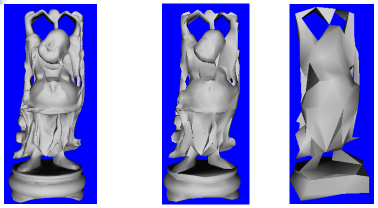
## Mesh Simplification



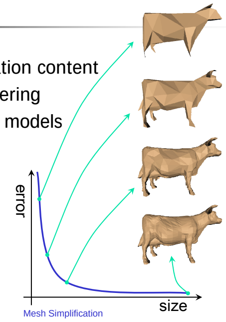
12,000 2,000 300

Simplifier



## Motivation

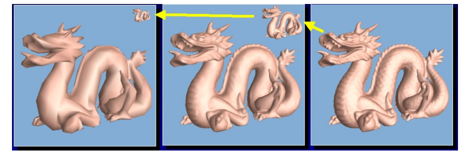
- Reduce information content
- Accelerate rendering
- Multi-resolution models






## Level of Detail (LOD)

- Refined mesh for close objects
- Simplified mesh for far





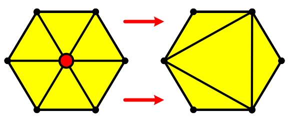
## Performance Requirements

- Real-time
  - Generate model at given level(s) of detail
  - Focus on speed
  - Requires preprocessing
  - Time/space/quality tradeoff



## Methodology

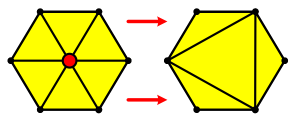
- Sequence of local operations
  - Involve near neighbors - only small *patch* affected in each operation
  - Each operation introduces error
  - Find and apply operation which introduces the least error





## Simplification Operations (1)

- Decimation
  - Vertex removal:
    - v ← v-1
    - f ← f-2

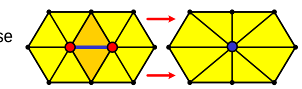


- Remaining vertices - subset of original vertex set



## Simplification Operations (2)

- Decimation
  - Edge collapse
    - v ← v-1
    - f ← f-2

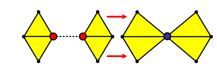


- Vertices may move



## Simplification Operations (3)

- Contraction
  - Pair contraction



- Vertices may move

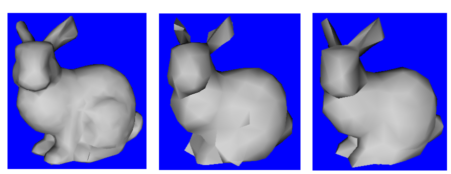


## Error Control

- Local error: Compare new patch with previous iteration
  - Fast
  - Accumulates error
  - Memory-less
- Global error: Compare new patch with original mesh
  - Slow
  - Better quality control
  - Can be used as termination condition
  - Must remember the original mesh throughout the algorithm



## Local vs. Global Error

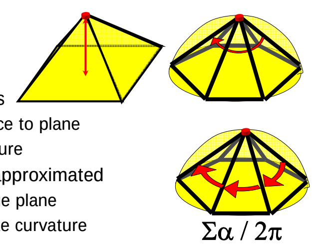


2000 faces 488 faces 488 faces



## Simplification Error Metrics

- Measures
  - Distance to plane
  - Curvature
- Usually approximated
  - Average plane
  - Discrete curvature






## The Basic Algorithm

- Repeat
  - Select the element with minimal error
  - Perform simplification operation (remove/contract)
  - Update error (local/global)
- Until mesh size / quality is achieved



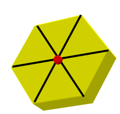
## Implementation Details

- Vertices/Edges/Faces data structure
  - Easy access from each element to neighboring elements
- Use priority queue (e.g. heap)
  - Fast access to element with minimal error
  - Fast update

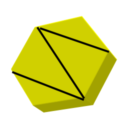


## Vertex Removal Algorithm

- Simplification operation: Vertex removal



- Error metric: Distance to average plane
- May preserve mesh features (creases)





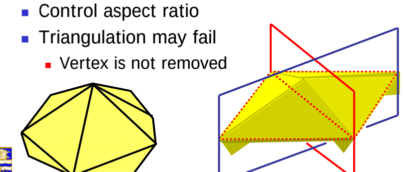
## Algorithm Outline

- Characterize local topology/geometry
- Classify vertices as removable or not
- ***Repeat***
  - Remove vertex
  - Triangulate resulting hole
  - Update error of affected vertices
- ***Until*** reduction goal is met

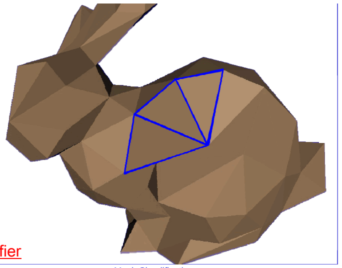


## Triangulating the Hole

- Vertex removal produces non-planar loop
  - Split loop recursively
  - Split plane orthogonal to the average plane
- Control aspect ratio
- Triangulation may fail
  - Vertex is not removed



## Example



Simplifier



## Pros and Cons

- Pros:
  - Efficient
  - Simple to implement and use
    - Few input parameters to control quality
  - Reasonable approximation
  - Works on very large meshes
  - Preserves topology
  - Vertices are a subset of the original mesh
- Cons:
  - Error is not bounded
    - Local error evaluation causes error to accumulate