Johns Hopkins
Engineering for Professionals

605.767 Applied Computer Graphics

Brian Russin

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Module 5F
Level of Detail

Levels of Detail

www.viewpoint.com

Helicopter shown here comes in four resolutions: 998, 9828, 115817, and 255,241 polygons

Very High. The ultimate in 3D models. For use in cinema and broadcast where maximum realism is required.

High. High level of detail models for use in foreground shots.

Medium. Perfect for models that occupy less prominent positions in scenes, keeping rendering time to a minimum.

Low. Low polygon count models for real time applications, simulators, and game developers.

Levels of Detail

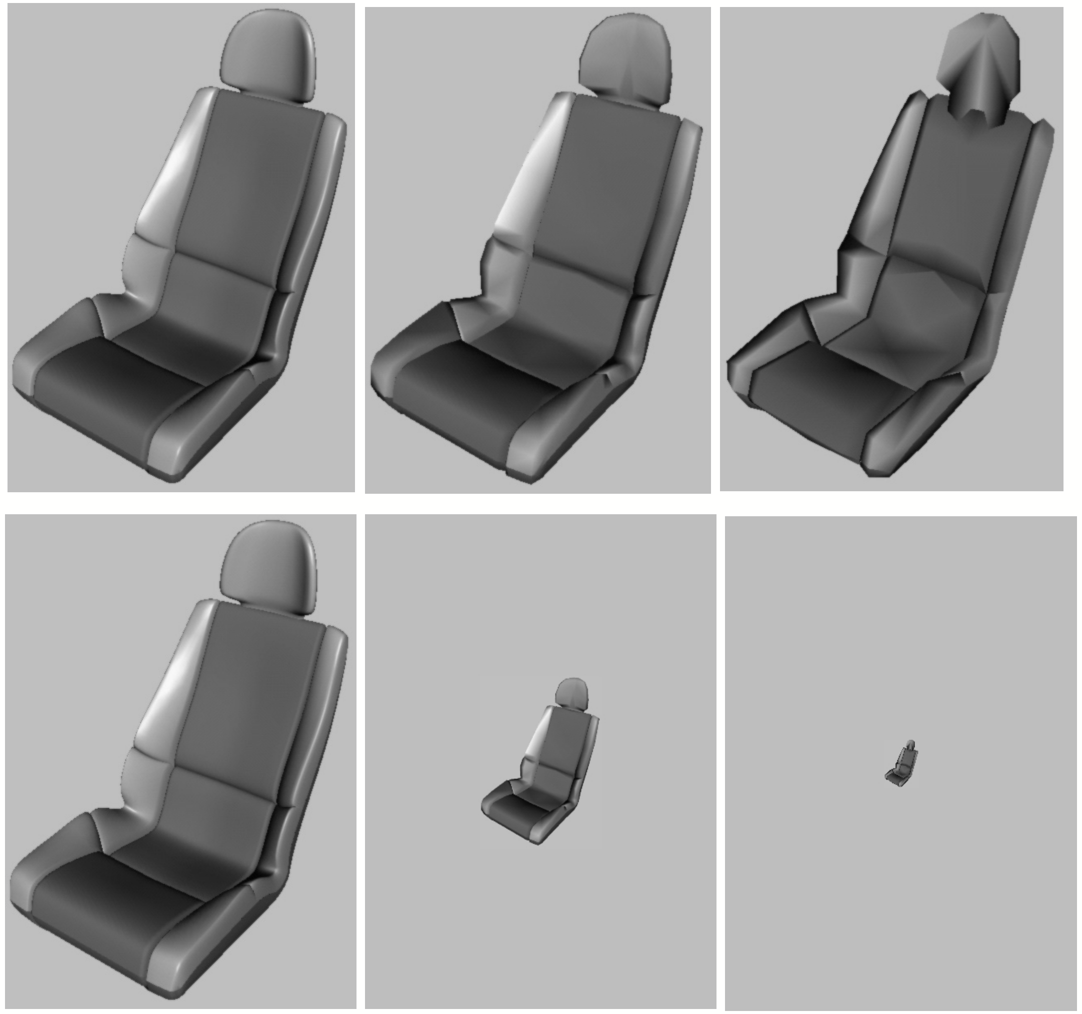
- Levels of Detail (LOD)
  - Use simpler versions of an object based on distance or projected size of object
    - Complex object may have 1000s of triangles
      - Provides detail when close to the object
    - When object is far away many triangles project to the same pixels
      - A far less detailed representation will appear nearly the same
      - Significant speedup
- Many objects have levels of detail as part of the description
  - Spheres, Bezier surfaces, subdivision surfaces
    - LOD describes the how curved surface is tessellated into polygons
- LOD algorithms consist of 3 primary parts:
  - Generalization – simplifying object geometry
  - Selection – choosing the level of detail
  - Switching – changing from one level to another

LOD Switching

- Abrupt changes in appearance can result when switching from one LOD to another
  - Can be distracting
  - Effect is known as **popping**
- Several different LOD switching methods are available
  - Discrete geometry LODs
    - Multiple versions of same model
    - Popping is typically the worst for this method
      - Select one discrete LOD on one frame, another on the next
  - Blend LODs
    - Linear blend between 2 LODs over a short period of time
      - Must draw both LODs
  - Alpha LODs
    - Fade the object as it gets beyond a certain range, until gone
  - Geomorph LODs
    - Smoothly transition between two different LODs

Discrete LODs

- Object representations model the same object with different numbers of primitives
  - Each LOD modeled separately
- Selection is made on a frame by frame basis
  - One LOD can be selected in one frame and another in the next
  - Popping can be noticeable unless a large number of LODs with small differences are provided



https://graphics.stanford.edu/courses/cs248-05/real-time-programming/moller-cs248-01-lecture.pdf

Blend LODs

- Perform a linear blend between 2 LODs over a period of time
  - While animation / view change is occurring
  - Render both LODs
    - More expensive – somewhat defeats the purpose of LODs
    - Often LOD switching is occurring for a small number of objects in the scene
  - Care must be taken to draw such that blending does not lead to artifacts
    - Blending depends on contents of the frame buffer
- Giegl and Wimmer proposed a method that works well
  - During transition from LOD1 (current LOD) to LOD2
    - Draw LOD1 opaquely
    - Fade in LOD2 by increasing its alpha from 0 to 1 over a number of frames
      - After which LOD2 becomes the current LOD
    - LOD2 should be rendered with depth buffer read-only
      - OpenGL: use glDepthMask(GL_FALSE)

Alpha LODs

- Fade out the object as it gets beyond a certain range
  - Simple method that avoids popping
- Uses only 1 detail level / instance
  - As LOD selection criteria increases the transparency of the object is increased
    - Until the object disappears
  - Two thresholds
    - Level when transparency starts
    - Level when object reaches full transparency – invisibility threshold
- Advantage
  - Avoids popping
  - Speedup occurs when object disappears
- Disadvantage
  - No speedup occurs until object disappears

Continuous Levels of Detail (CLOD)

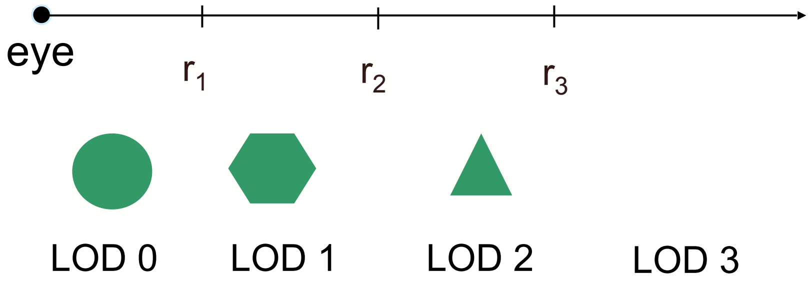
- Mesh simplification processes can be used to create LODs from a single object
  - **Edge collapse** methods allow other ways of transitioning between LODs
    - Edge collapse – a vertex is joined with another
    - Creates model with 2 fewer polygons
- CLODs (Continuous Levels of Detail)
  - Large set of models
  - Each level has 1 edge collapse
  - Keep track of edge collapses to reverse (vertex split)
  - Can animate edge collapse for smooth transition
  - Problems:
    - Not all models look good
    - Have to go through all prior edge collapses to get to a particular LOD

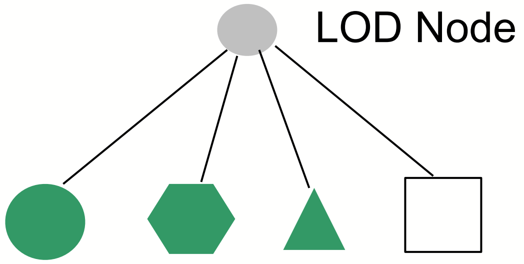
Geomorph LODs

- Geomorph LODs smoothly transition between two different LODs
  - Set of discrete models created by simplification
    - Associate vertices on the more complex model with vertices on the simpler model
  - When switching from complex model to simpler model interpolate between vertex positions between the 2 models
  - Advantages
    - Can select discrete models of high quality
    - Avoids popping
  - Disadvantages
    - Cost of interpolating between vertices
    - Objects always appear to be changing

LOD Selection

- LOD selection chooses which LOD to use
  - Or which to blend
- Metric based on current view position and object location
  - 2 basic metrics: range-based and projected area-based
- Range-based metric
  - Associate different LODs with different ranges
    - LOD 0 (most detailed) from range 0-r1
    - LOD 1 from range r1-r2
      - And so on…

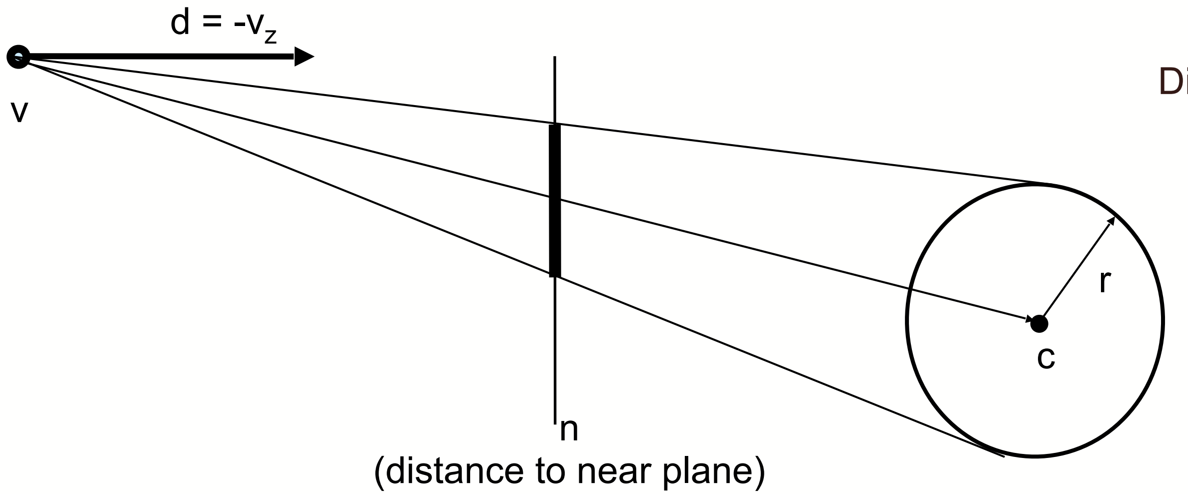




One child of the LOD node is selected based on range r
Last LOD node is empty object (nothing is drawn beyond r3)

Projection Area-Based Metric

- Screen space coverage
  - Which LOD to use is related to the number of pixels it covers
- Estimate the size of the object's BV
- For perspective projection - size of an object diminishes as distance from the viewer increases along the view direction
  - Size of projection is halved as distance is doubled
- Estimating projected area of a bounding sphere:



Distance along d to sphere center

$$d \cdot (c - v)$$

Estimate of projected radius:

$$p = \frac{nr}{d \cdot (c - v)}$$

$$area = \pi p^2$$

Estimating Area of a Bounding Box

- Schmalstieg and Tobler
  - Classify viewpoint with respect to the box
    - Use this to determine which projected vertices are in the silhouette of the projected box
    - Done via a Lookup Table (LUT)
  - Determine how many (1, 2, or 3) and which faces of the box are visible
    - See Figure 19.38 (14.28 in 3rd Edition)
  - Project the vertices of the silhouette edge to the screen
    - 4 or 6 vertices
  - Compute the area from these points
- LUT implementation
  - For each pair of planes defining the box along one axis, classify the eye location as above, below, or between these planes
    - 3*3*3 = 27 possibilities
    - LUT defines how many vertices there are in the silhouette
  - A second LUT is used to find the actual vertices of the silhouette
    - Listing the 4 or 6 silhouette edge vertices
- www.cg.tuwien.ac.at/research/publications/1999/Fuhr-1999-Conc/TR-186-2-99-05Paper.pdf

Culling Summary

- For most scenes: rapidly compute a potentially visible set using BVs and view frustum culling
  - Use LODs for complex objects if need to maintain high frame rate
  - Let hardware handle the rest
    - Use OpenGL backface culling
      - Unless application has a bottleneck in the geometry stage
- For scenes with high depth complexity may need to perform some form of occlusion culling
  - First try using OpenGL occlusion query extensions
- For architectural models use cells and portals