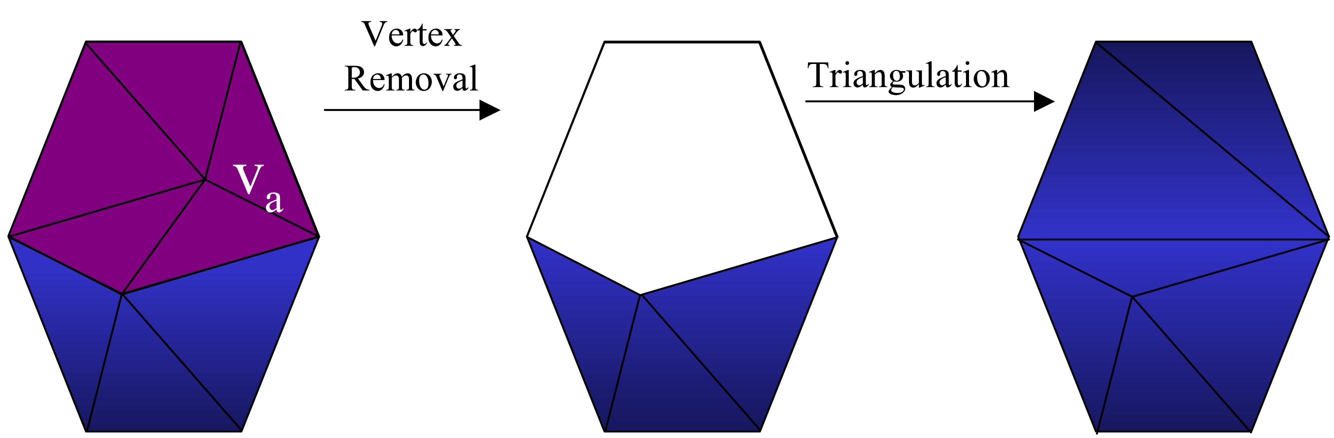
Local Operators: Vertex Removal Steps

Vertex Removal Steps:

a) A vertex is chosen for removal.

b) The vertex and all surrounding triangles are removed, leaving a hole in the mesh.

c) The hole is retriangulated with two fewer triangles than the original mesh.

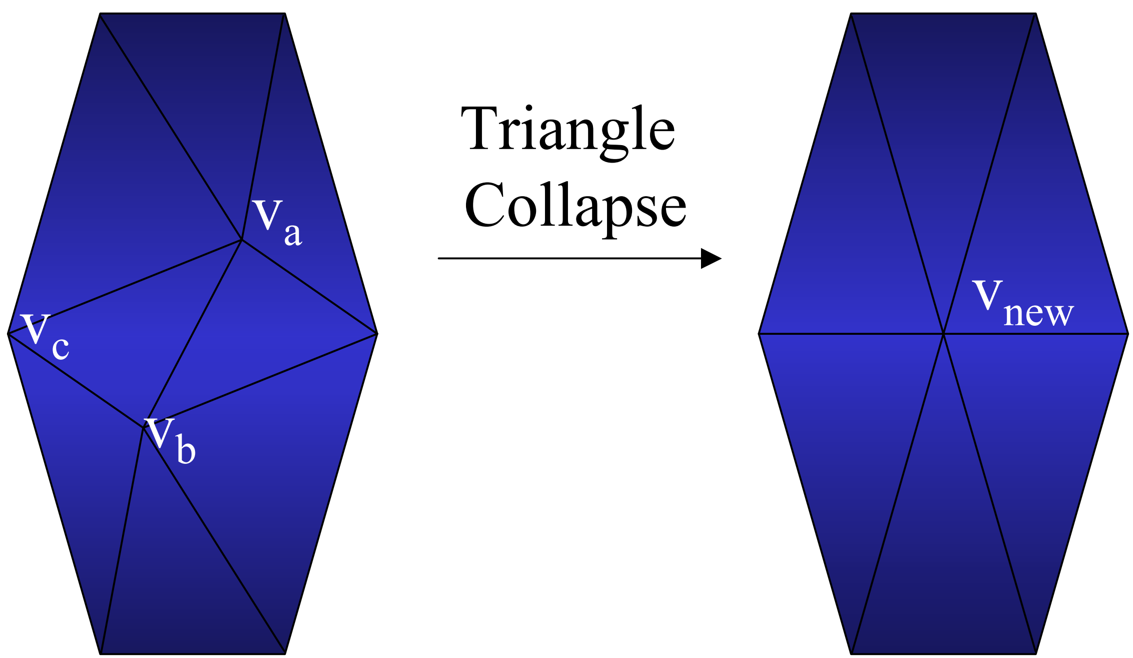


[image courtesy of LOD for 3D Graphics, pg. 26]

Local Operators: Triangle Collapse

Triangle Collapse

<u>Technique</u>: A triangle collapse operator simplifies a mesh by collapsing a triangle *(va, vb, vc)* to a single vertex



[image courtesy of LOD for 3D Graphics, pg. 24]

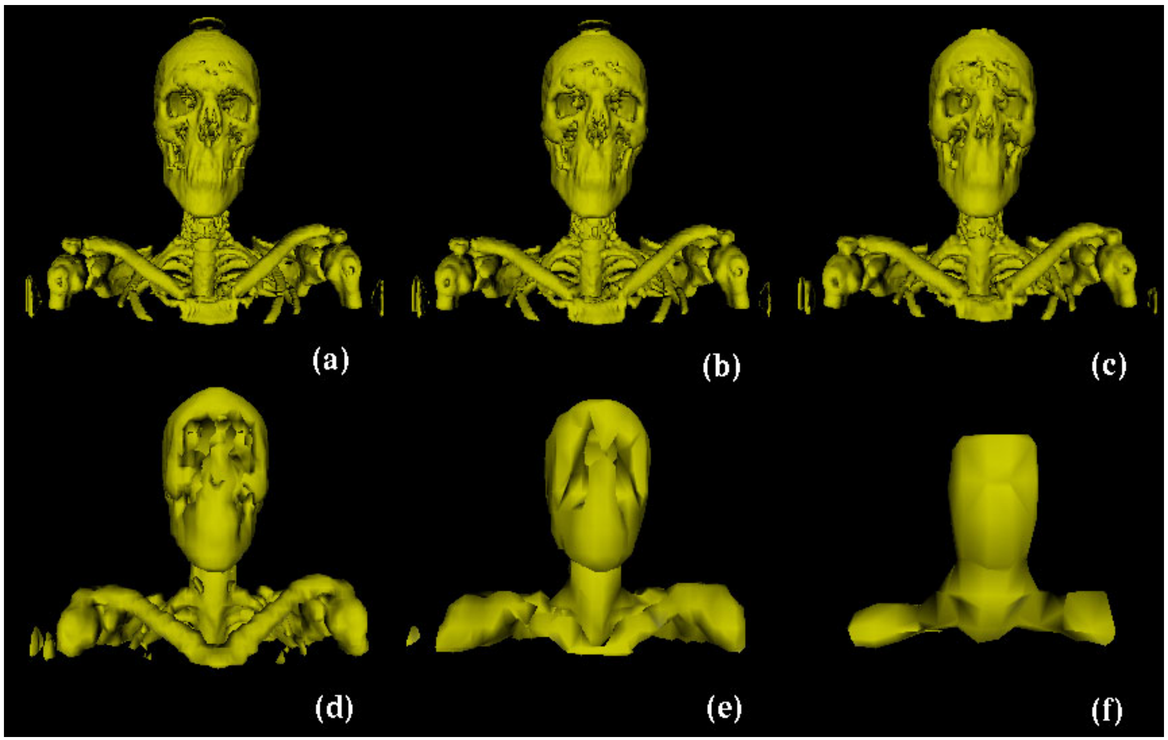
Global Operators

Various Global Operators

- Alpha-Hull
- Volume Processing
- Others (***not mentioned in this talk, but they're out there***)

Global Operators: Volume Processing

Low Pass Filtering: Removing Detail Via Signal Processing



Generating L.O.D. Review

Techniques :

- *Local* Simplification
- *Topological* Simplification
- *Global* Simplification

Generalizations:

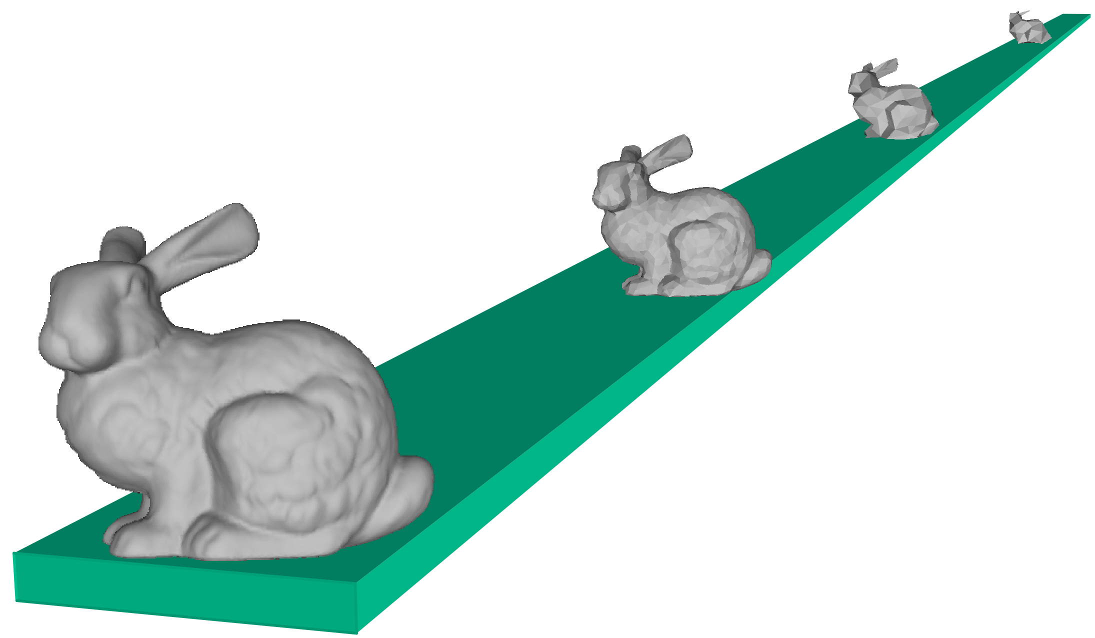
- Most Applications use static L.O.D. (discrete meshes).
- Local operators are more popular than Global
- Generating L.O.D. is not the end of the story

Selection Factors for L.O.D.

When do we use our generated L.O.D. models?

<u>Potential Factors:</u>

- Distance
- Size
- Priority
- Environment
- Perception
- More …

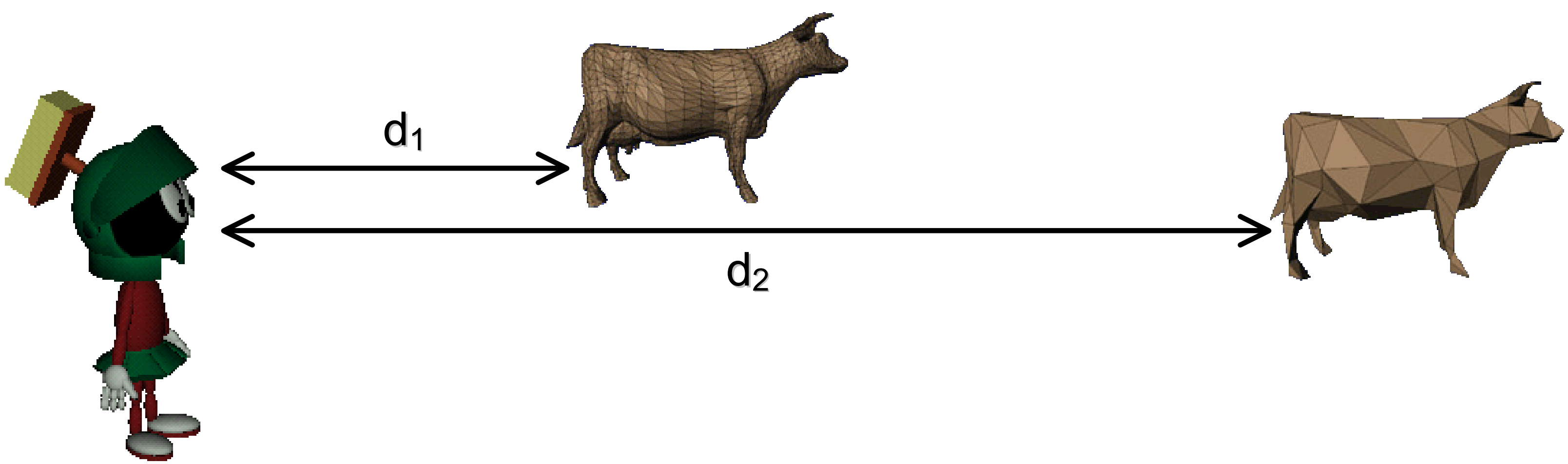


Selection Factor: Distance

Main idea: Distance Metric to determine what model resolution to use.

Attributes:

- Easiest to implement
- Intuitive
- Efficient



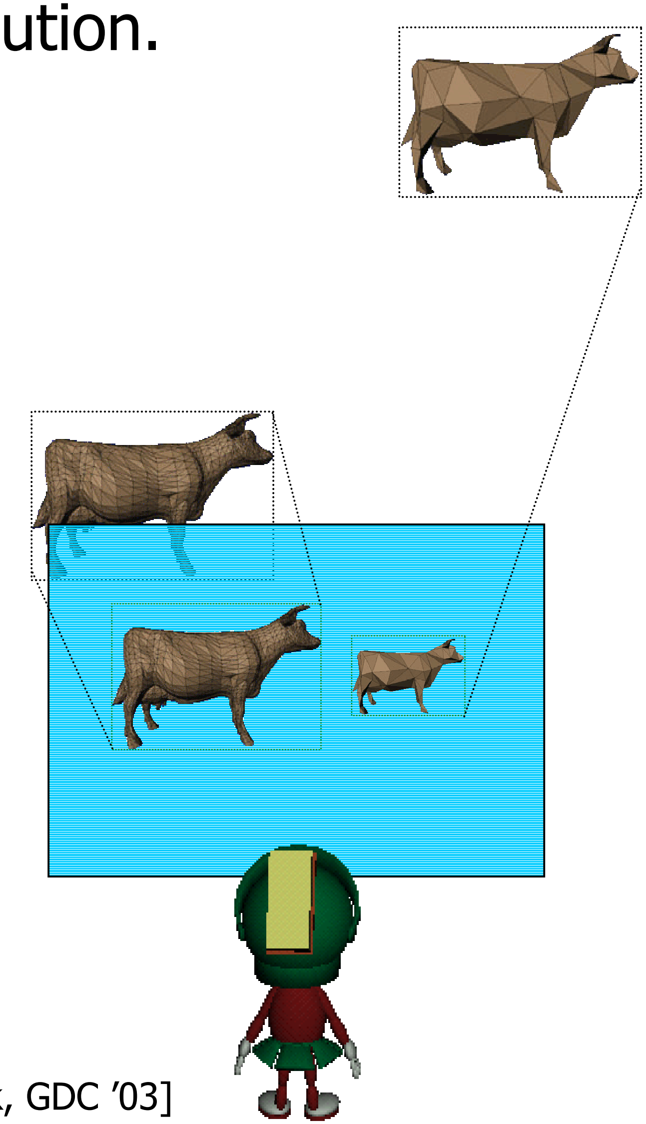
[image borrowed from Martin Reddy's (Pixar)Talk, GDC '03]

Selection Factor: Size

<u>Main idea:</u> Uses screen space to determine size of model and adjust model resolution.

<u>Attributes:</u>

- insensitive to display resolution, object scaling, or field of view.
- More accurate than distance
- Less efficient than distance



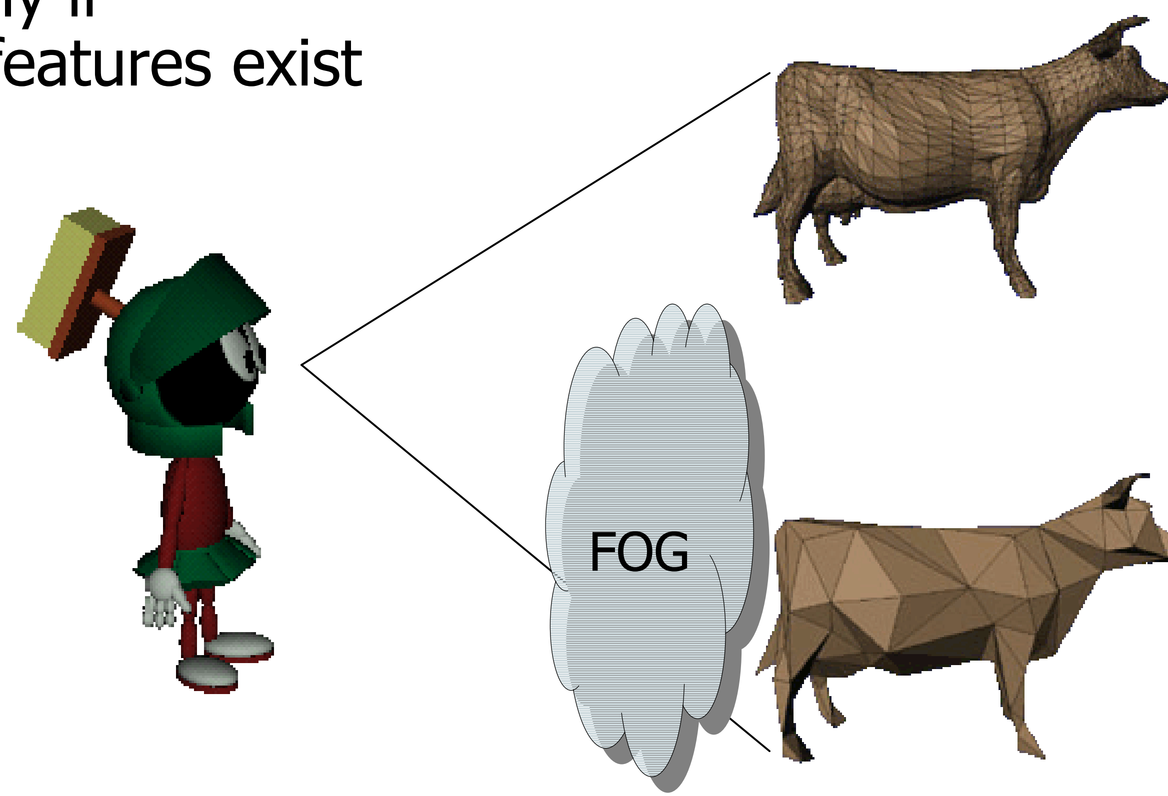
[image borrowed from Martin Reddy's (Pixar)Talk, GDC '03]

Selection Factor: Environmental conditions

<u>Main idea:</u> Use environmental features (smoke, fog, haze, fire, etc.) to mask detail of objects inorder to lower resolution with out loss of realism.

<u>Attributes:</u>

- Can be used only if environmental features exist
- Can be artificial

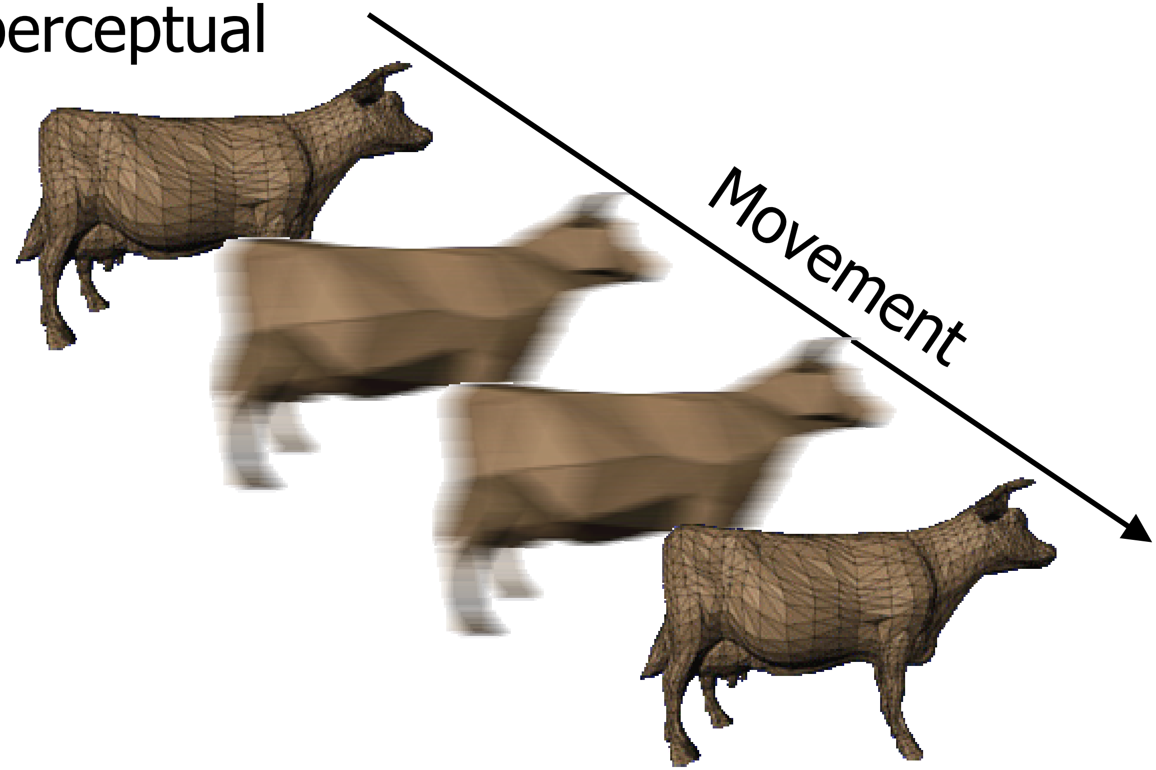


Selection Factor: Perceptual

Main idea: Use perceptual factors to our advantage to lower model resolution. I.E., motion, peripheral, silhouettes, etc.

Attributes:

- Efficient, but extra complexity
- Only be used in perceptual context

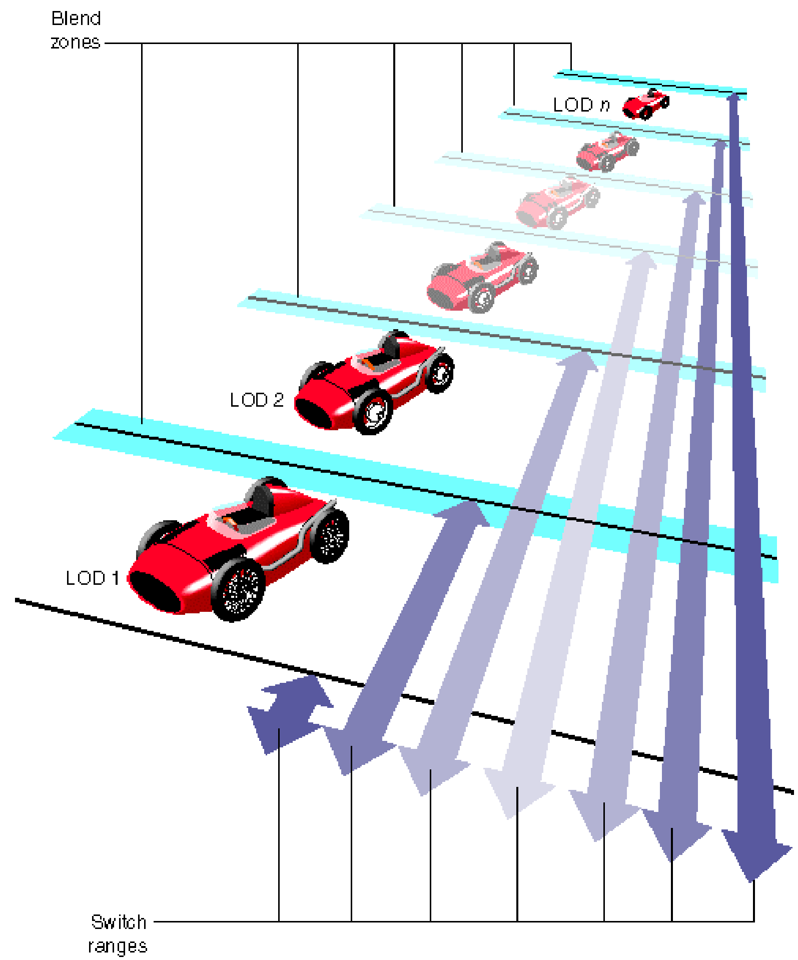


Selection Factors Review

Review: Selection factors are the run-time metrics used to determine how we choose a desired geometry resolution

Factors:

- Distance
- Size
- Priority
- Environment
- Perception
- More …



[Image courtesy of OpenGL Performer: Click Here To Go To Page!]

Switching Geometry

Main Idea: Switch between L.O.D with out noticeable artifacts.

Some Example of Artifacts:

- Popping of geometry (**BIG** problem in terrain rendering)
- Obvious upgrade or downgrade of detail
- Flickering near threshold

Preventative Techniques:

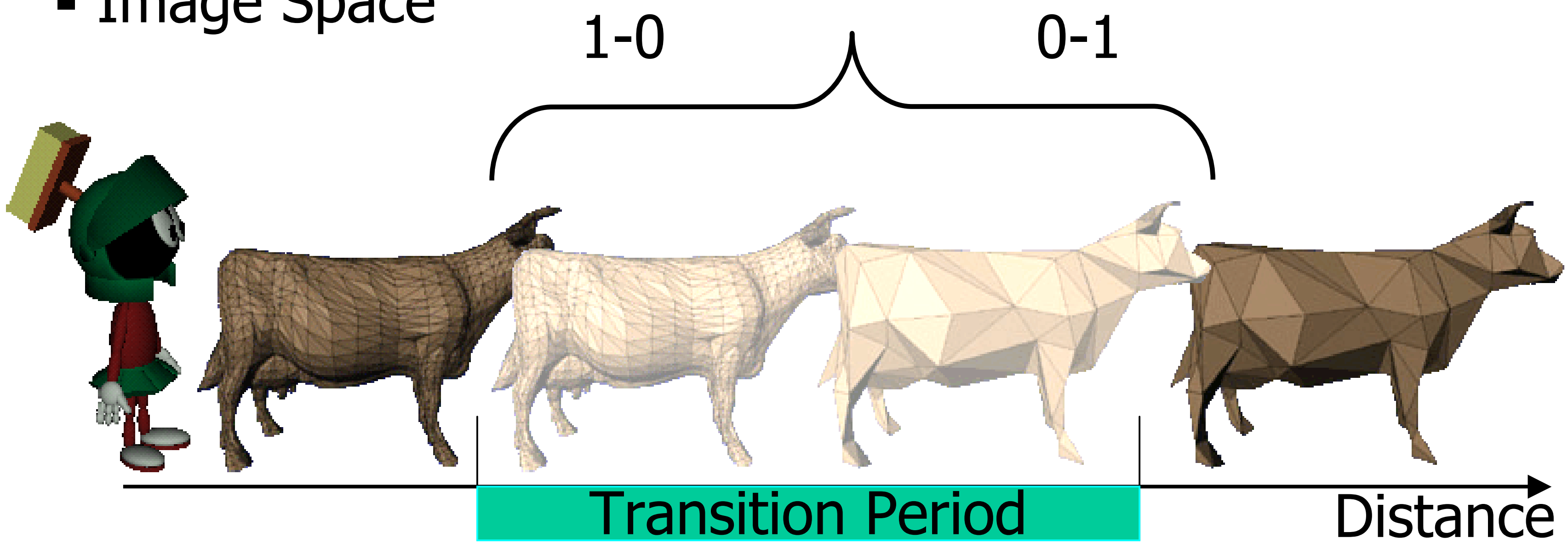
- Alpha Blending
- Geomorphs
- Hysteresis

Alpha Blending

<u>Main idea:</u> Use alpha blending to transition to a different resolution. Normally done during a small transition period.

<u>Attributes:</u>

- Two versions of object during transition
- Potential self-occlusion artifacts
- Implemented in most GPU hardware
- Image Space

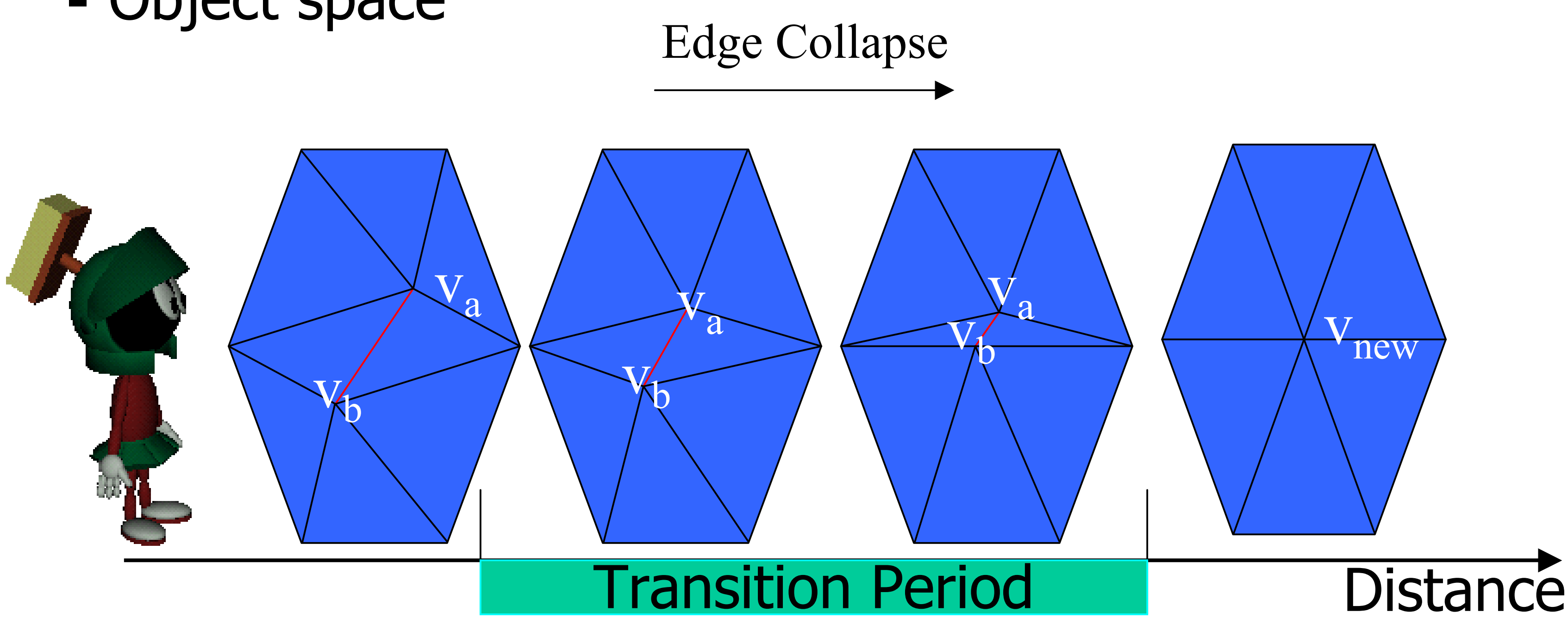


Geomorphs

Main idea: Linearly interpolates vertex and edge decimation and creation.

Attributes:

- Low error
- Complex technique, but very efficient
- Object space



[Image courtesy of LOD for 3D Graphics, pg. 21]

Geomorphs Example: Unreal

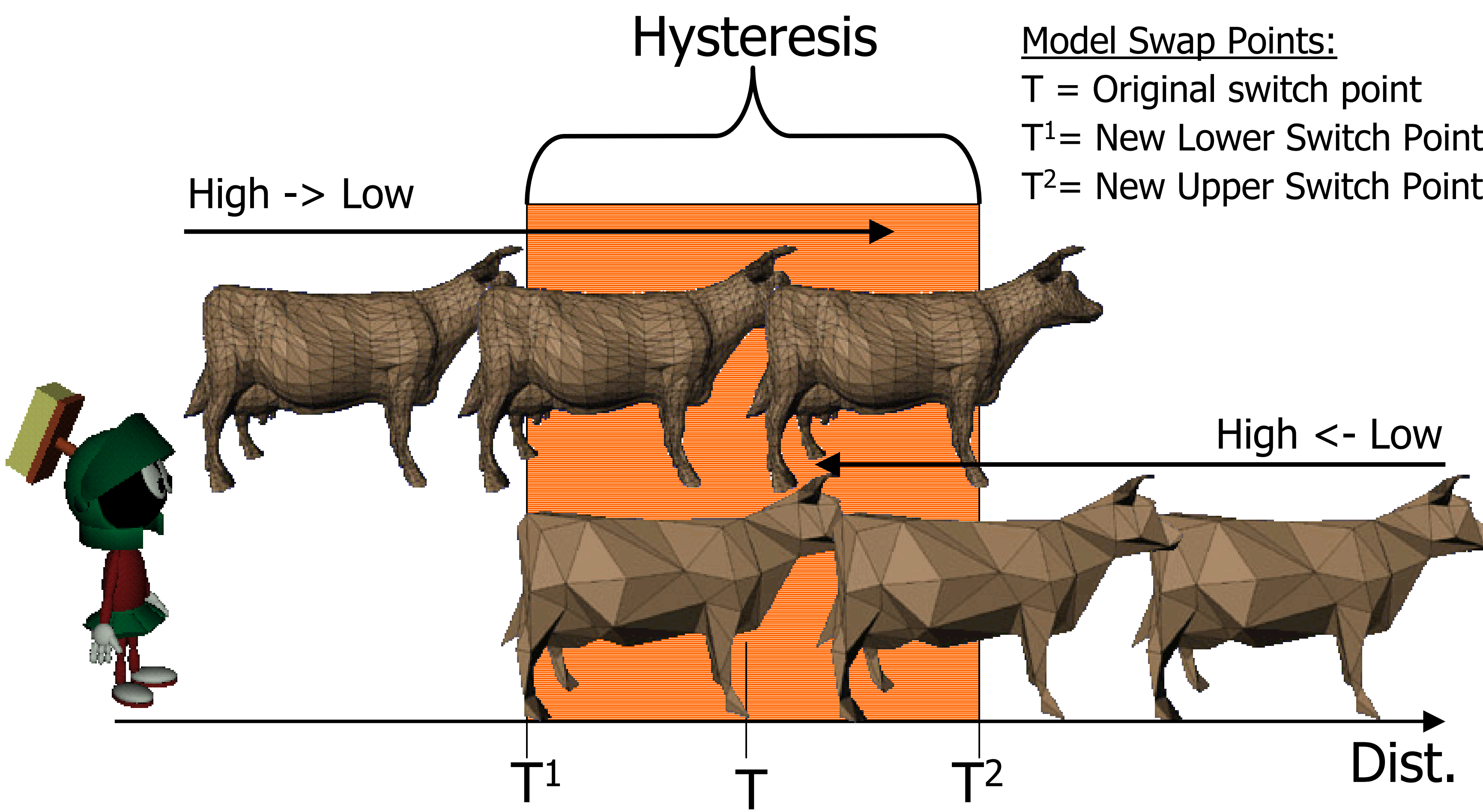
- These screenshots show an Unreal Tournament player model with an original face count of 600+ polygons (not including the weapon mesh) as its vertex count gets reduced automatically, fixed at 100%, 75%, 50% and 25%,

- Normally, these deformations are not visible like this, but continue to fall in the few-pixel-size range as the distance to the player increases.

[Image courtesy Epic, available at http://unreal.epicgames.com/Images/LODpictures.htm]

Hysteresis

<u>Main idea:</u> Use an overlap distance within the transition period to avoid quick back and forth transitions between two L.O.D. resolutions.



<u>Model Swap Points:</u>

T = Original switch point

T^1 = New Lower Switch Point

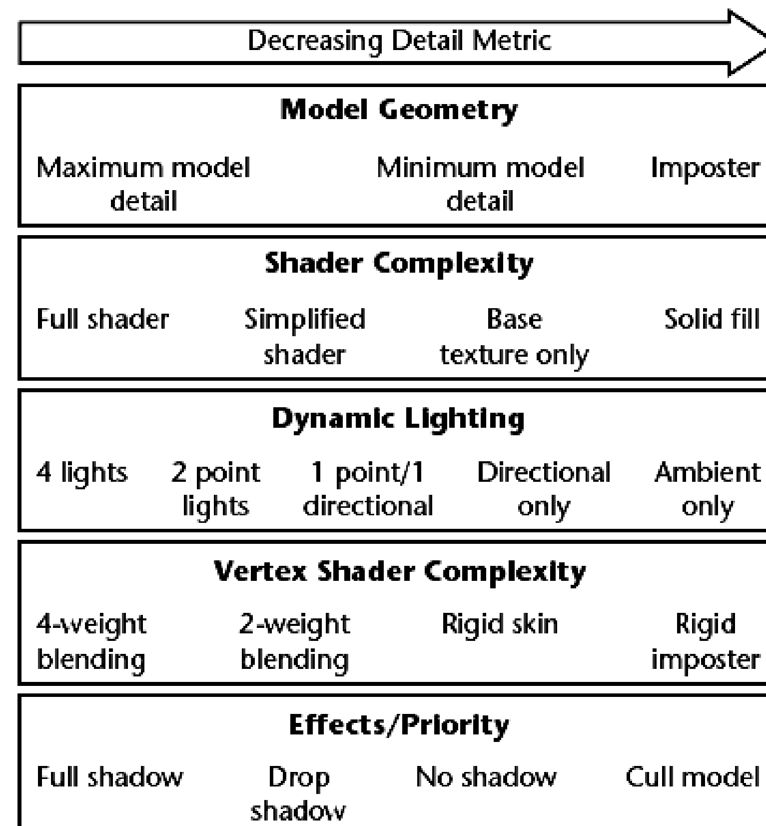
T^2 = New Upper Switch Point

Non-Geomtric L.O.D

<u>Main idea:</u> That L.O.D. is not limit to the traditional geometry simplification, anything that lowers the level of detail for the purpose of efficiency can be considered L.O.D.

Examples:

- Shader L.O.D.
- Occlusion
- Skeletal
- Lighting/reflection
- Impostors, render to texture
- Depth of Field
- Shadows
- What ever you can think of…



[Diagram from LOD for 3D Graphics, pg. 81]

Shader L.O.D.

Shader L.O.D

Pixel Shader

- Reduce # of pass effects
- Use less accurate but more efficient
- Reduce texture fetches

Vertex Shader

- Use unnormalized normals
- Use reduced complexity shaders

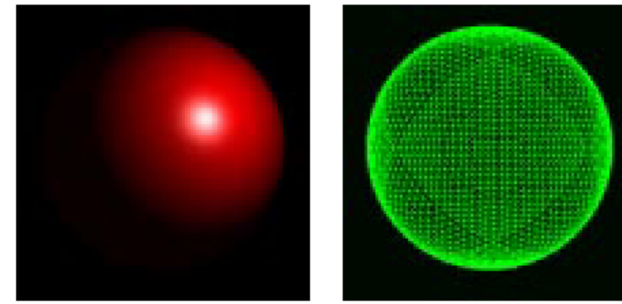
Lighting & Reflection L.O.D.

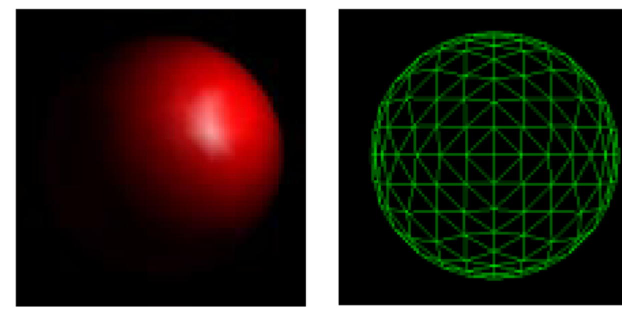
Lighting Reduction

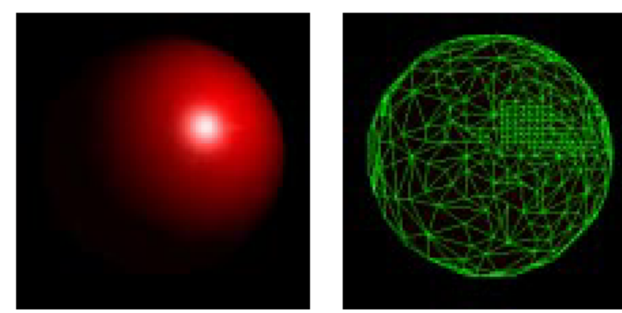
- Reduce Number of lights
- Reduce influence of point & directional lights
- Prelight: Light maps, per vertex
- Simplify light equations (envmap, ambient only, etc.)

Reflection(BRDF) Reduction

- Use less detail BRDF
- Maintain fidelity by adding polys at highlights
- Reflection calculation in different spaces


(a) Sphere with 8192 triangles (uniform LOD)


(b) Sphere with 512 triangles (uniform LOD)


(c) Sphere with 537 triangles (adaptive LOD)

[Image courtesy of GDC, Martin Reddy, Pixar Animation Studios]

Shadow L.O.D.

Shadow Complexity Reduction

- Use Model Approximate techniques:
  - Low resolution models to generate shadows
  - Reduce off screen buffer size(shown below)
- If Using PRT for self shadow:
  - Reduce the coefficients in use

[Image courtesy of GDC, Martin Reddy, Pixar Animation Studios, originally from Openworlds]

Impostors & Dynamic Textures

Texture Based L.O.D.

Techniques:

- Use Render to texture (available on most GPUs)
- Billboards
- Impostors

[Images courtesy of the U.S. Navy, available here]

Demo Time!!!

Applications of L.O.D.

"Real World*" Applications of L.O.D. in Real-time Rendering

Terrain Rendering

- *ROAM* Algorithms (variants not discussed)
- Recent Developments
  - *Planet-Sized Batched Dynamic Adaptive Meshes (P-BDAM)*
  - *Geometry clipmaps: terrain rendering using nested regular grids*

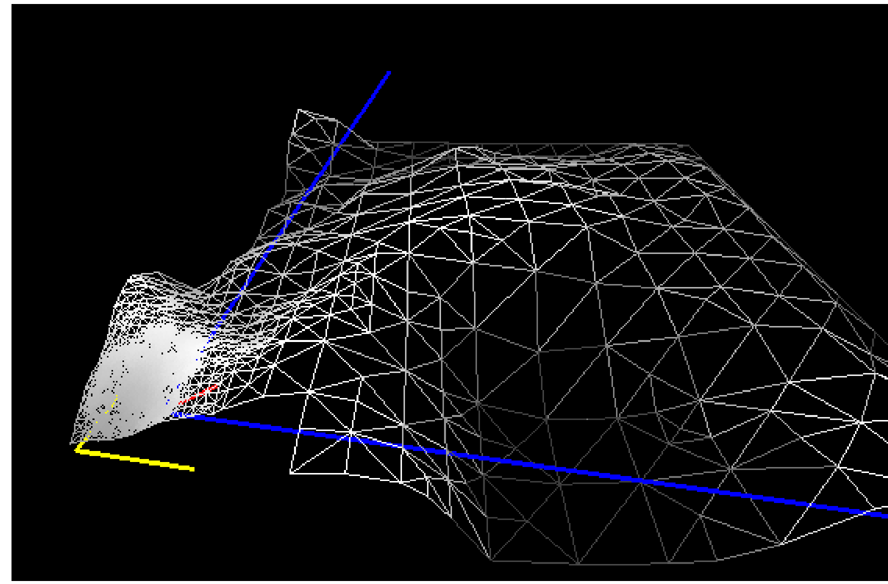
* "Real World" is an oxymoron because we all know we are rendering Virtual Worlds! ;-)

Terrain LOD: ROAM Algorithm

Mark Duchaineau, 1997 (LLNL)

<u>Goals:</u>

- Interactive Frame rates
- High Accuracy Terrain
- Frame Coherence
- Specified on # of triangles to render



Generation	Split/Merge on diamonds
Selection Factor	Screen space error metric
Switching Algorithm	Split/Merge & Geomorphing (if needed)

[image generated from ROAMSimple Demo, courtesy of Bryan Turner, Gamasutra.com]

Terrain LOD: ROAM Algorithm

Main Idea: ROAM (**R**eal-time **O**ptimally **A**dapted **M**esh) is an incremental priority-based using a binary triangle tree structure.

Features:

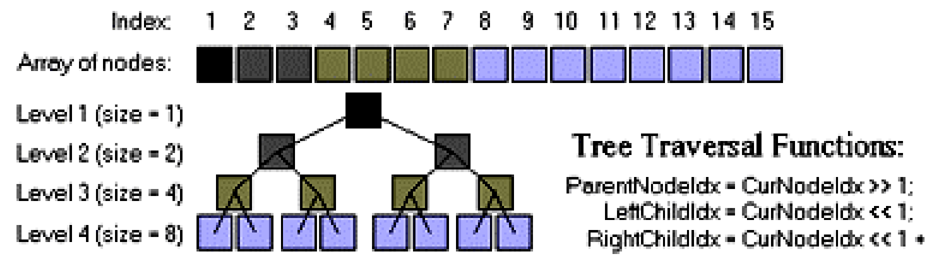
- Dual Priority Queues (Split/Merge)
- View Dependent mesh
- Continuous Mesh from height field
- Basic Unit: triangle pairs that share hypotenuses (Diamonds)
- Frame Coherence

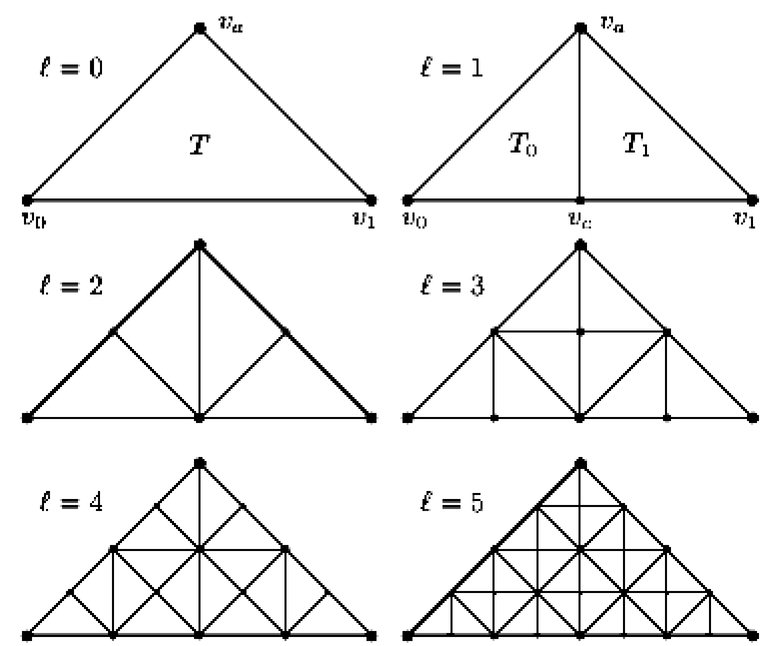
[image courtesy of Triston O'Connors demos]

ROAM: Binary Tree

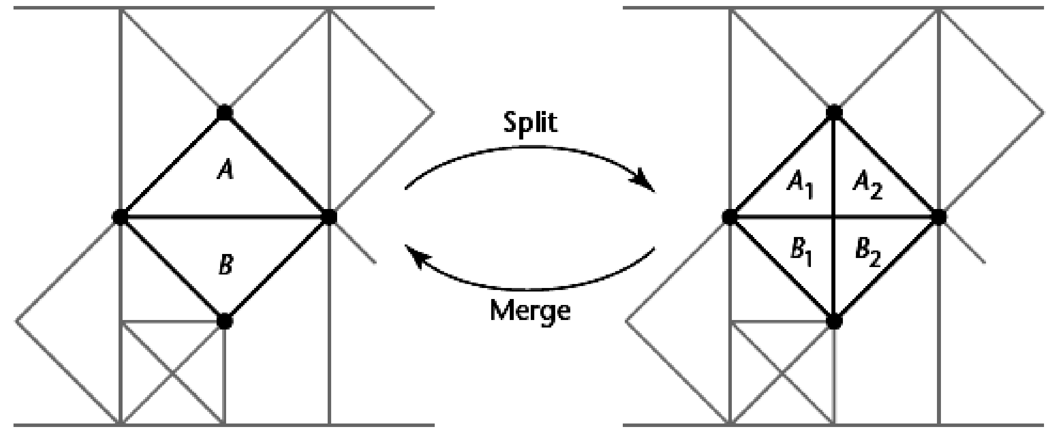
<u>Technique:</u>

- No Vertex data stored
- Implicit position of tris.
- Neighboring Triangles differ by no more than 1 level
- No cracks (built in to the algorithm)



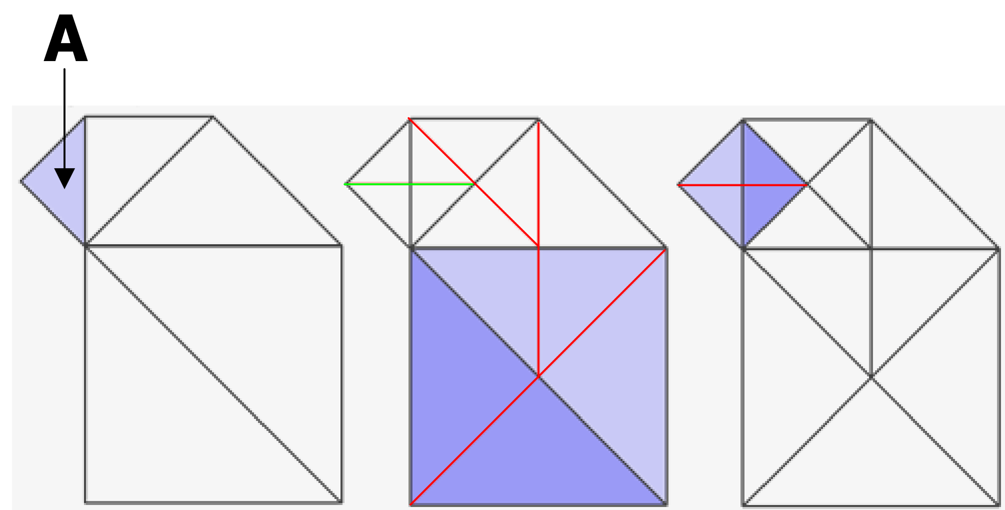


ROAM: Split & Merge



Forced Recursive Split:

1. Triangle **A** needs split.
2. No base neighbor, so split neighbor too.
3. Repeat until base neighbor found.
4. Then split original.



1) original 2,3) Force recursive split 3) Now split original

ROAM: 2 Queues

Split & Merge Priority Queues

<u>Technique:</u>

- Triangle priority determined by error metric
- **Split queue** will repeatedly split same triangle until the desired resolution is reached
- **Merge queue** inherits the data from previous frame exploits the coherence

ROAM: Split Algorithm

```
Let T = the base triangulation
For all t ∈ T, insert t into Q_split
While T is too small or inaccurate
{
        1. Identify highest-priority t in Q_split.
        2. Force-split t.
        3. Update Q_split as follows:
        {
            1. Remove t & other split triangles from Q_slit.
            2. Add any new triangles in T to Q_split.
        }
}
```

[Psuedocode adapted from **ROAMing Terrain: Real-time Optimally Adapting Meshes**, Mark Duchaineau, 1997]

ROAM: Merge

In Addition To Split, add the following logic:

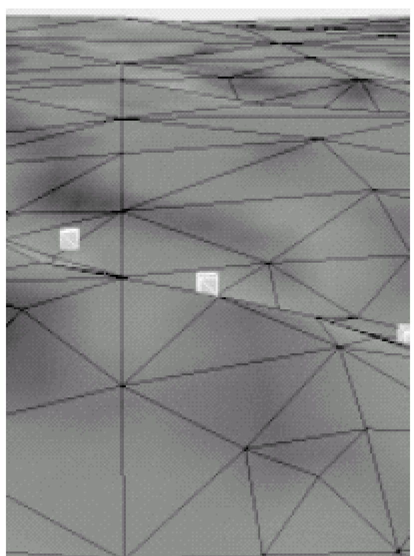
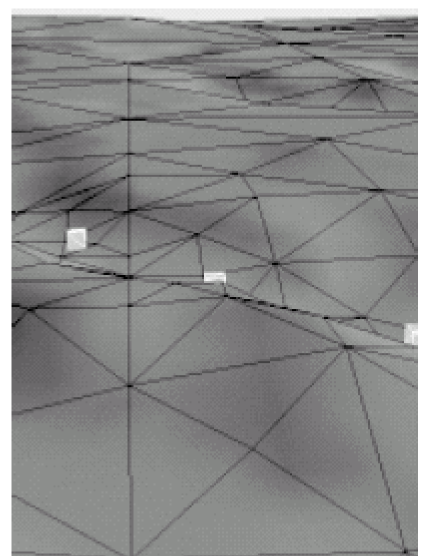
Continue processing $T = T_{previous}$
if T is too large or accurate
{
Identify lowest-priority (T; T_{Back}) in Q_{merge}.
Merge (T; T_{Back}).
Update queues as follows:
{
1. Remove all merged children from Q_{split}.
2. Add merge parents T,T_{Back} to Q_{split}.
3. Remove (T; T_{Back}) from Q_{merge}.
4. Add all newly-mergeable diamonds to Q_{merge}.
}
}

[Psuedocode adapted from **ROAMing Terrain: Real-time Optimally Adapting Meshes**, Mark Duchaineau, 1997. See paper for more details see pg. 5]

ROAM: Error Metrics

Principal Error Metrics (AKA Priority)

- Nested world-space bounds (see Wedgies, next)
- Geometric Screen Distortion
  - $\text{distortion}(v) = ||S(v) - S_T(v)||_2$
  - $\text{distortion}_{max} = \max_{v \in V} \text{distortion}(v)$
- Line of sight

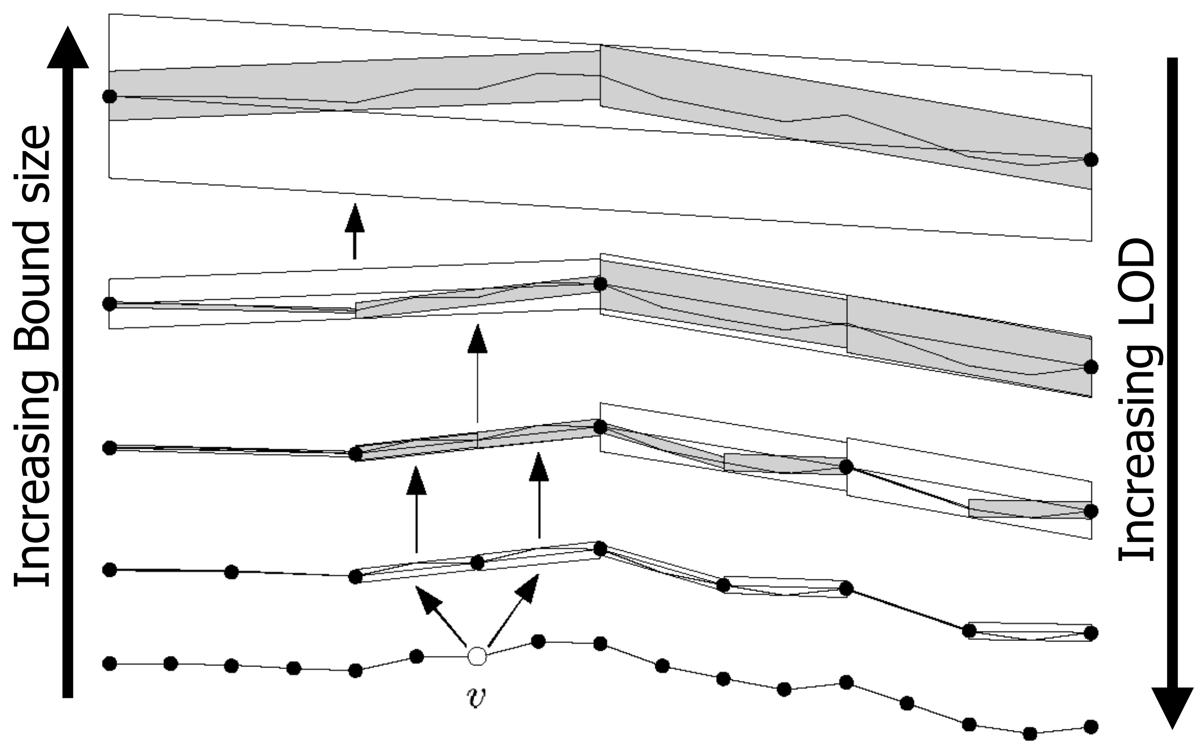



[example LOS improvisation, image from Mark Duchaineau, 1997]

ROAM: Wedgie!

Wedgie: Bounding Volume

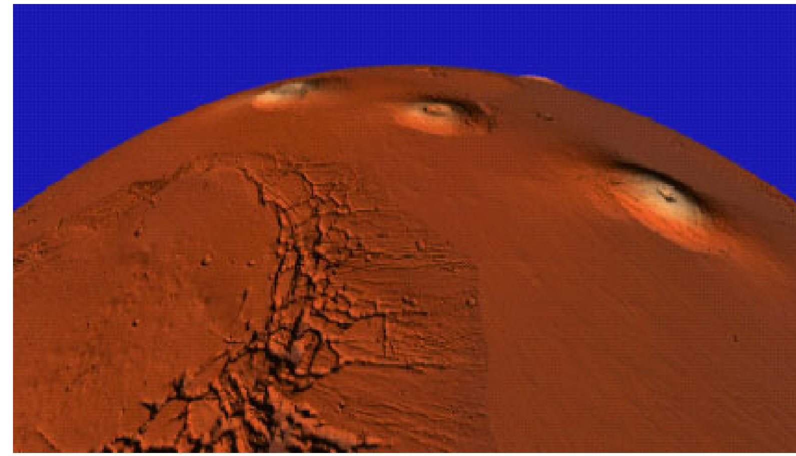
- Covers the *(x,y)* extent of a triangle and extends over the height range $z\text{-}e_T$ through $z\text{+}e_T$



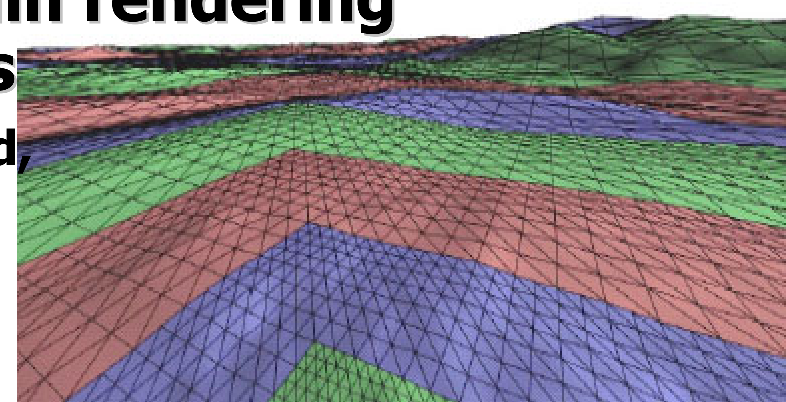
[image Adapted from Mark Duchaineau, 1997]

Recent Developments

- Planet-Sized Batched Dynamic Adaptive Meshes (P-BDAM)
  - Paolo Cignoni, et al. IEEE Viz 2003
  - a batched host-to-graphics communication model which outperforms other adaptive tessellation (compressed out-of-core)
  - exploits programmable graphics hardware



- Geometry Clipmaps: terrain rendering using nested regular grids
  - F. Losasso, H. Hoppe (Stanford, Microsoft), Siggraph 2004
  - Simplifies purely on distance
  - Uses GPU to do smoothing



[images from conference papers, see references]

Demo Time!!!

References

- ***Real-Time Dynamic Level of Detail Terrain Rendering with ROAM***, *Bryan Turner*, Gamasutra.com, April, 2000
- ***Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids***, Losasso, F. and Hoppe, H. (2004). In Proceedings of SIGGRAPH 2004, Los Angeles, CA.
- ***Planet-Sized Batched Dynamic Adaptive Meshes (P-BDAM)***, Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. In *Proceedings IEEE Visualization*. Pages 147-155. IEEE Computer Society Press, October 2003.
- ***ROAMing Terrain: Real-time Optimally Adapting Meshes***, Duchaineau, M., M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, M. B. Mineev-Weinstein. (1997). Visualization '97. pp. 81-88
- ***Level of Detail for 3D Graphics***, D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner, Morgan Kaufmann, 2003
- ***Real-Time Rendering 2nd Edition***, Moller, Tomas, Haines, Eric, AK Peters, 2002
- ***Run-Time Management of LOD***, Martin Reddy, GDC 2003 Course Materials, slides

References

- ***Generating Levels of Detail***, Amitabh Varshney , GDC 2003 Course Materials, slides
- ***Terrain LOD***, Martin Reddy, GDC 2003 Course Materials, slides
- ***Level Of Detail for Games***, Robert Huebner, GDC 2003 Course Materials, slides
- ***Concepts and Algorithms for Polygonal Simplification,*** Jonathan D. Cohen, SIGGRAPH 99 Course Tutorial #20: Interactive Walkthroughs of Large Geometric Datasets. pp. C1-C34. 1999.