All the Polygons You Can Eat

Doug Rogers

Developer Relations

doug@nvidia.com

Future of Games

- **Very high resolution models**
  - 20,000 triangles per model
  - Lots of them
- **Complex Lighting Equations**
  - Floating point
  - Usually rely on surface normal

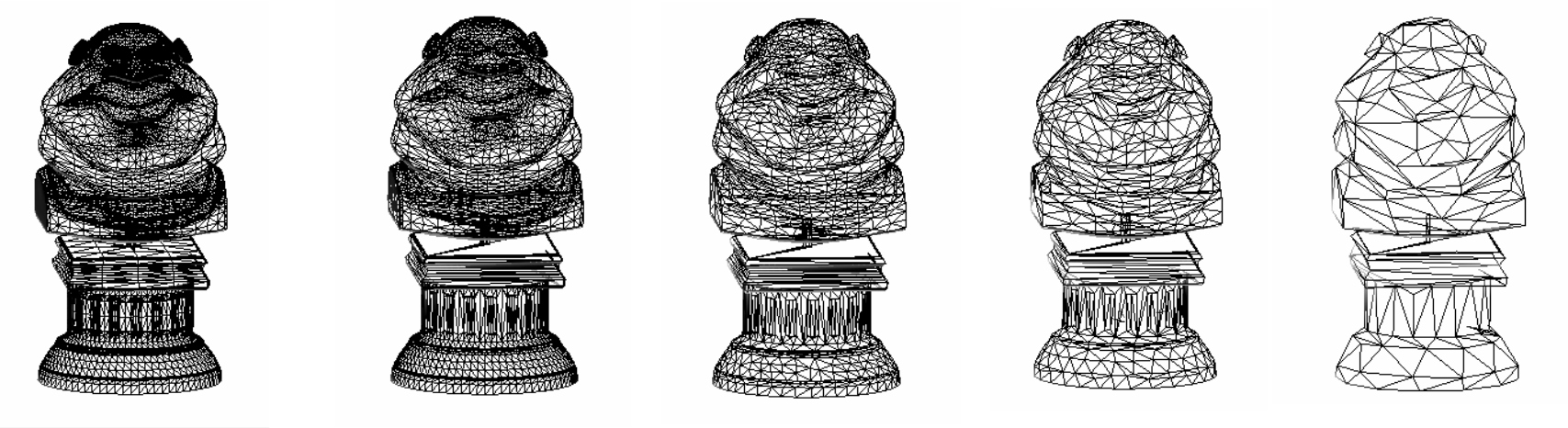
Managing Levels of Detail

- Model of 20,000 is good
- Model of 20,000 is that covers 20 pixels is not
- Reduce triangle count
- Retain as much quality as possible
- Melody (Multiple Level Of Detail Extraction)

Melody Features

- Normal Map Generation
- Texture Coordinate Generation
- Texture Coordinate Optimization
- Simplification
- Optimization
- Subdivision Surface
- Hull

Polygon Reduction



19k 9k 4k 2k 500

- Melody automatically creates these model

But just reducing the polygon count reduces the quality

We want this model:

500 faces – low res model

To be lit like the original

19k faces

Lighting a low resolution model

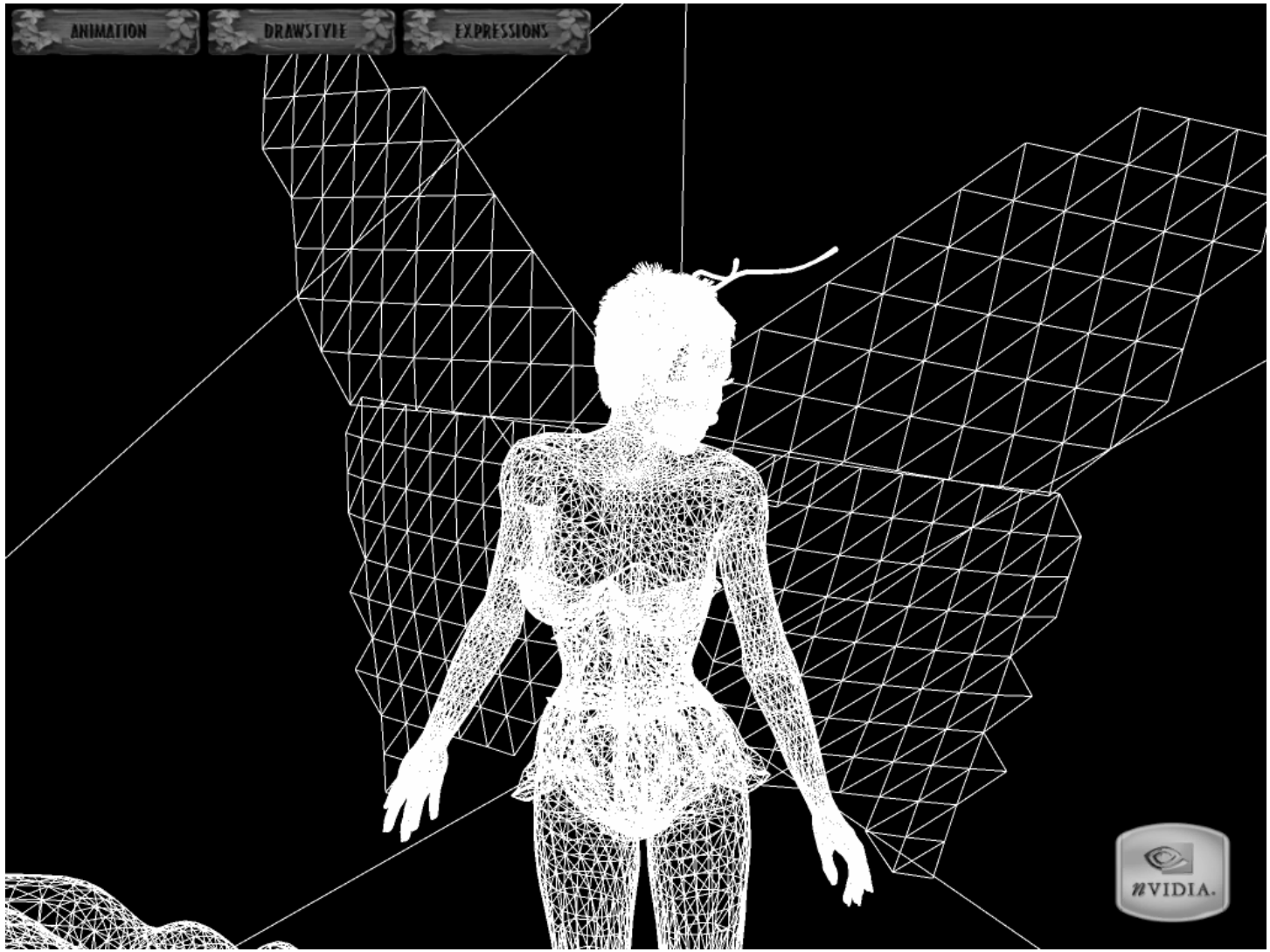
- Using lighting information from the hires model for the low res model

500 faces

19k faces

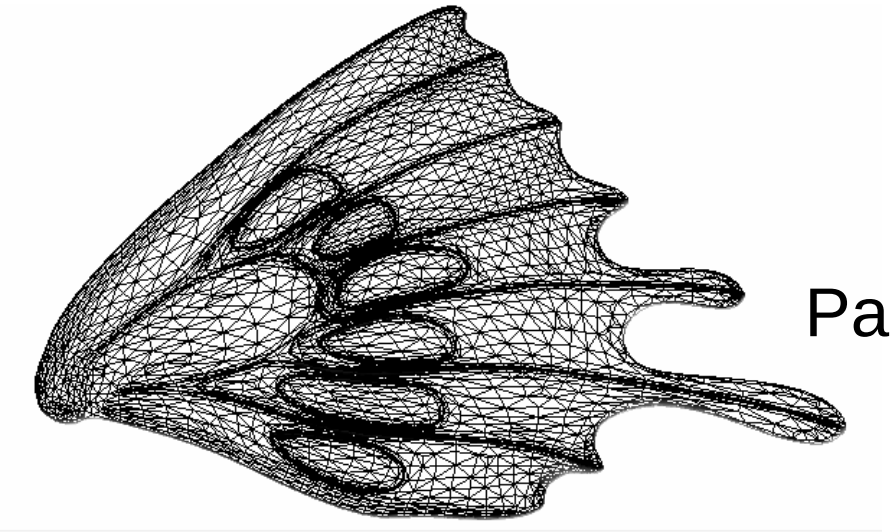
Real World Example

In Wire Frame



Terms: Reference Model

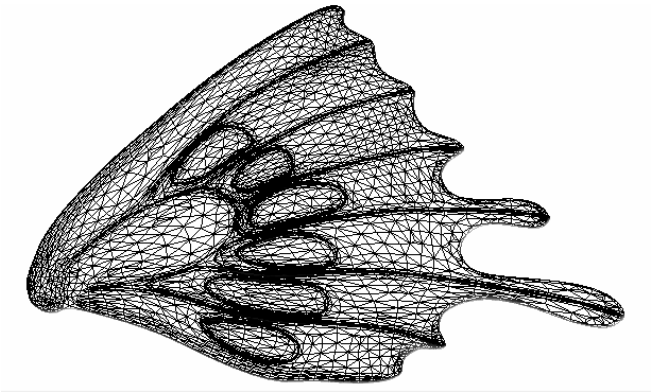
- **High resolution model used as a reference**
- **Used for surface attribute info**



Part of Dawn’s wing

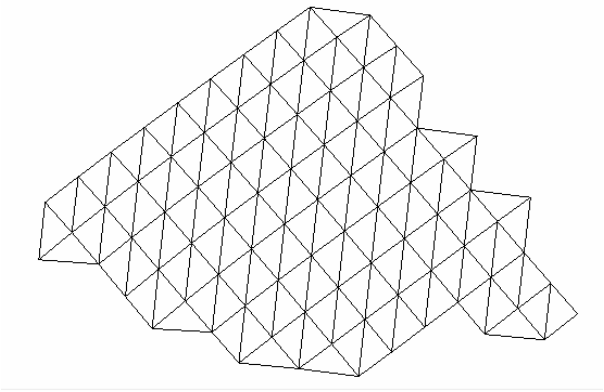
Terms: Working Model

- **Model that is to be simplified to create the lo-res model**
- **May be same as reference model or a hand simplified model**



Same as reference

or



Manually simplified

Lighting calculations are based on surface normals

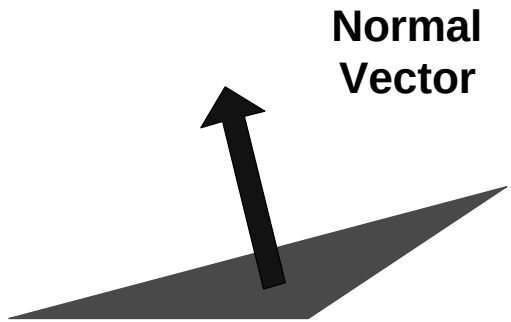
- Calculate the normals for the reference (hi-res) resolution model
- Store them into a texture (normal map)
- Use them on the working (low res) model

Calculating Surface Normals for hi-res mesh

- Calculate face normals
- Use face normals to calculates vertex normals
- Use vertex normals to calculate surface normal

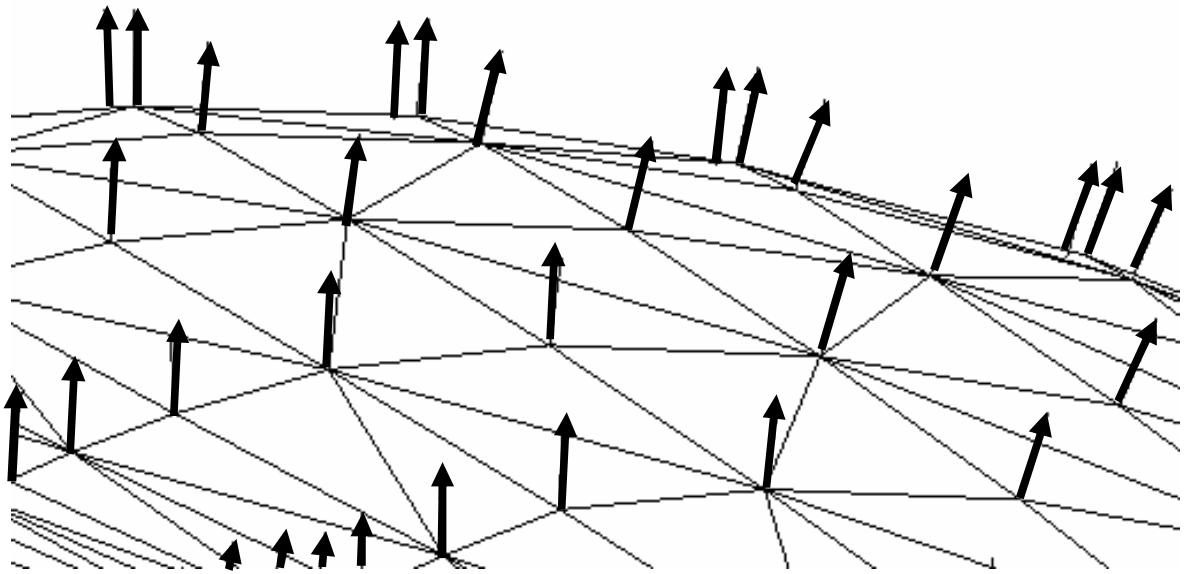
What is a Face Normal?

- A vector is normal to a surface when its direction is perpendicular to the plane which contains this surface
- When the magnitude of the vector is equal to 1 unit, the vector is called normalized
- The direction the triangle is facing, or the ‘up’ direction



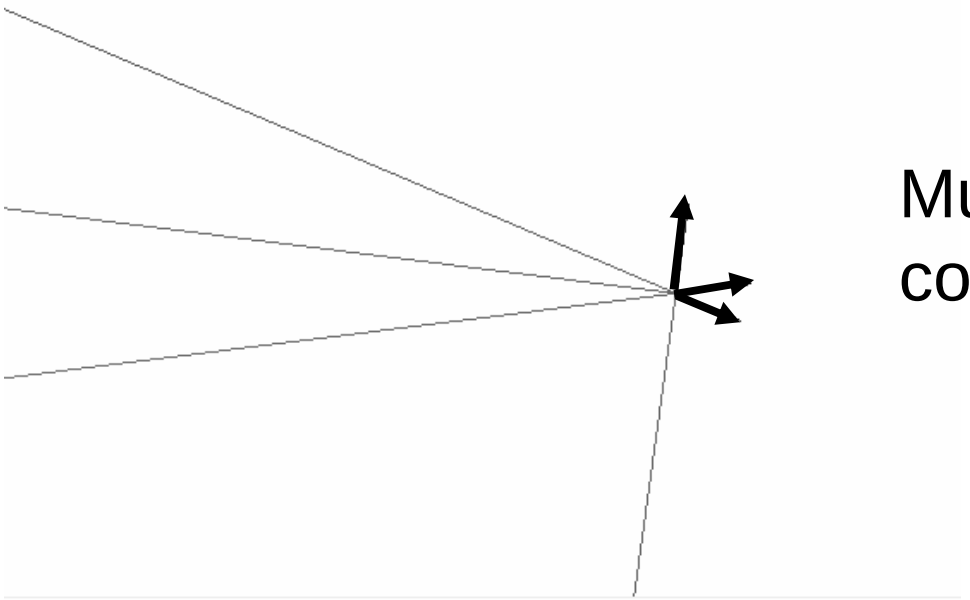
What is a Vertex Normal?

- **Summing all the face normal and normalizing the result yields *one* vertex normal**
- **This vector is the vertex normal and is used for vertex lighting**



Multiple Normals per Vertex

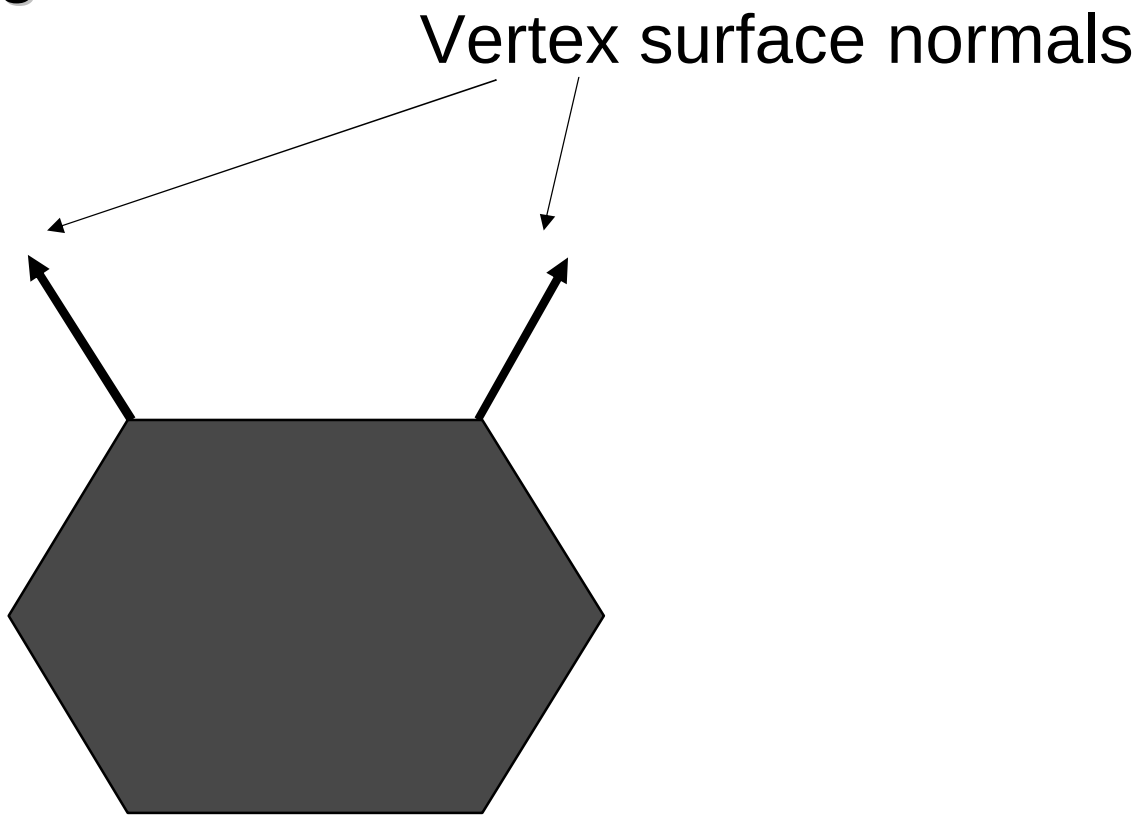
- **Sharp edges or borders of smoothing groups do not share normal**
  - Multiple normals per vertex



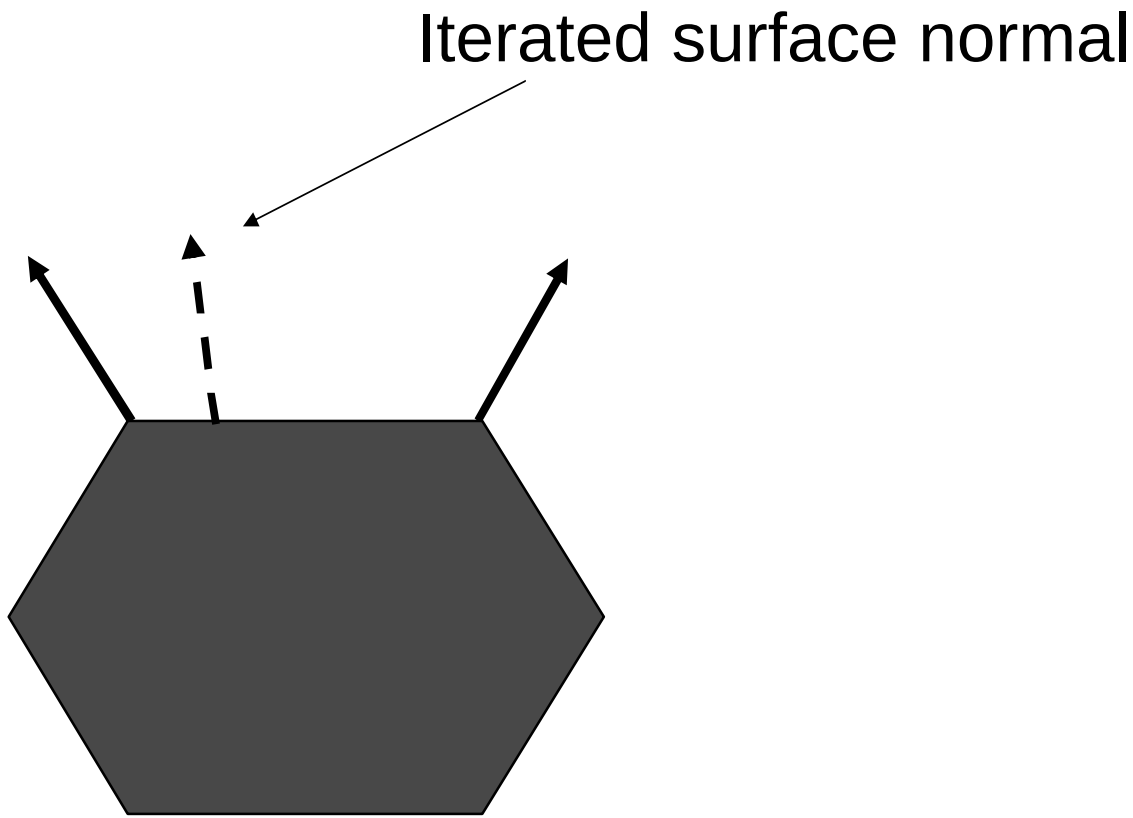
Multiple normals on a cube corner

Surface Normals

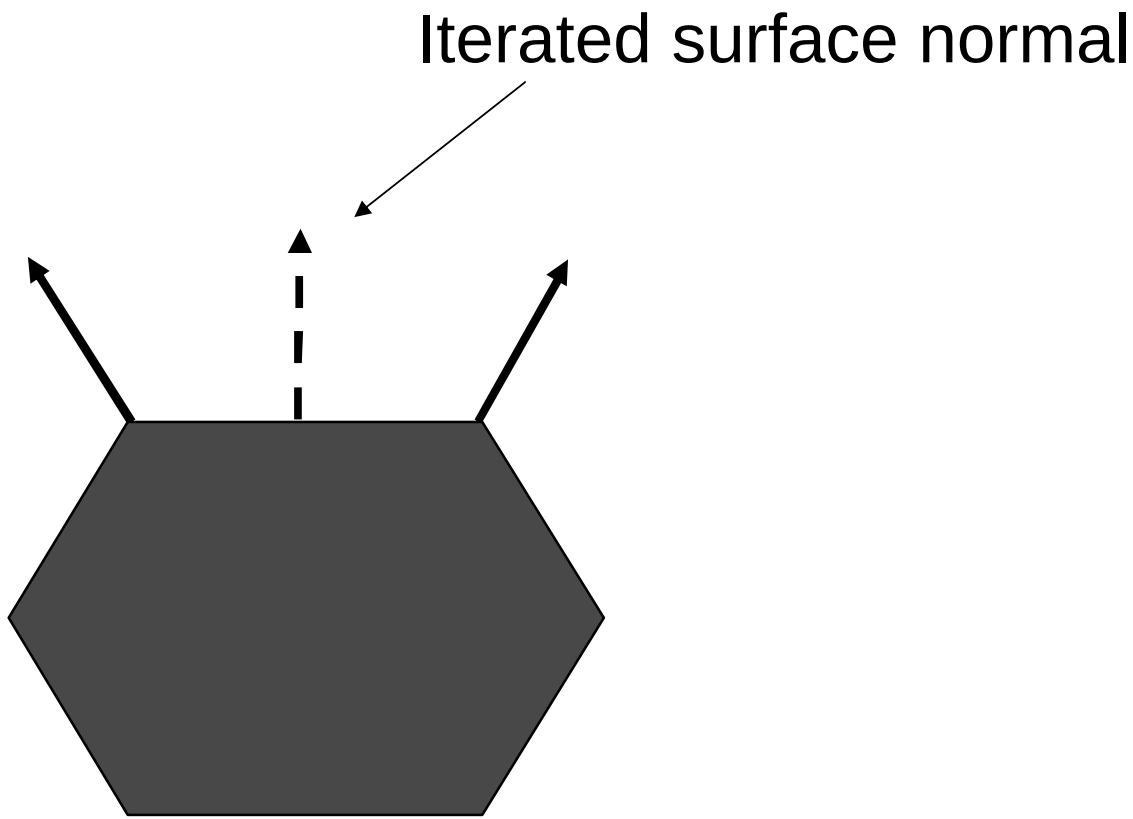
- **Normals are interpolated across a face (Barycentric)**
- **Always length 1.0**



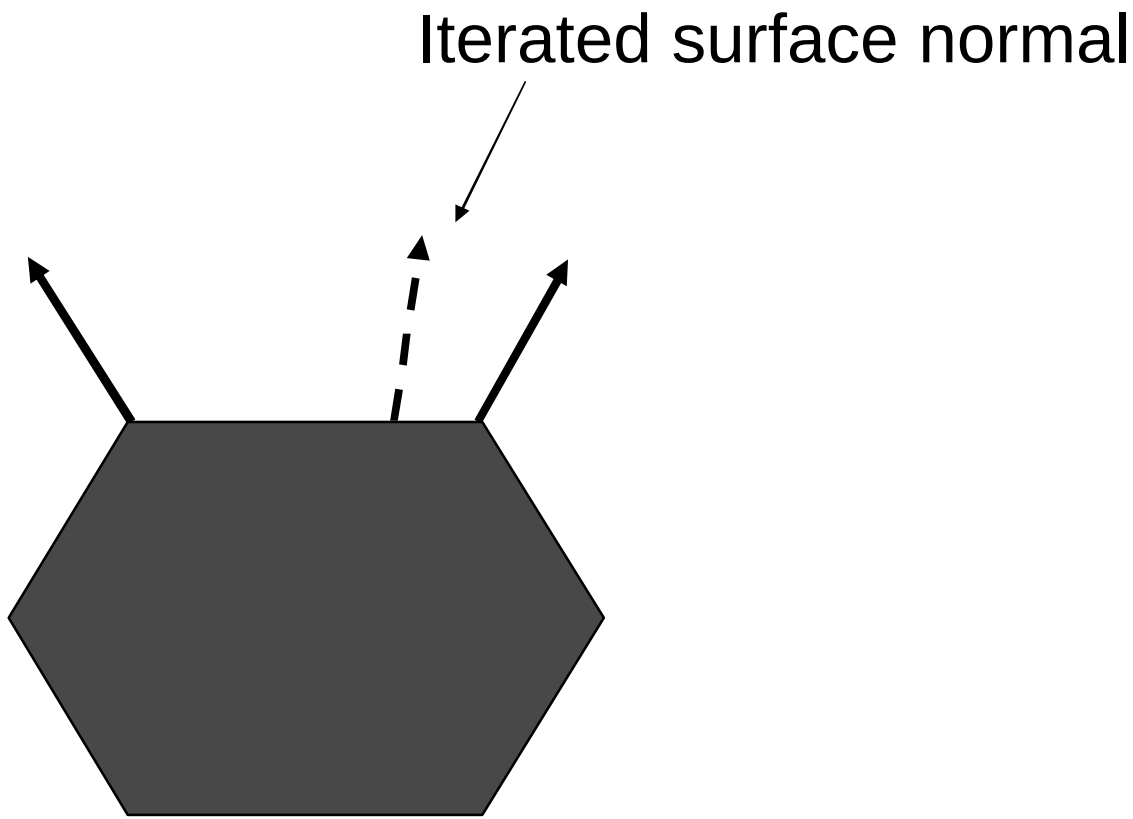
Iterate surface normal



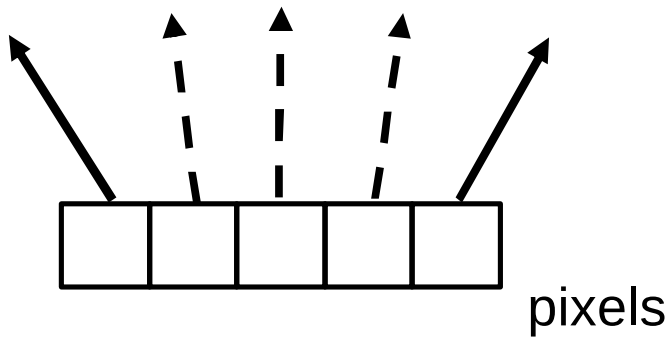
Iterate surface normal



Iterate surface normal

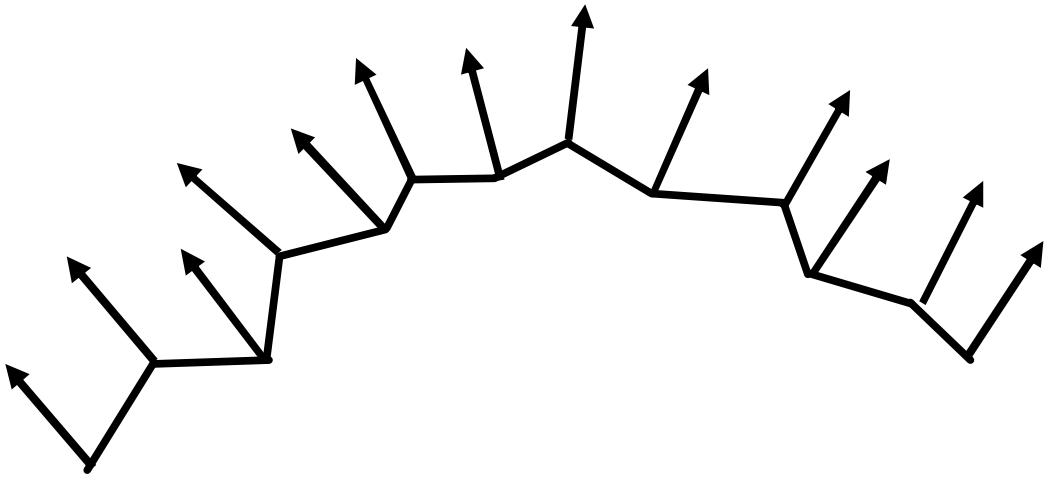


Iterate surface normal calculated per pixel



Getting the hi-res normals to the low res model

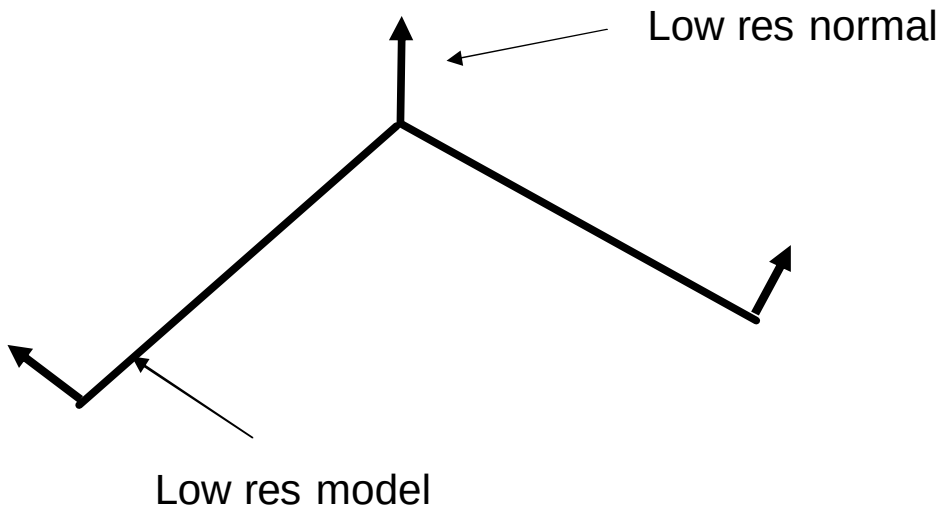
- On the high resolution surface, we have many vertex normals



Hi res model faces

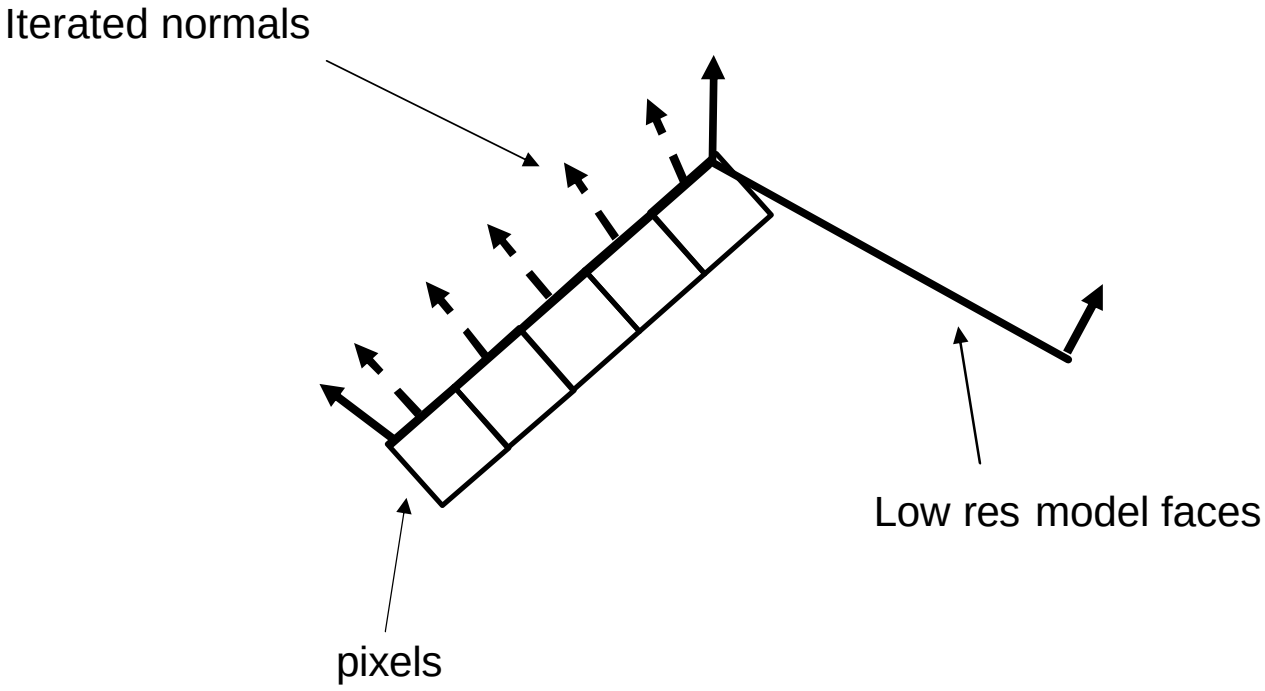
Low res model

- **Fewer vertices, fewer normals**



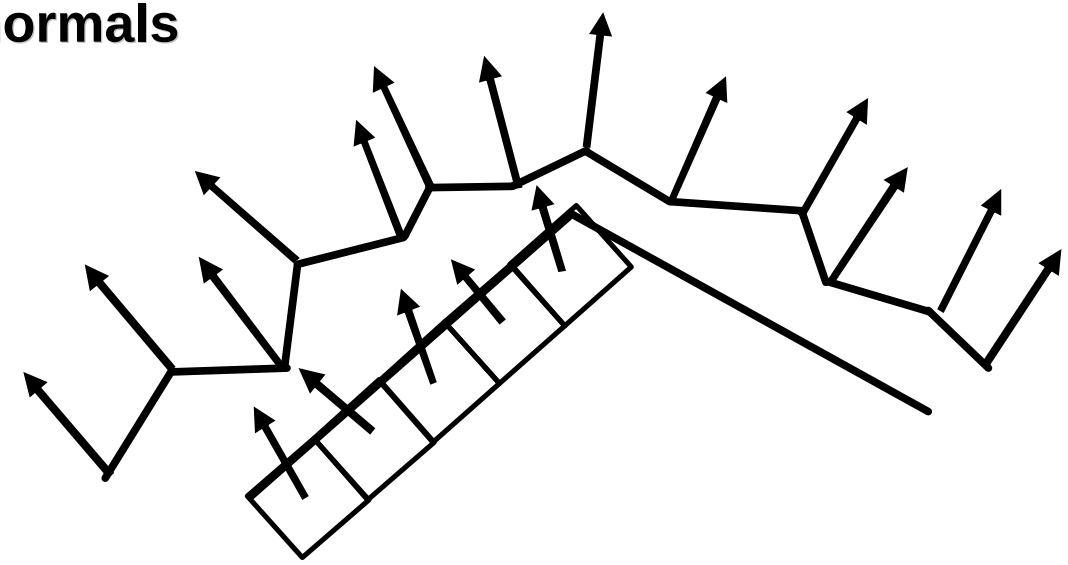
Low res model

- **Iterated normals on low res model contains much less surface detail**



Textures are per-pixel storage

- So grab the normal from the high res surface
- Store that in a texture (called a normal map)
- Use those normals instead of the low-res iterated normals



Properties of Normals

- Normals are 3D vectors (x,y,z)
- Unit length (always 1.0)
- So each vector component has range [-1, 1]

Normal Maps

- **We can store the xyz components of the normal in the RGB color channel of the texture**
- **Map directions [-1,1] to color [0,255]**

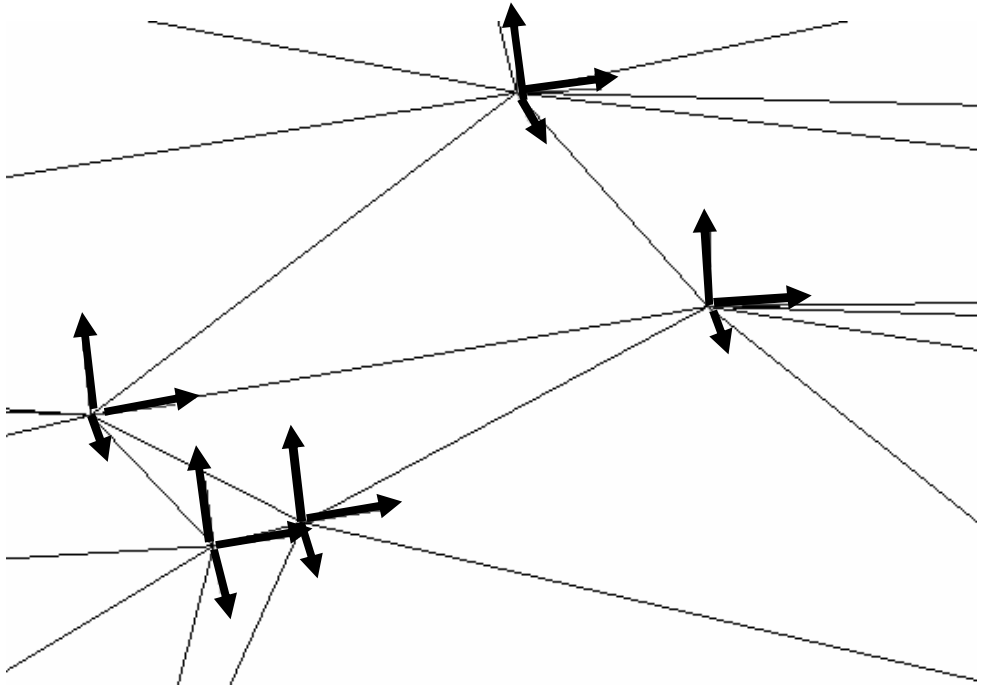
Normal	to	Color
-1		0
0		128
+1		255

Two types of Normals

- **Object Space**
  - Relative the object
- **Tangent Space**
  - Relative to each face
- **Melody creates both**

Tangent Space

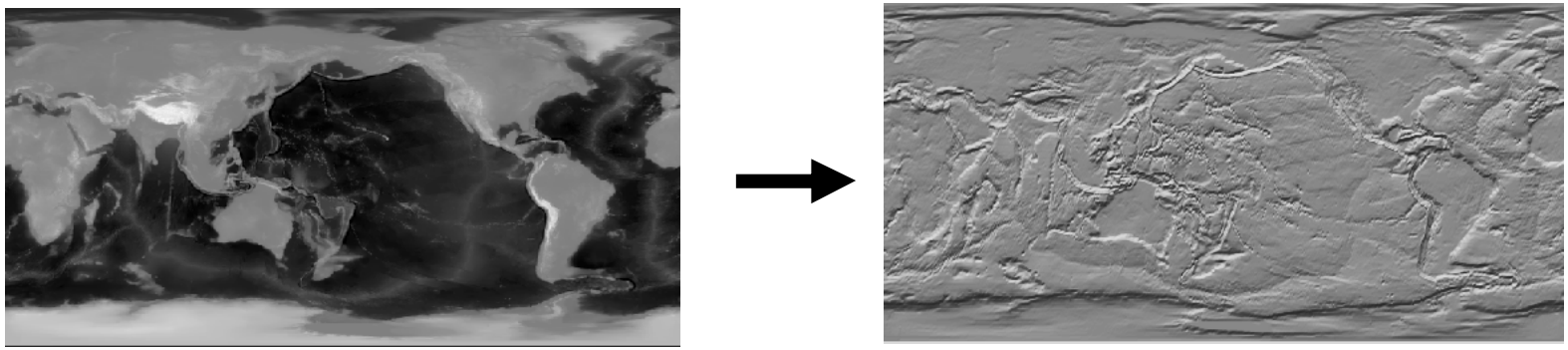
- **Local coordinate system defined per vertex**
- **Allow relative normals, or normals defined in this local space**
- **Rotate the light into this space, or the normal into object space**
- **Coordinate system may be rotated by the deformation**
- **Tangent space transforms object Space to tangent Space**



Tangent, binormal and Normal define tangent basis

Tangent Space Normals

- Texel normals are relative to the face normal
- The vector (0,0,1) is considered the ‘up’ direction and coincident with the face normal
- (0,0,1) is stored as color (128,128,255)
- Can be derived from height maps



- **Use 2D Images, using the Normal Map Plugin for Photoshop**

http://developer.nvidia.com/view.asp?IO=ps_texture_compression_plugin

Normals Displayed as Colors

Tangent space

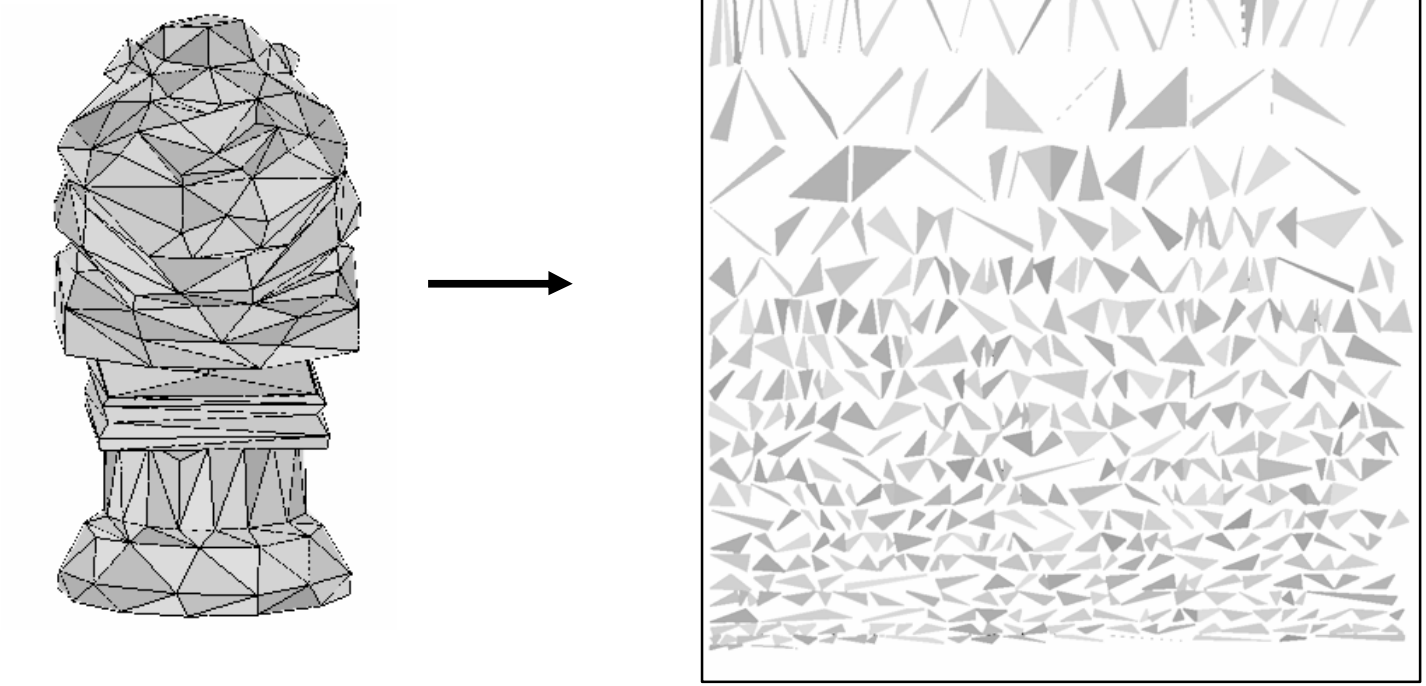
Object space

Normal Maps

- Need to have texture coordinates
  - Artist supplied
  - Automatically created
- Texel cannot be used in more than one place on the model
  - Texels correspond to a position on the model
  - Example, tiling or mirroring is not allowed

Automatic Texture Coordinate Creation

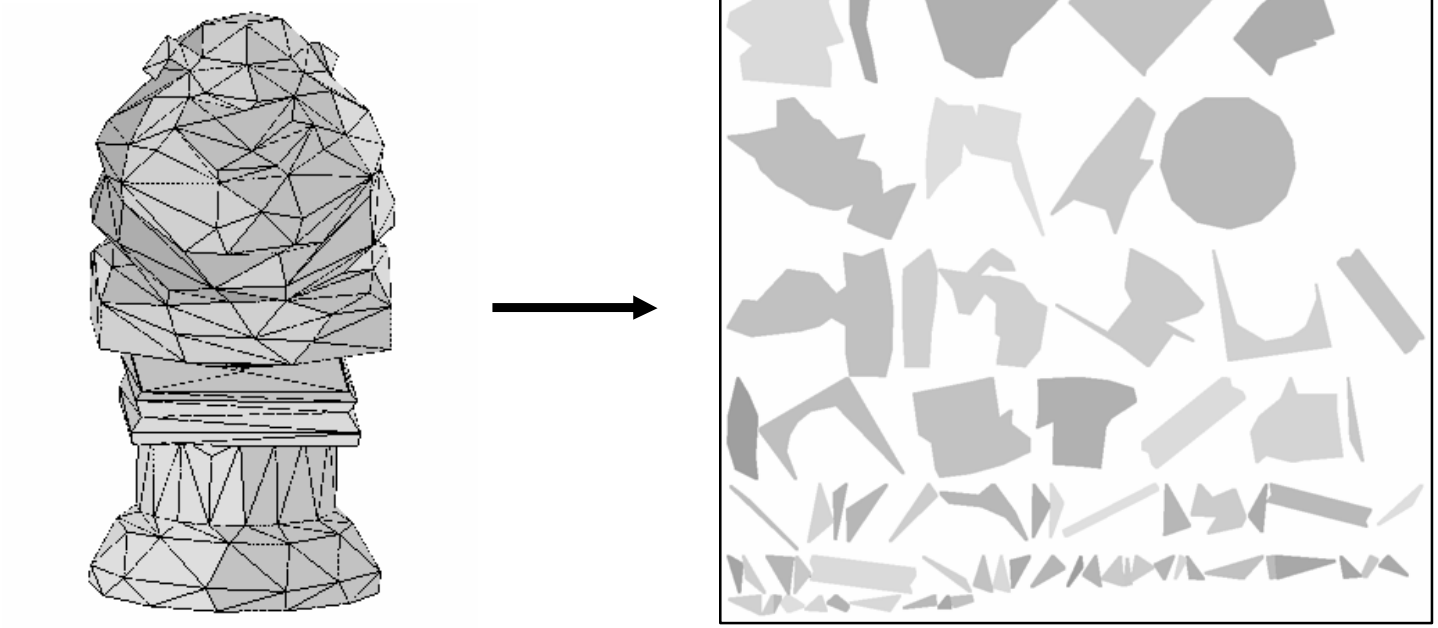
- We could just flatten the triangles and pack them in the texture
- No index reuse (poor cache performance)
- Waste texture space



Resulting texture map

Charts

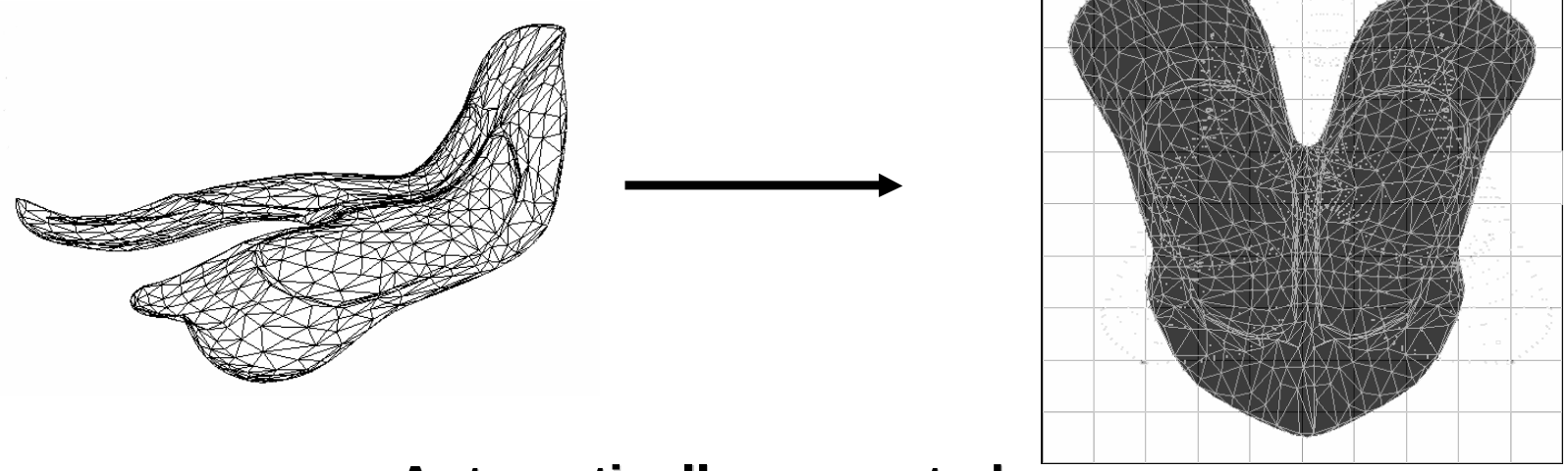
- Instead, we combine the faces to form ‘charts’
- Combine faces until some criteria is met
  - Flatness
  - One perimeter



Resulting texture map

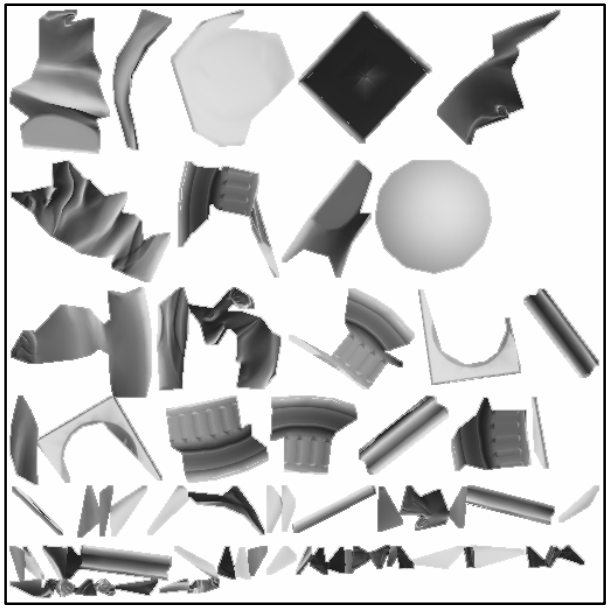
One Chart

- Special case
- Needs one connected border
- Not required to be flat
- No flipped triangles
- Can be automatically generated



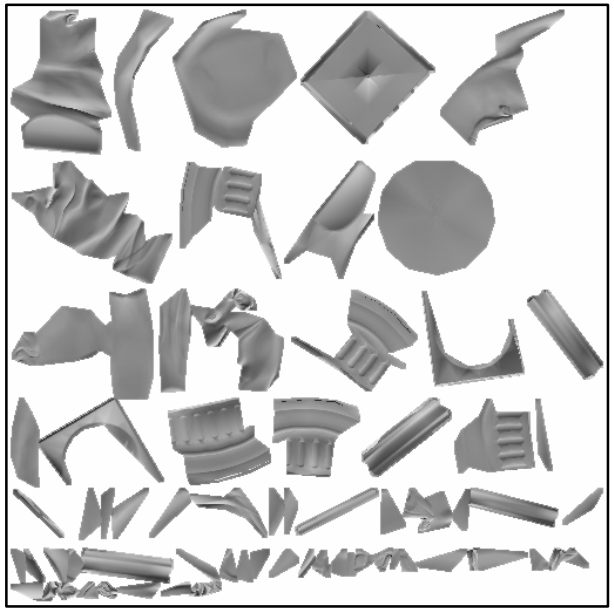
Automatically generated

Normal Map



Object space normal map

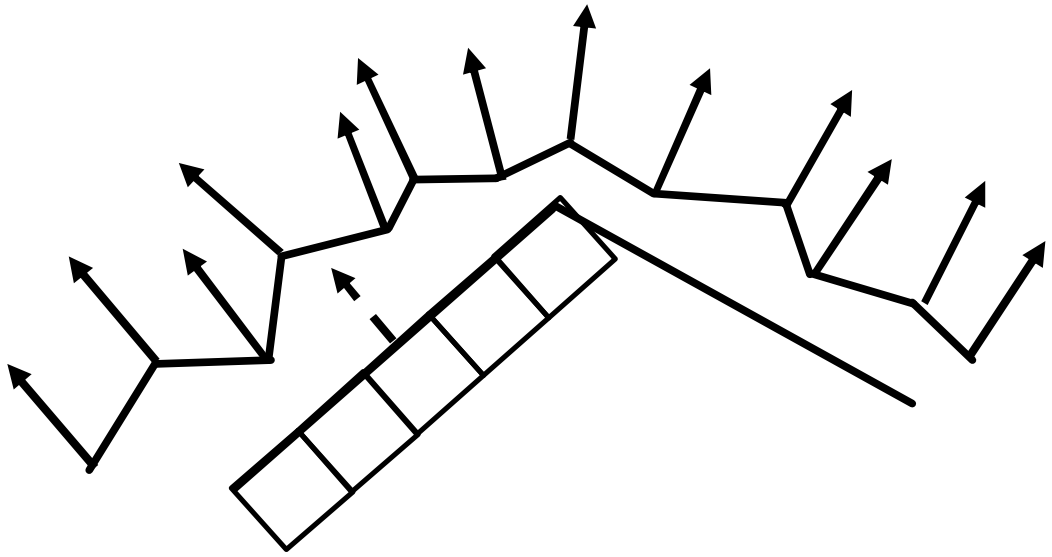
- Now that we have texture coordinates
- Store normals
- Wait! How did we get the normals to store in the texture?



Tangent space normal map

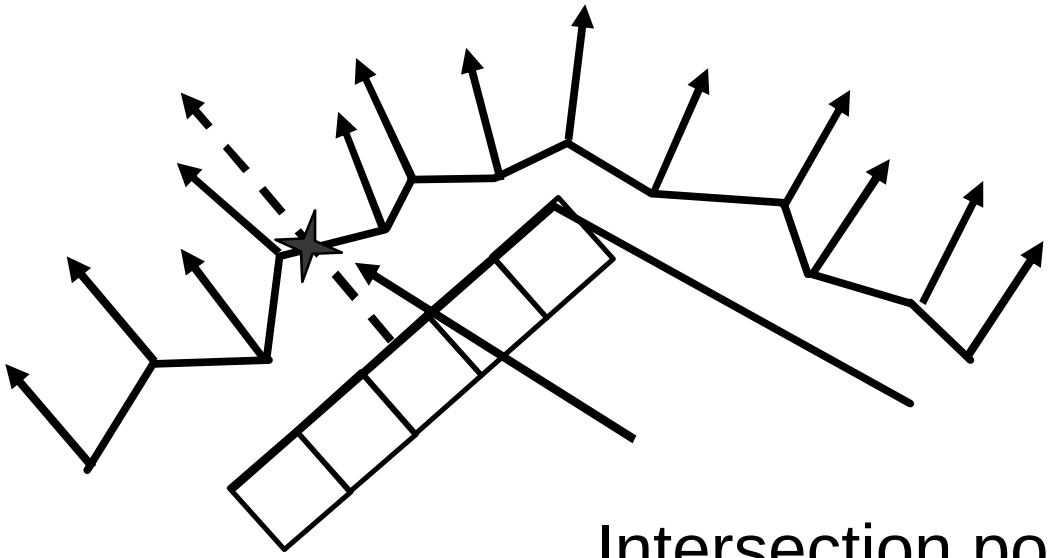
Ray Casting

- **Follow the iterated normal from the lo-res surface to the hires surface**



Ray Casting

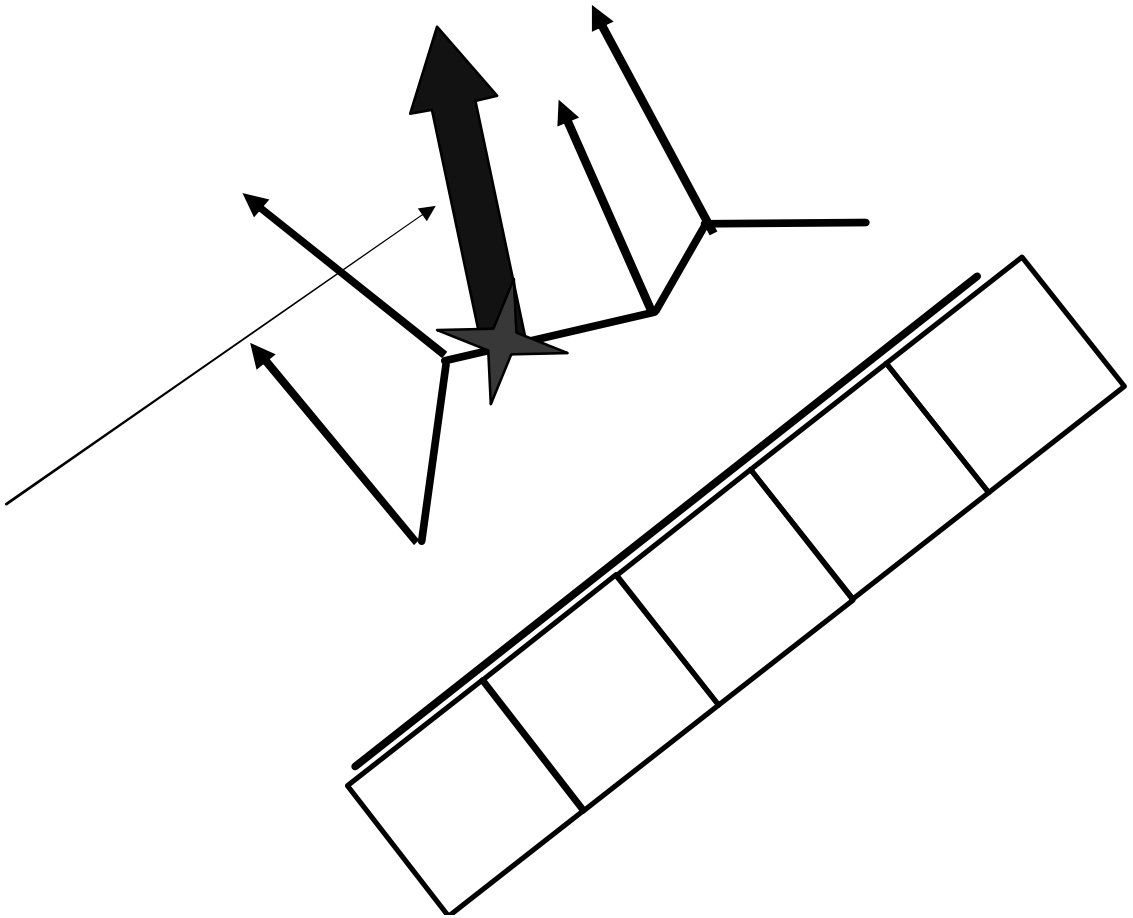
- **Extend the iterated normal to find the intersection of the hi res model**



Ray Casting

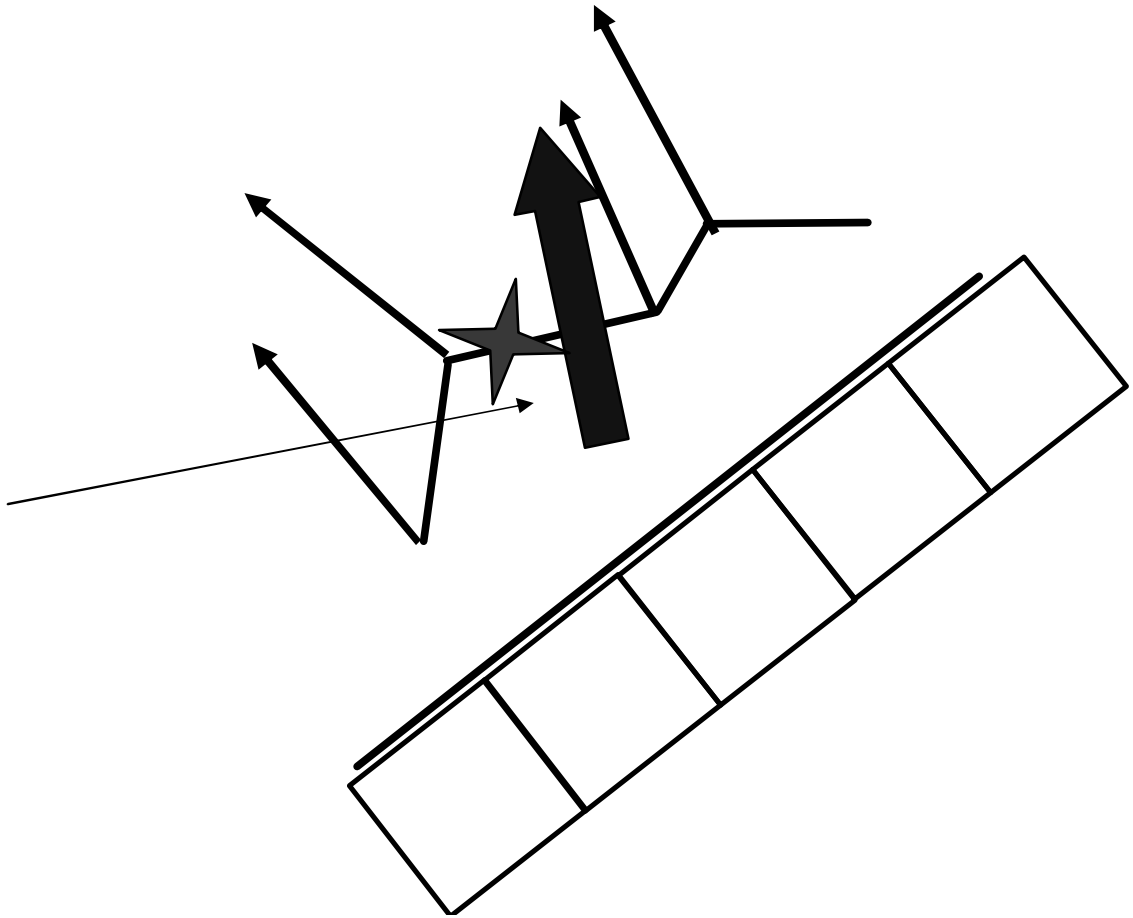
- **Calculate the iterated normal for the hi res model at the intersection point**

Calculate hi-res normal at intersection point



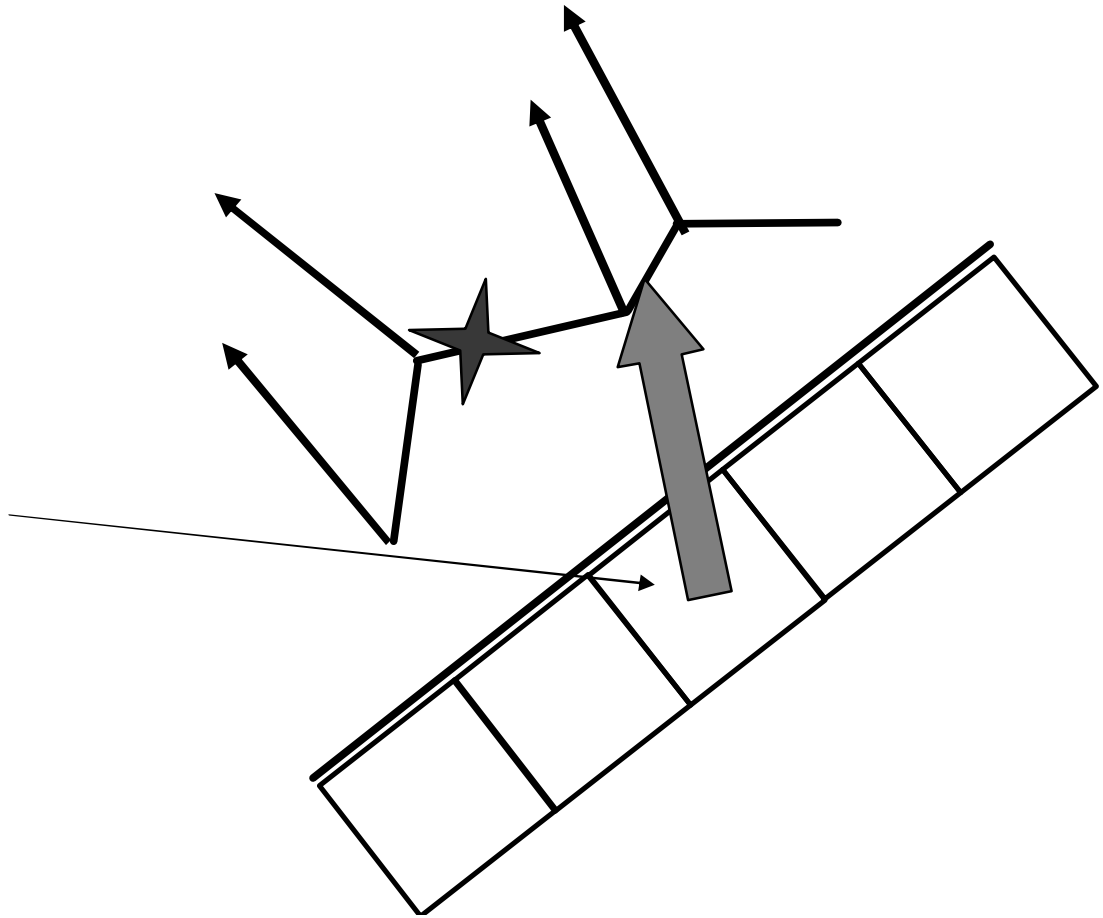
Ray Casting

Store this in…

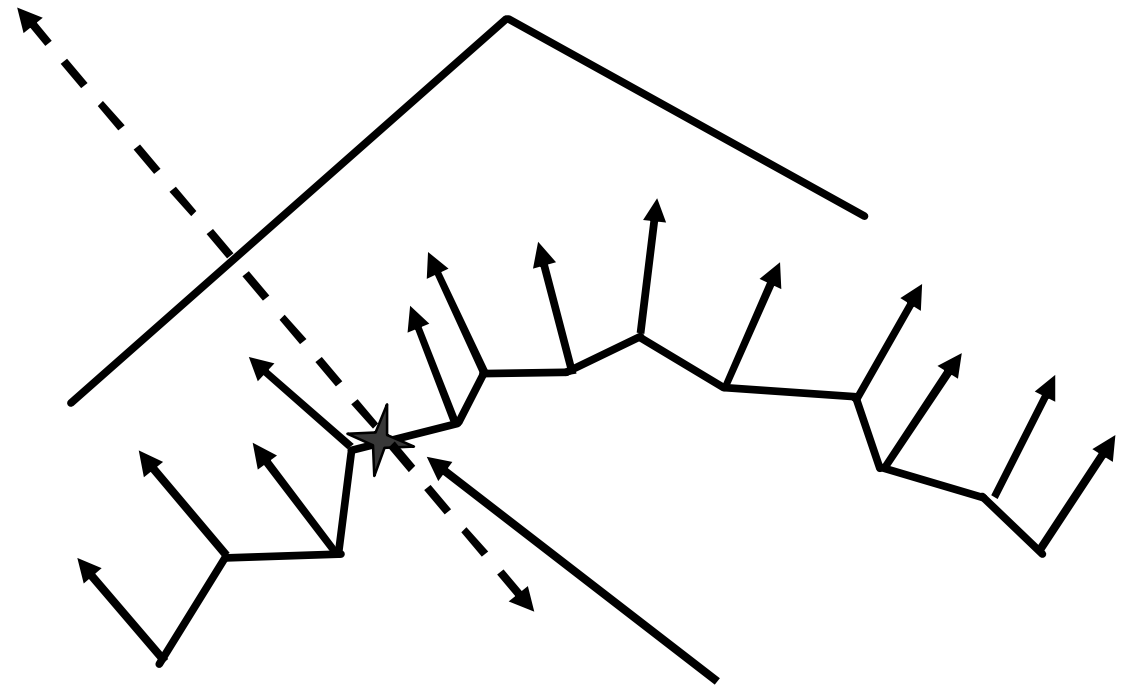


Ray Casting

the normal map



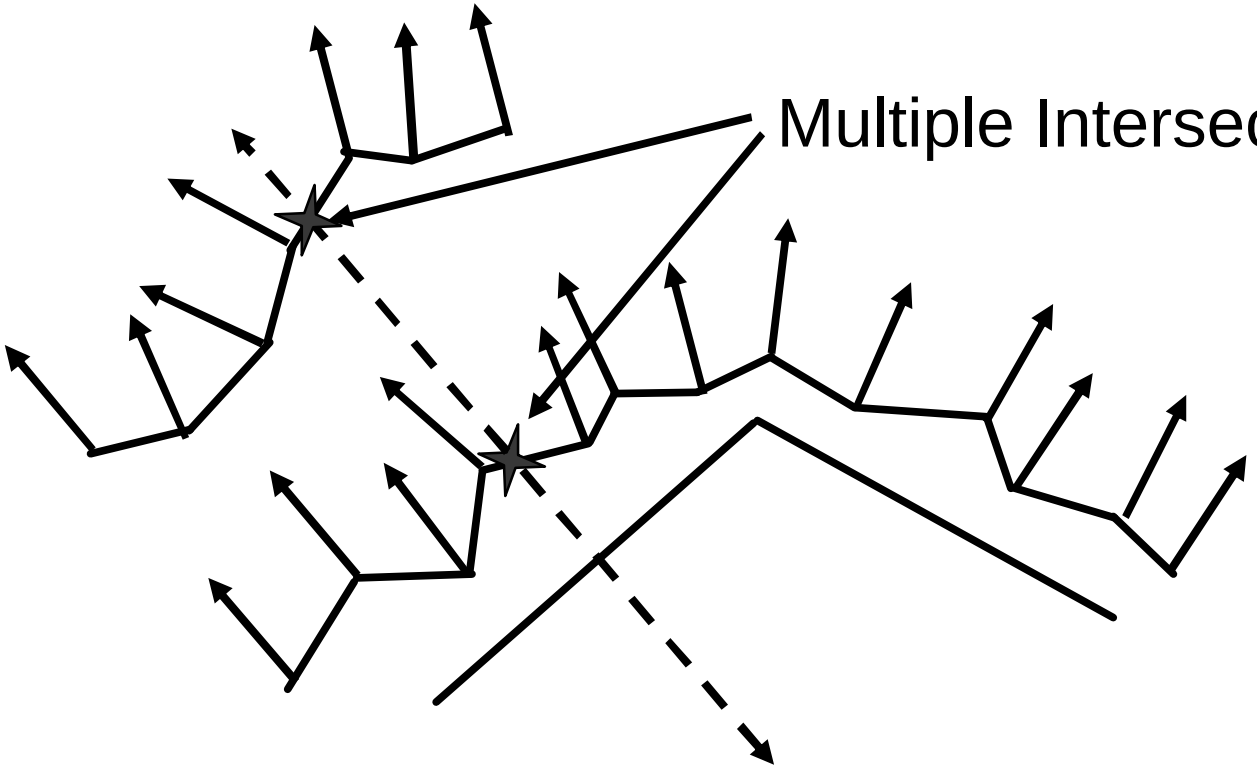
What can happen in during raycasting



Intersection point behind ray

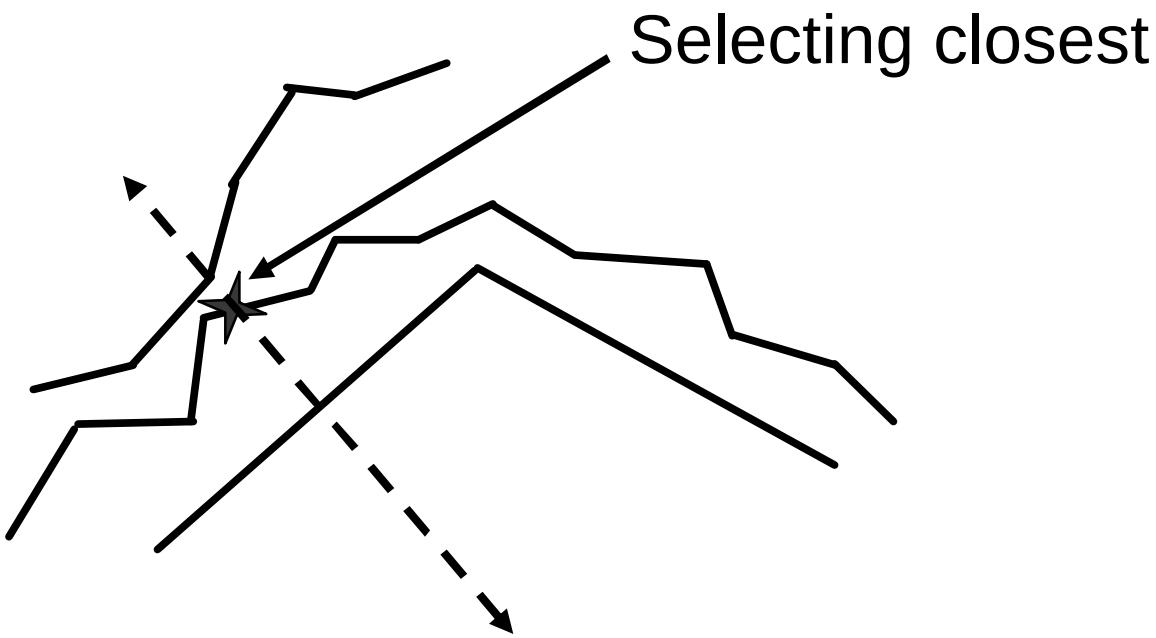
- **Allow ray to go forward and backwards**

What can happen in during raycasting

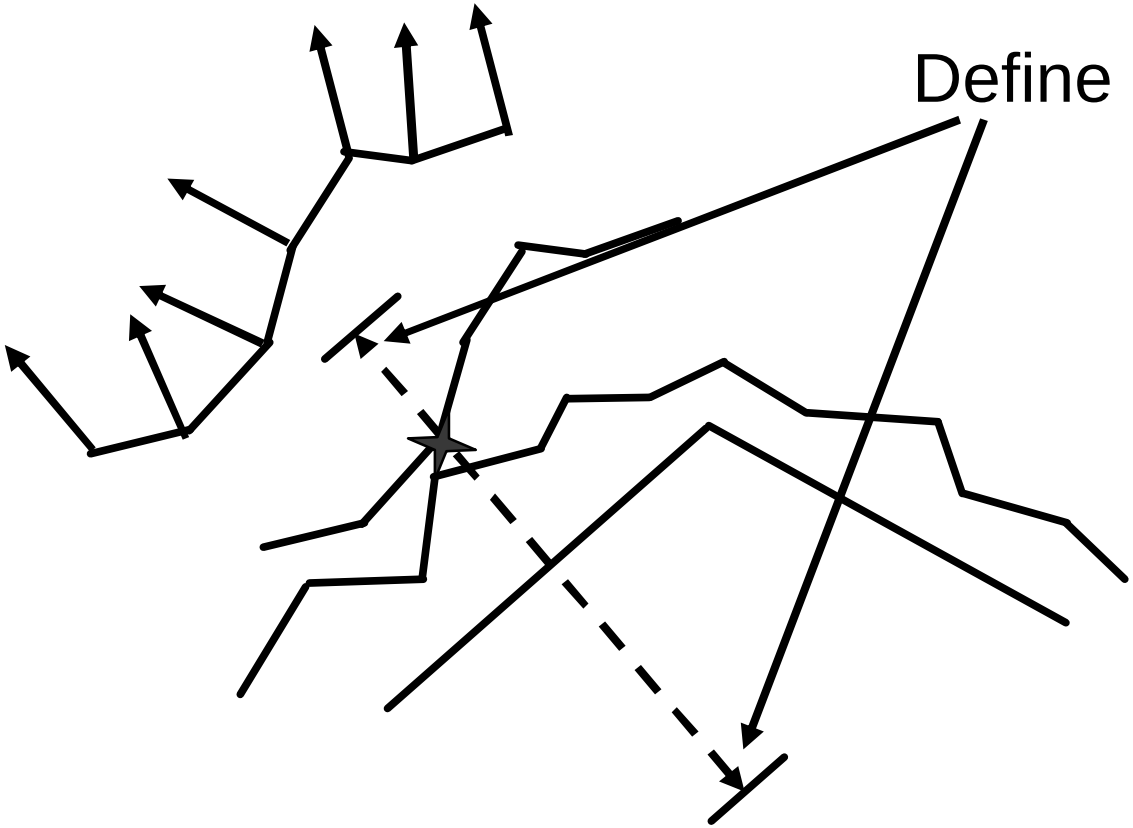


Chosing closest intersection

- This might not be what you want
- May hide the detail on the top most surface

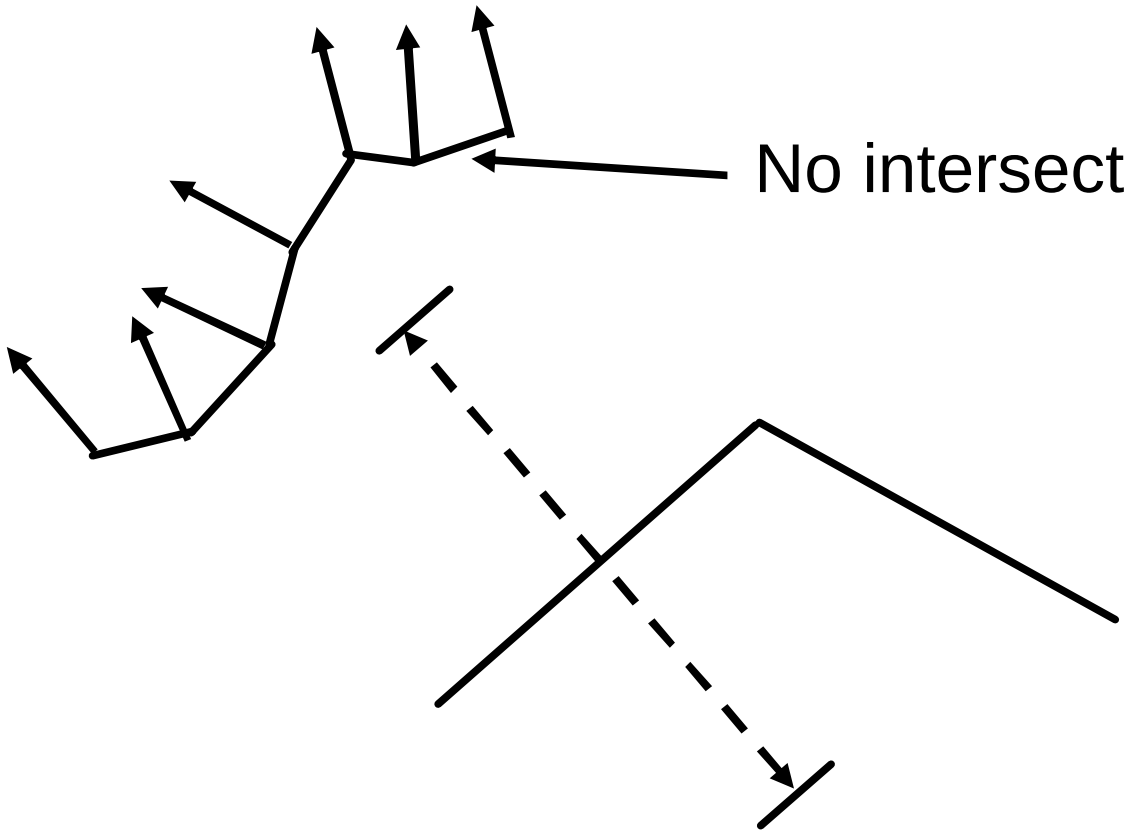


Restricting ray length



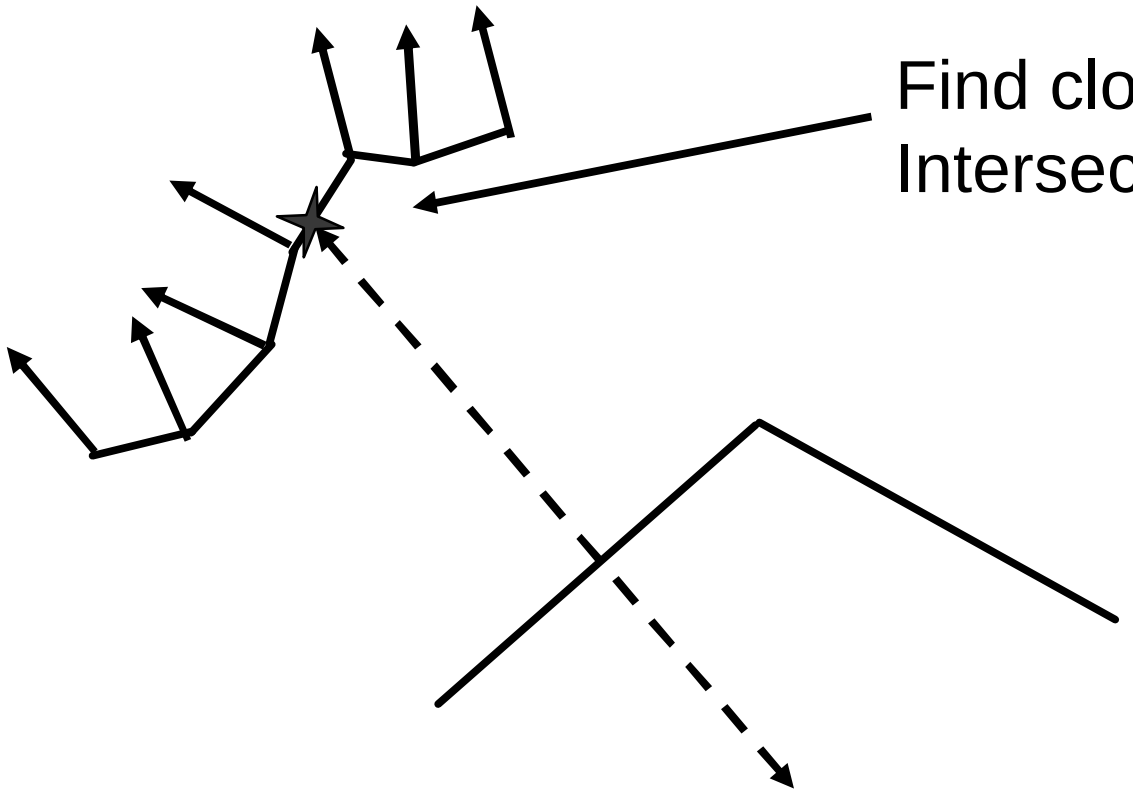
- Chose farthest within bounds

Restricting ray length



No intersection within bounds

Restricting ray length



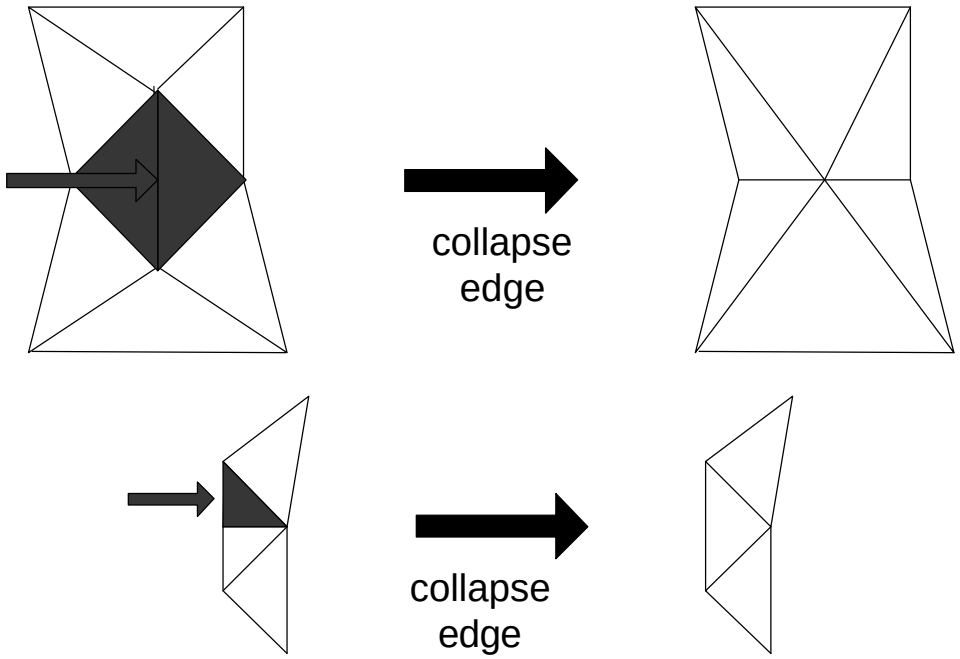
Find closest intersection if no Intersection within bounds

Simplification

- **Attribute discontinuities**
  - Texture
  - Normal
  - Color
- **Features**
  - Sharp Edges
  - Seams
- **Boundaries – perimeter of material**

Simplification – Edge Collapse

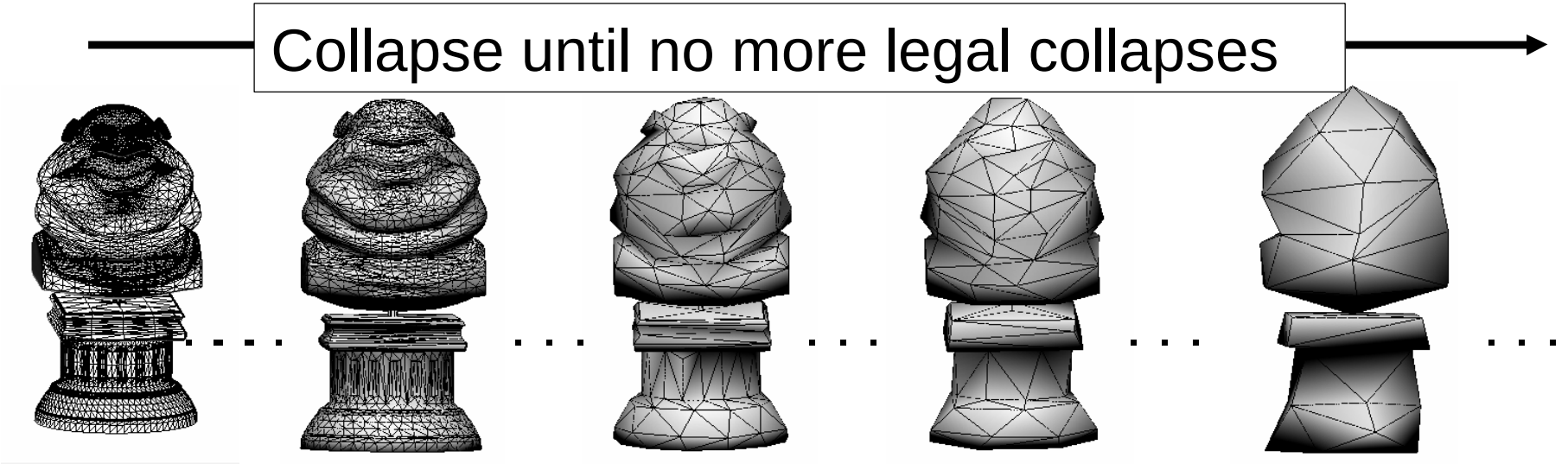
- Edge collapse method
- removes one or two faces



Simplification Option

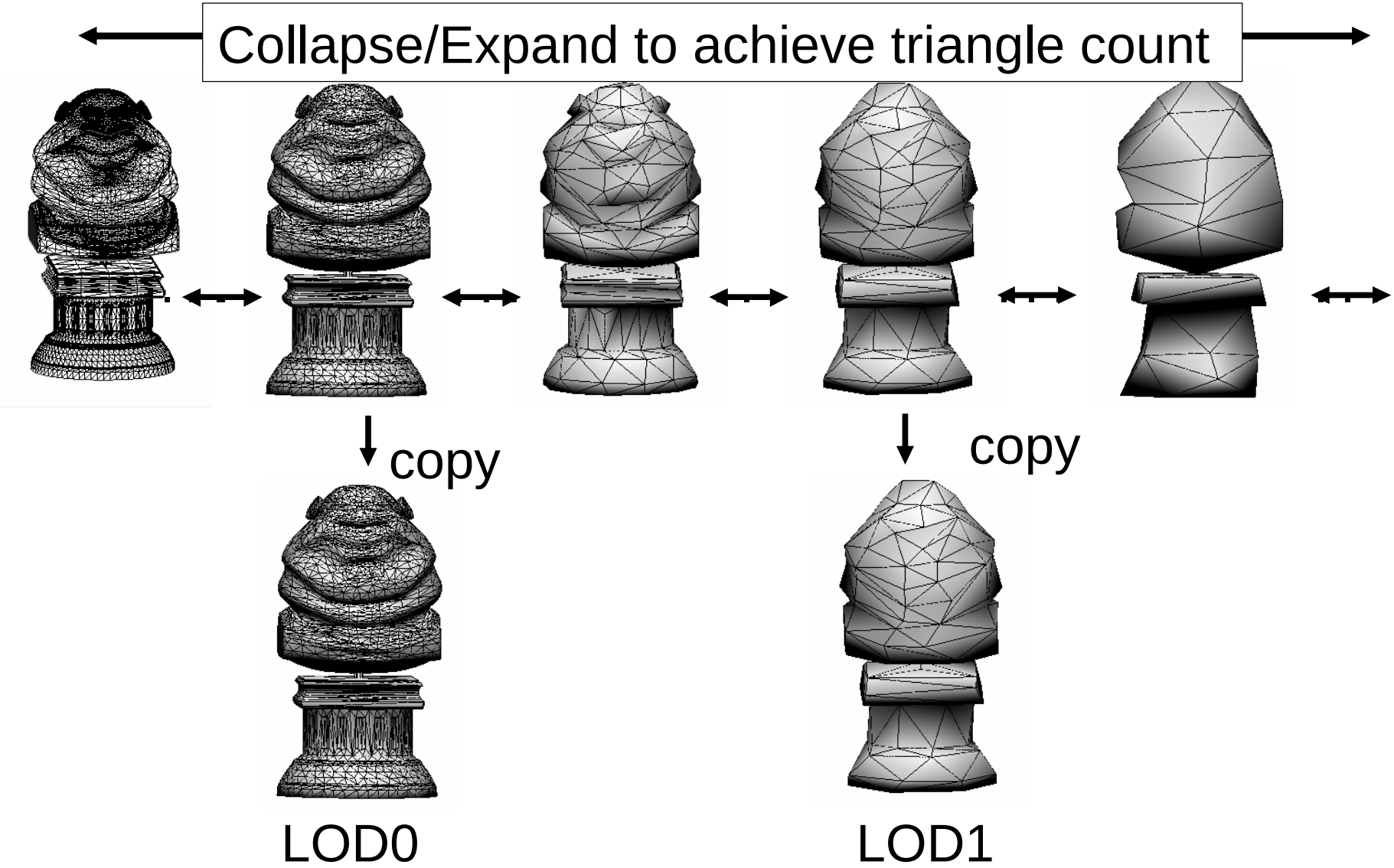
- **Attempt to preserve seams, sharp edges or boundaries**
- **Check topology after collapse so no illformed geometry is created**
- **When edge is collapsed (p2→ p1), placement of p1:**
  - Optimal position
  - Any where along edge
  - Endpoints (p1 or p2) or edge midpoint
  - Endpoints only (vertex removal). Can be used if you have weighted vertices

Progressive Mesh



- Edge collapses based on weight
- Collapse each edge, one at a time and record all collapses
- Allows undo of collapses
- Assignment to LOD for additional processing

Save to LOD from Progressive Mesh

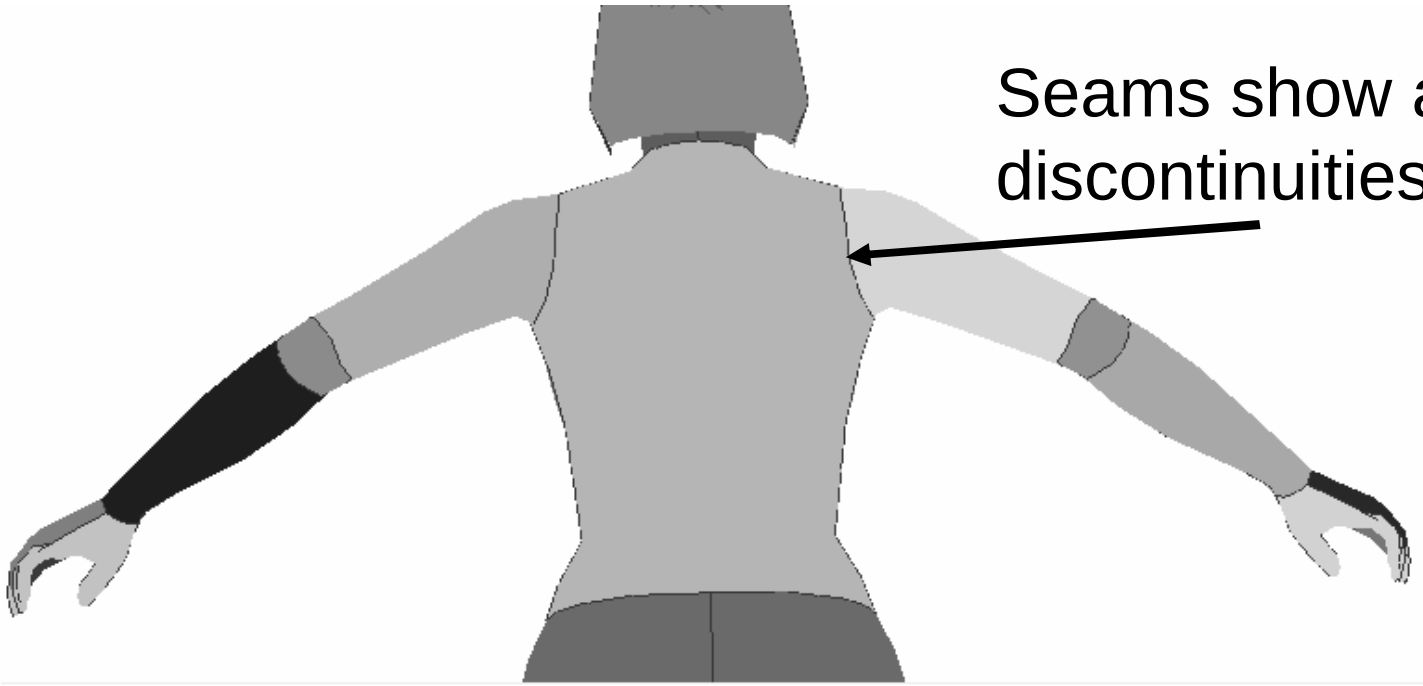


Melody Simplification Methods

- **Quadric Error Metric (QEM)**
  - **Move new point to closest point on all faces**
- **Volume Preservation**
  - **Maintain volume of model**
- **Energy Minimization (EMIN)**
  - **Minimize new faces to data points sampled from reference model. Slow**

Seam

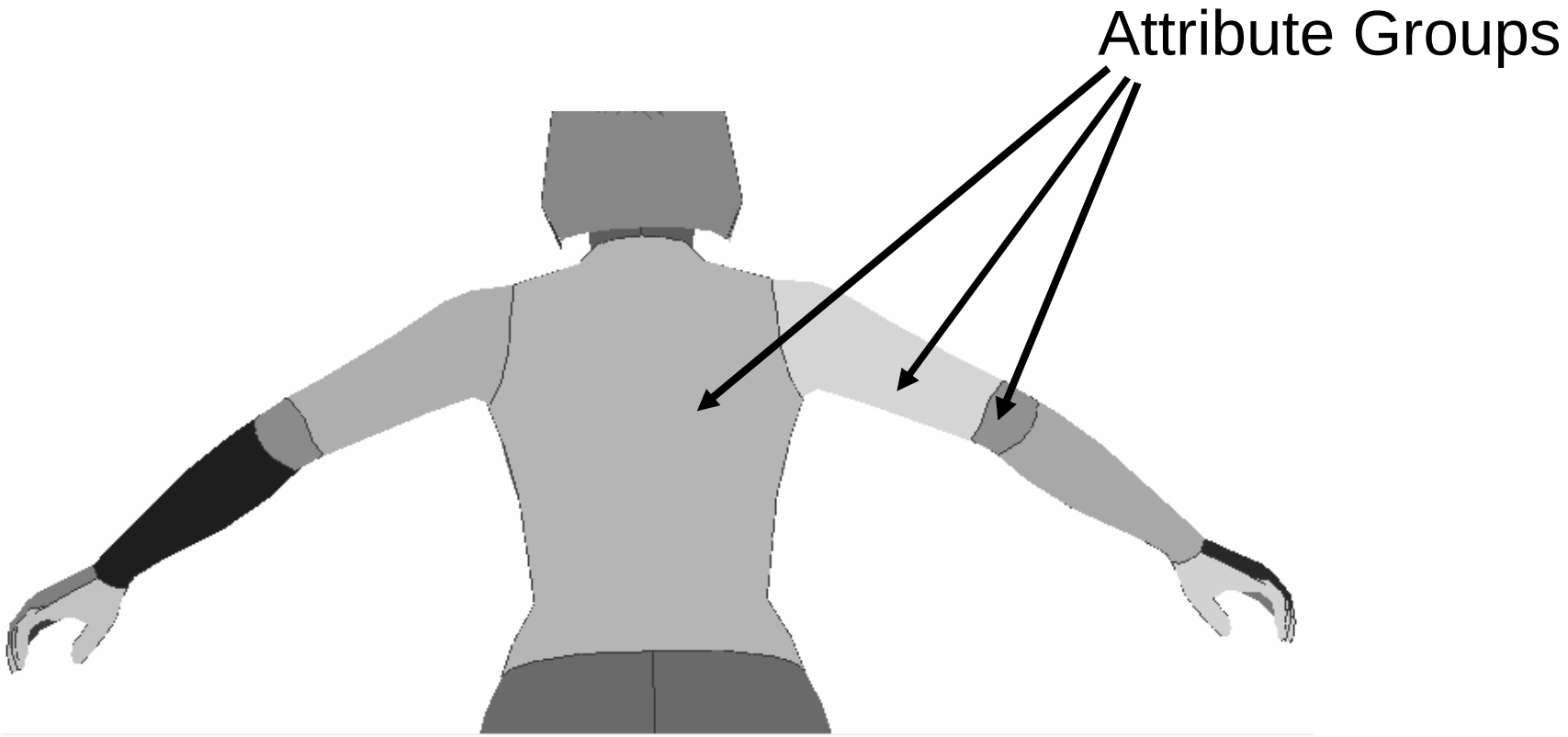
- Any vertex that shares a position with another vertex and all the attributes do not match
- Discontinuity in color, texture coordinates, normal etc.
- Simplification can try to maintain seam positions



Seams show attribute discontinuities. Show in red

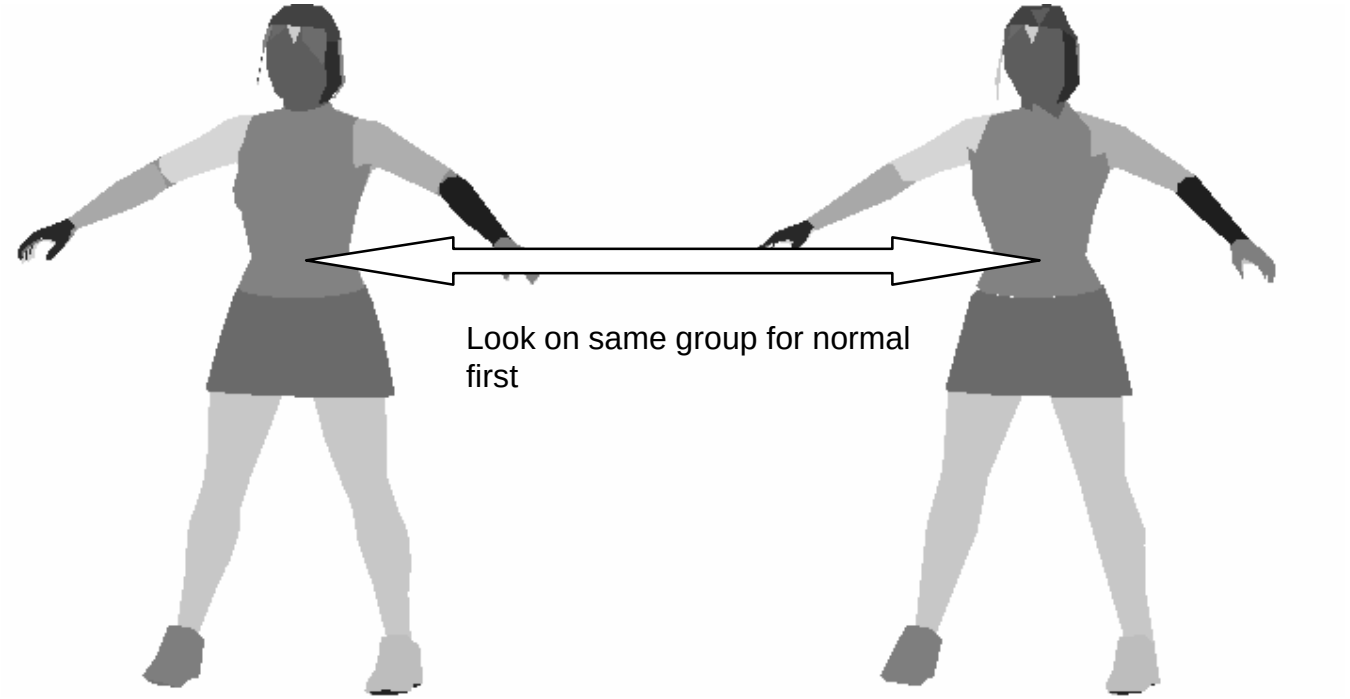
Attribute Group

- **A group of faces that are bordered by seams**



Attribute Group Matching

- **Match Attribute Groups from low res model to reference model. Fetches correct normal**
- **For Simplification and Normal Map Generation**

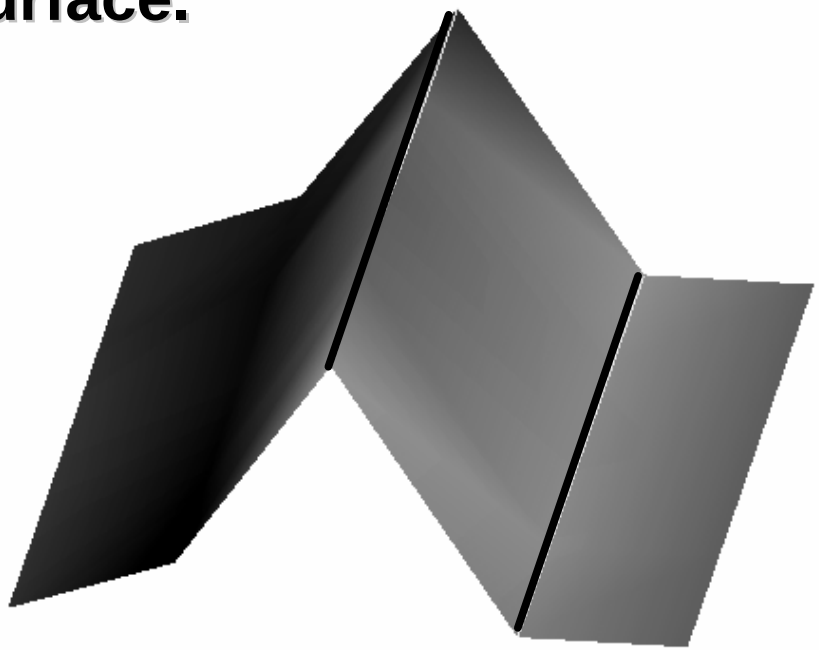


Reference Model

Lo res model

Sharp Edge

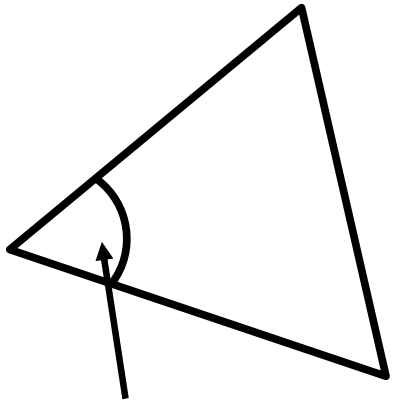
- **Angle between two faces that exceeds a specified values**
- **Typical values for sharp edge is 140 degrees. 180 is a flat surface.**



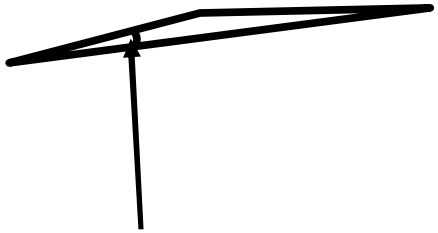
Sharp edges shown as yellow

Corner Angle fatter lines.

- Angle between two edges on one face
- Option not to generate during simplification
- Small corner angles produce slivers
  - Very small area/perimeter ratio
  - Poor GPU performance



Corner angle



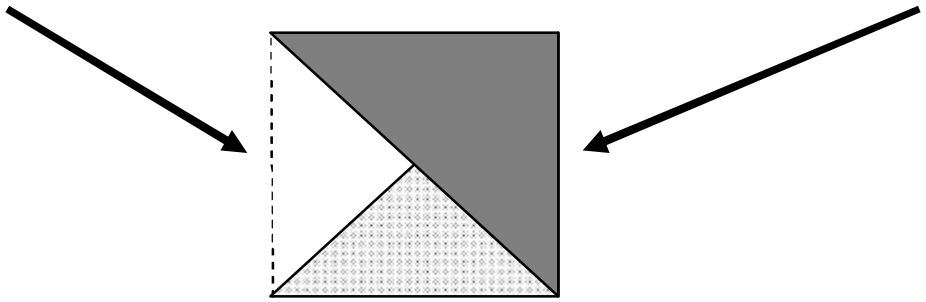
Sliver: small corner angle

Boundary

- Edge that has only one face attached
- Defines the perimeter of a model

Illformed Geometry: Folded Edge

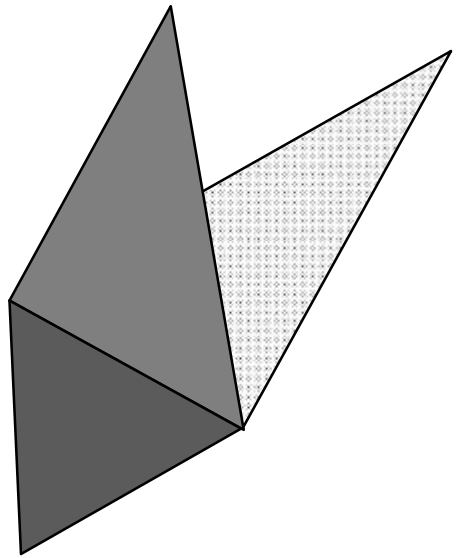
- **Commonly called “BowTie”**
- **When two faces**
  - Share an edge
  - and are Coplanar
  - and Face in opposite directions
- **Melody can simplify this, but its slower**



Shared edges between green and blue faces

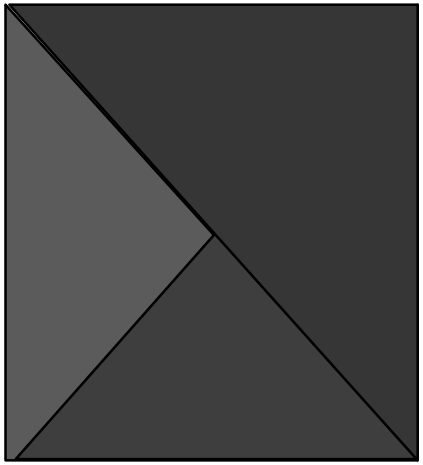
Illconditioned Geometry

- Three or more faces sharing an edge
- Melody can simplify this, but its slower
- Non 2 manifold

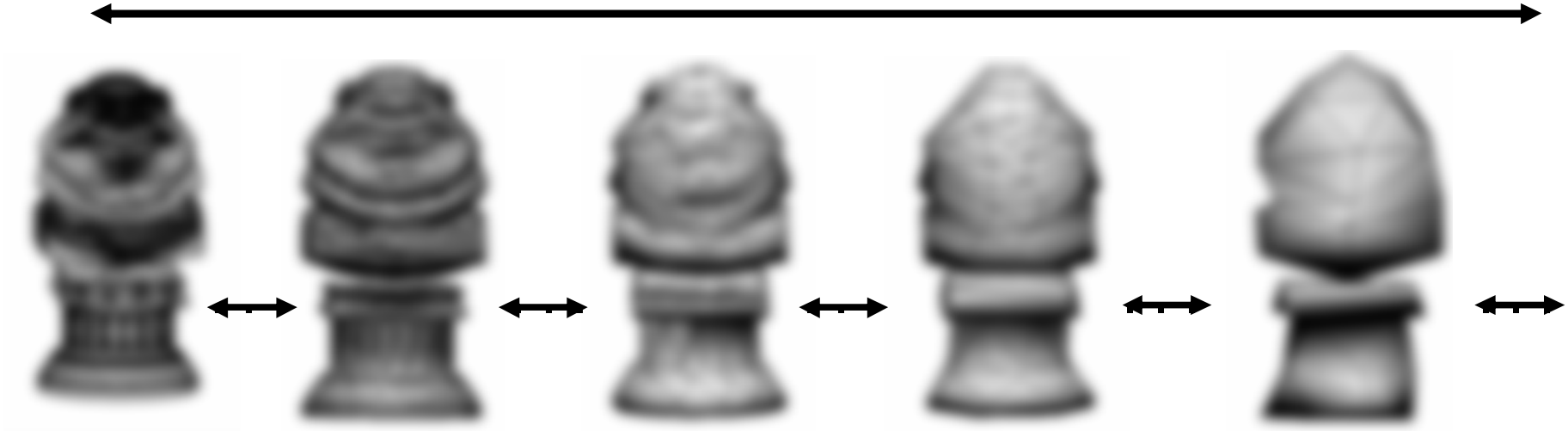


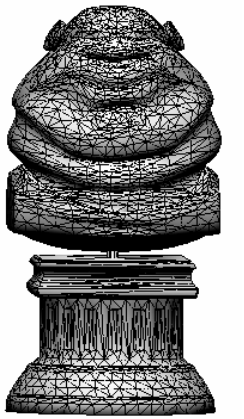
Illconditioned Geometry

- T-junctions
- Vertex splits the edge on the red face



After LOD Creation



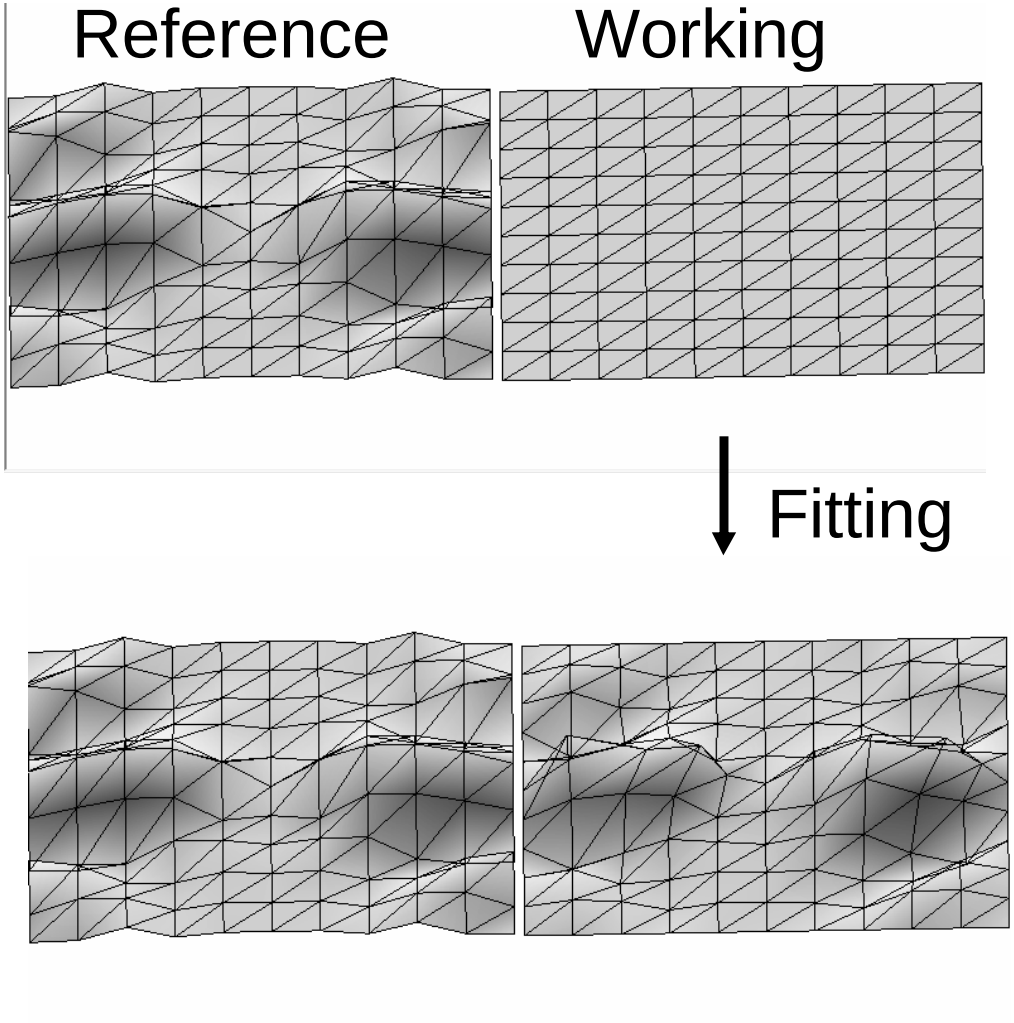


LOD

- Normal Map Creation
- Model Optimization

Fitting to a Reference Model

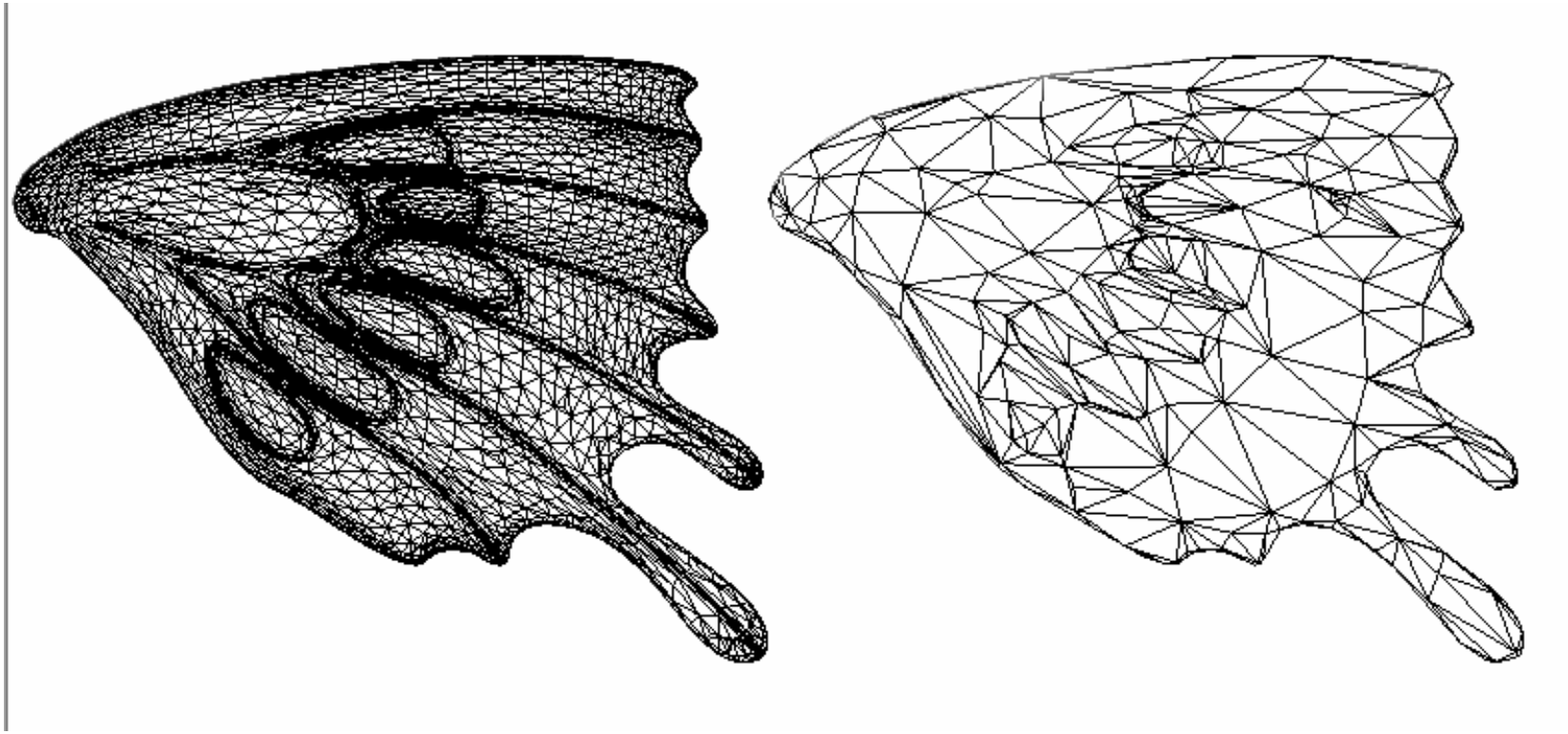
- After simplification, fit all vertices in the working model to the reference model
- Tries to make a better match of low res models to high res models
- This just moves vertices



Model Optimizations

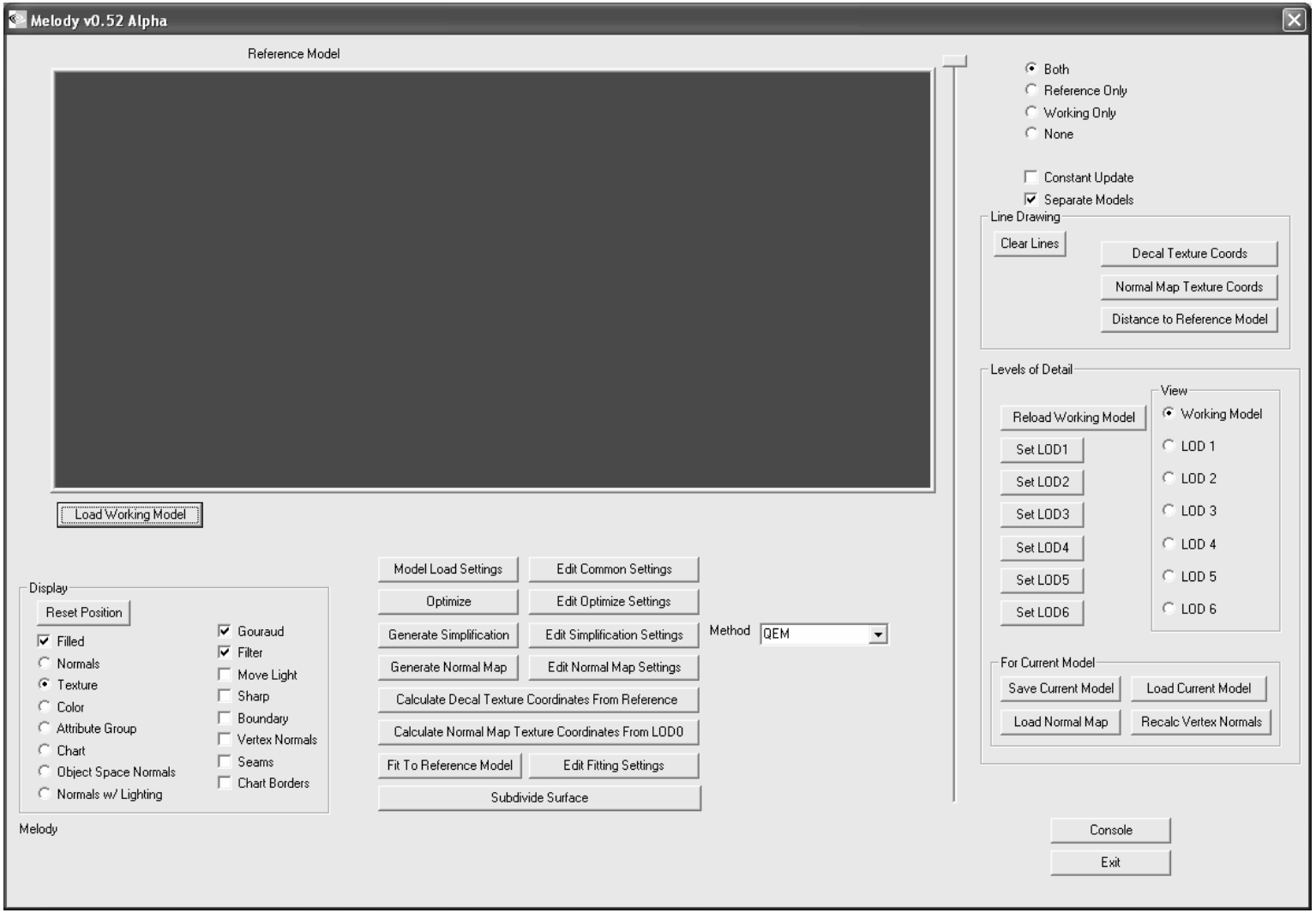
- **Fitting to reference model**
  - **Move vertex only**
- **Geometry Optimization**
  - NOT simplification
  - Adds or removes geometry
    - Split Edge (create vertex and 4 edges)
    - Flip Edge
    - Collapse Edge (remove two face)
  - Solves a spring based system
  - Hugues Hoppe’s technique

Optimize example



optimize

Melody Demo


Melody v0.52 Alpha
Reference Model
Load Working Model
Both
Reference Only
Working Only
None
Constant Update
Separate Models
Line Drawing
Clear Lines
Decal Texture Coords
Normal Map Texture Coords
Distance to Reference Model
Levels of Detail
Reload Working Model
Set LOD1
Set LOD2
Set LOD3
Set LOD4
Set LOD5
Set LOD6
View
Working Model
LOD 1
LOD 2
LOD 3
LOD 4
LOD 5
LOD 6
For Current Model
Save Current Model
Load Current Model
Load Normal Map
Recalc Vertex Normals
Display
Reset Position
Filled
Normals
Texture
Color
Attribute Group
Chart
Object Space Normals
Normals w/ Lighting
Gouraud
Filter
Move Light
Sharp
Boundary
Vertex Normals
Seams
Chart Borders
Model Load Settings
Edit Common Settings
Optimize
Edit Optimize Settings
Generate Simplification
Edit Simplification Settings
Method
QEM
Generate Normal Map
Edit Normal Map Settings
Calculate Decal Texture Coordinates From Reference
Calculate Normal Map Texture Coordinates From LOD0
Fit To Reference Model
Edit Fitting Settings
Subdivide Surface
Melody
Console
Exit

Links to Related Material

- http://developer.nvidia.com/
- http://research.microsoft.com/~hhoppe/
- http://talika.eii.us.es/~titan/magica/
- http://www.cbloom.com/3d/galaxy3/index.html
- http://mirror.ati.com/developer/index.html
- http://www.okino.com/conv/conv.htm
- http://graphics.cs.uiuc.edu/~garland/research/quadrics.html
- http://gts.sourceforge.net
- http://www.loria.fr/~levy/Papers/2002/s2002_lscm.pdf
- http://deas.harvard.edu/~pvs/research/tmpm/
- http://developer.nvidia.com/view.asp?IO=ps_texture_compression_plugin

Future

- Extreme simplification using Hull
- Subdivision surface + displacement map generation
- Better chart creation and packing
- Reduce memory usage
- Command line version
- DCC integration