Image courtesy of Epic Games

Dieter Schmalstieg

Levels of Detail

Based on material from Michael Wimmer and Markus Grabner

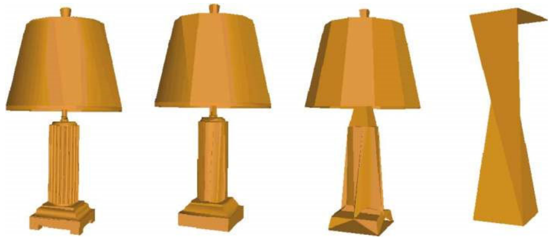
Case Study: Unreal Engine 5

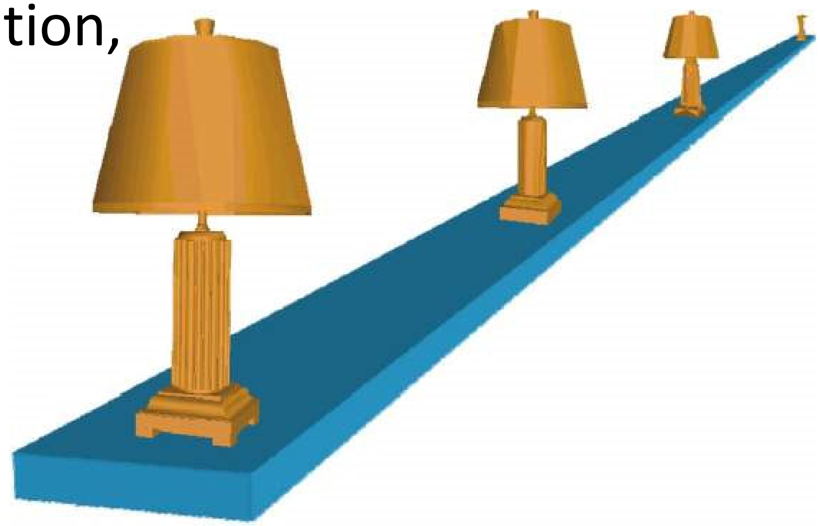
- Siggraph Tutorial https://www.youtube.com/watch?v=TMorJX3Nj6U
- Reveal Trailer https://www.youtube.com/watch?v=qC5KtatMcUw&ab_channel=UnrealEngine
- Leading game engine made by Epic
  - Open source :-)
  - High code complexity :-(
- Supports for huge models → our topics today
  - Level of detail
  - Visibility culling
  - Virtual textures and geometry

LOD - Basic Idea

- Problem: even after visibility, model may contain too many polygons
- Idea: Simplify the amount of detail used to render small or distant objects
- Known as levels of detail (LOD)

  A.k.a. multiresolution modeling, polygonal/geometric simplification, mesh reduction/decimation, …

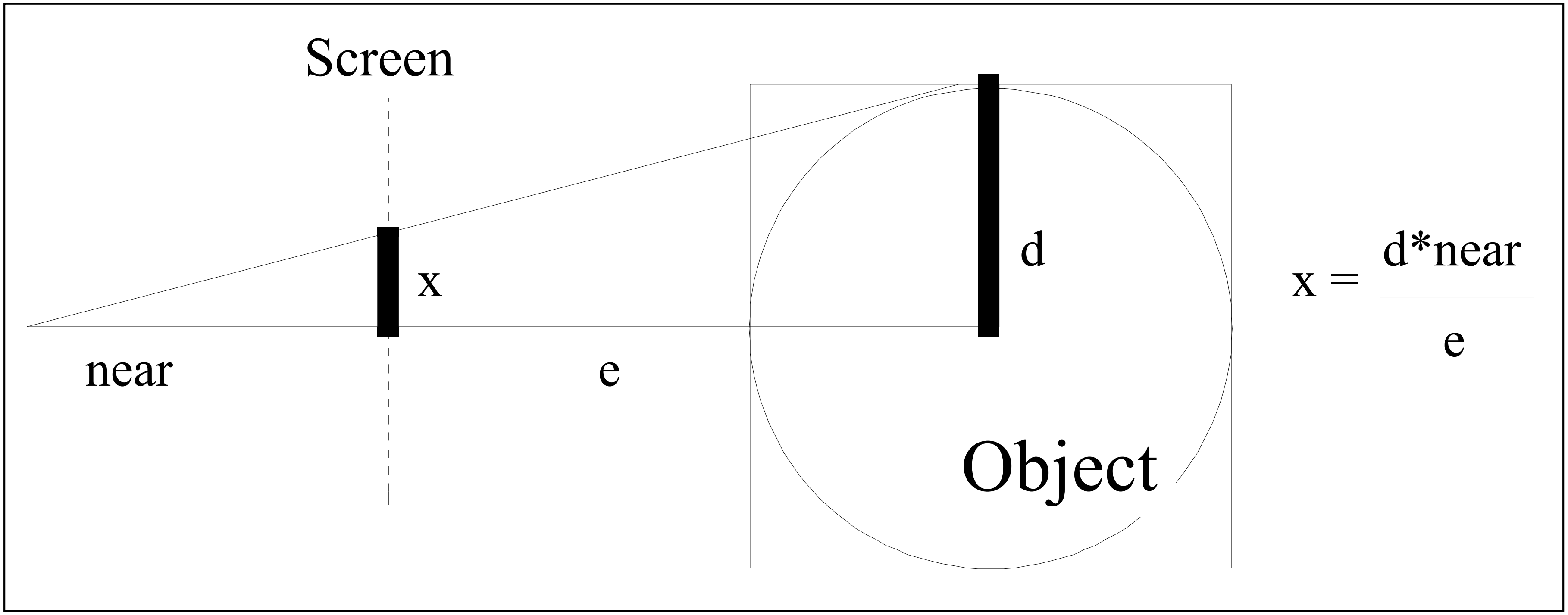




https://www.youtube.com/watch?v=mIkIMgEVnX0

Static LOD Selection

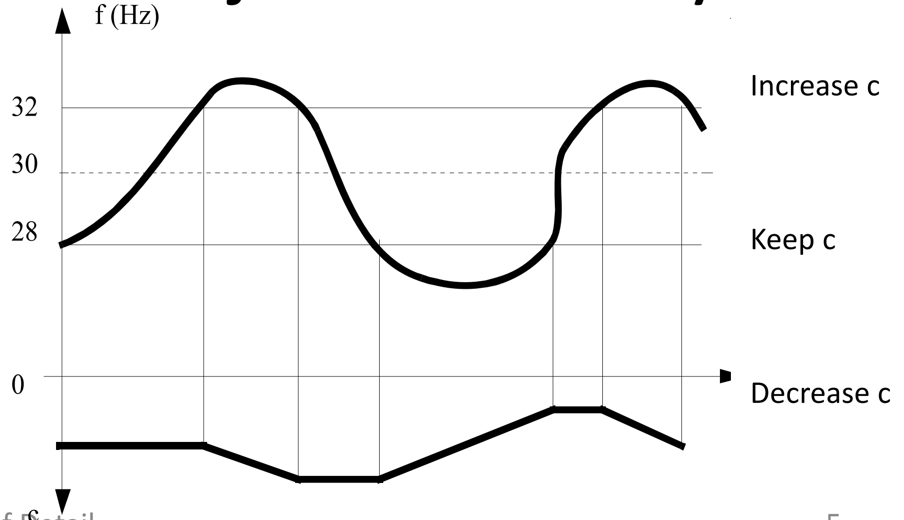
- LOD Selection steered by size of object in image
- Cannot control resulting frame rate



$$x = \frac{d*near}{e}$$

Reactive LOD Selection

- Multiply object size with factor *c*
- If frame rate too low → decrease *c*
- If frame rate too high → increase *c*
- Results in roughly constant frame rate
- Problems occur, if complex objects suddenly become visible
- Requires hysteresis



Predictive LOD Selection

[Funkhouser & Sequin 1993]

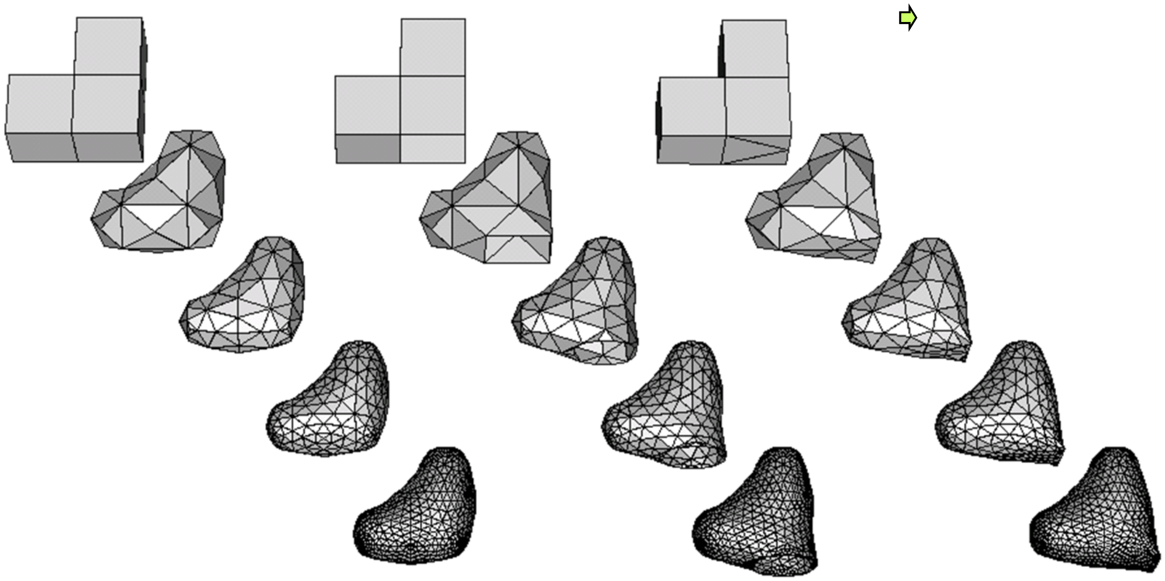
- COST = time for drawing object with a given LOD
- Goal: best possible image quality
- BENEFIT = contribution of object to image quality
  - Most important: screen-size of object
- Optimization (*Rucksack*) problem
  - Sum(BENEFITS) → max, BUT
  - Sum(COSTS) ≤ FRAMETIME

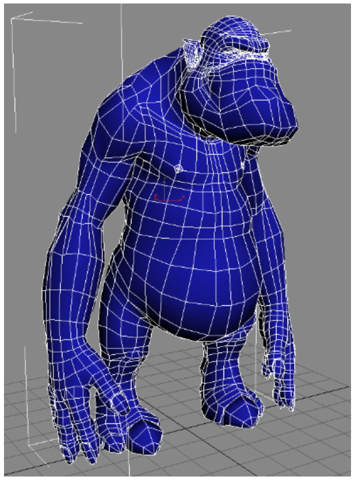
LOD Switching

- Hard Switching
  - + Simple
  - - “Popping” artefacts
- Blending
  - + For all types of LOD
  - - Temporarily increased rendering load
  - - Problems with transparences, shadows, etc…
- Geomorphing https://www.youtube.com/watch?v=I20Zyr4U_Xk
  - → Interpolate triangles shapes from 1st to 2nd LOD
  - + Best quality
  - - Requires geometric correspondence between LODs

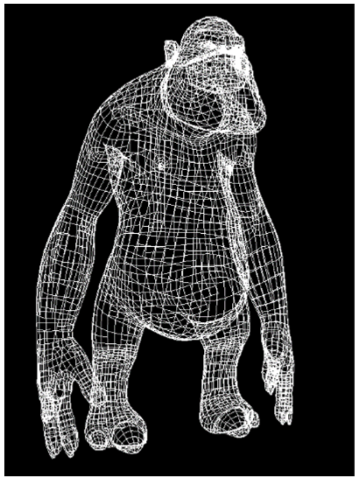
LOD by Subdivision Surfaces

- Curved surface defined by repeated subdivision steps on a polygonal model
- Subdivision rules create new vertices, edges, faces based on neighboring features
- Compute in geometry shader





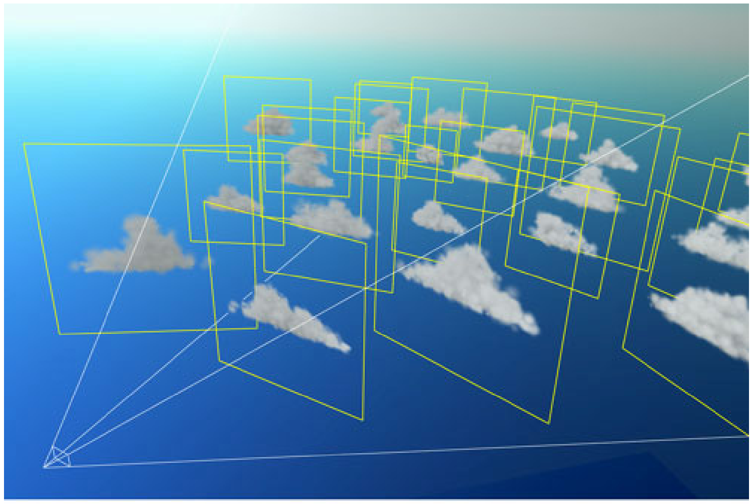
4K triangles



17K triangles

LOD by Shading and Rendering

- Shading and illumination
  - From simple local to complex global illumination
- Geometry vs. images
  - Textures, impostors

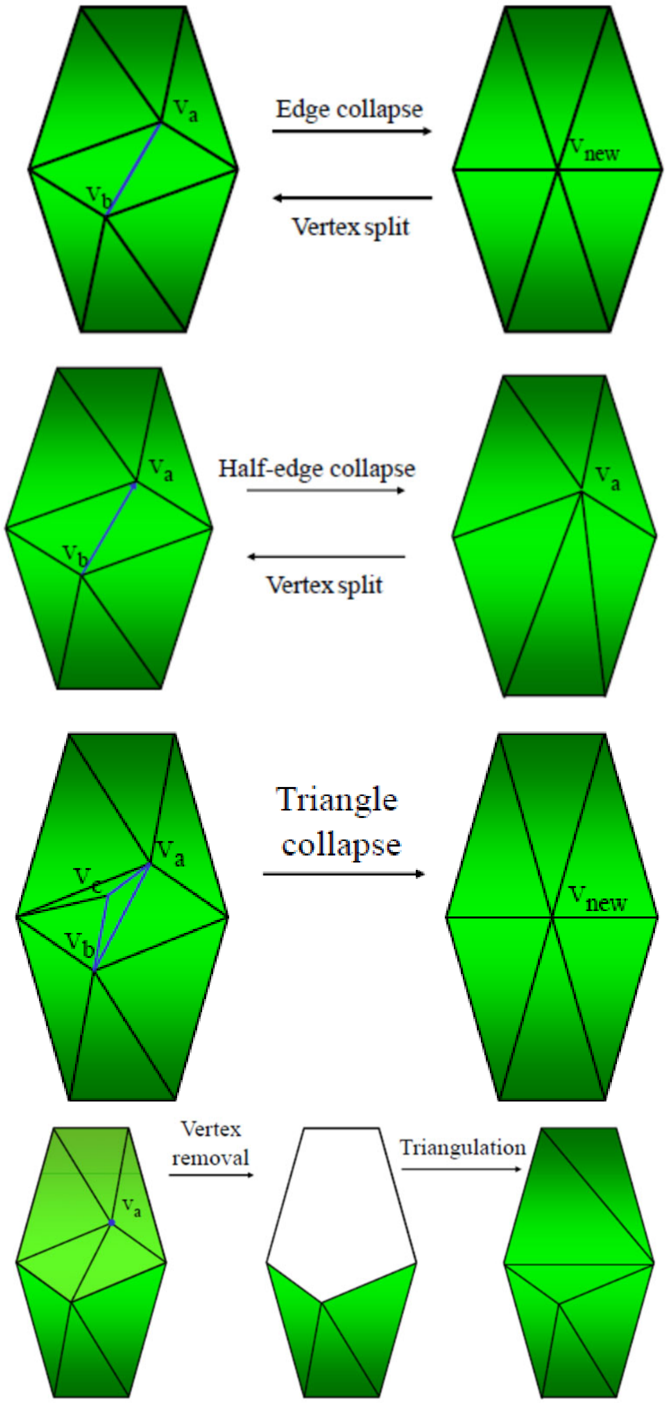


LOD by Geometric Simplification

- Iteratively reduce number of primitives
  - Vertices, edges, triangles
- Topology simplification
  - Reducing number of holes, tunnels, cavities
- Does not change rasterization
  - Fragment shader load remains roughly identical

Local Simplification

- Edge collapse
- Vertex-pair collapse
- Triangle collapse
- Cell collapse
- Vertex removal



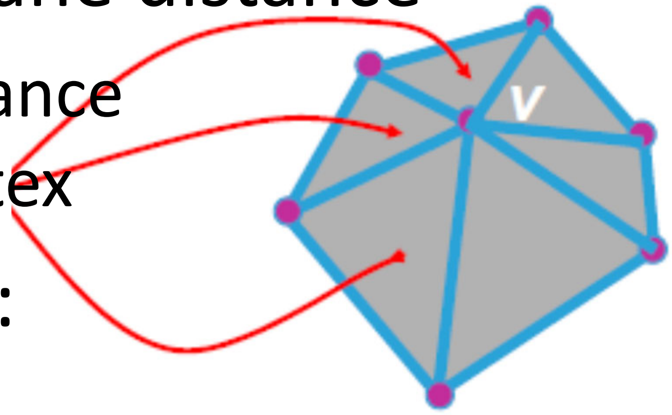
Lazy Greedy Local Simplification

- Fewer cost evaluations

```
compute costs for each possible operation
insert them into queue
set „dirty“ flags to false
while the queue is not empty
    extract head of queue (i.e., element with smallest error)
    if head is dirty
        re-compute cost
        set „dirty“ flag to false
        re-insert into queue
    else
        perform operation
        for each neighbor
            set „dirty“ flag to true
```

Quadric Error Metric

- Error measure = vertex-to-plane distance
  - Minimize sum of squared distance to all planes attached at a vertex



- Plane equation for each face:

  p: $Ax + By + Cz + D = 0$

- Distance to vertex *v for* a single plane:

  $\Delta v = p^{\mathrm{T}} \bullet v = [A\ B\ C\ D] \bullet [x\ y\ z\ 1]^T$

Using the Quadric Error Metric

$$\Delta(v) = \sum_{p\in planes(v)} (p^T v)^2$$

$$= \sum_{p\in planes(v)} (v^T p)(p^T v)$$

$$= \sum_{p\in planes(v)} v^T (pp^T) v$$

$$= v^T \left(\sum_{p\in planes(v)} pp^T \right) v$$

$$\Delta(v) = v^T (Q) v$$

Q

- pp^T = plane equation squared:

$$pp^T = \begin{bmatrix} A^2 & AB & AC & AD \\ AB & B^2 & BC & BD \\ AC & BC & C^2 & CD \\ AD & BD & CD & D^2 \end{bmatrix}$$

- $\Sigma\ pp^T$=**Q** is also a matrix
- Sandwich product Δ*v* computes distance metric *v* → all planes
- For edge collapse of vertices v_1+v_2, just add Q_1+Q_2