Polygonal Meshes

COS 426, Spring 2022

Princeton University

Felix Heide

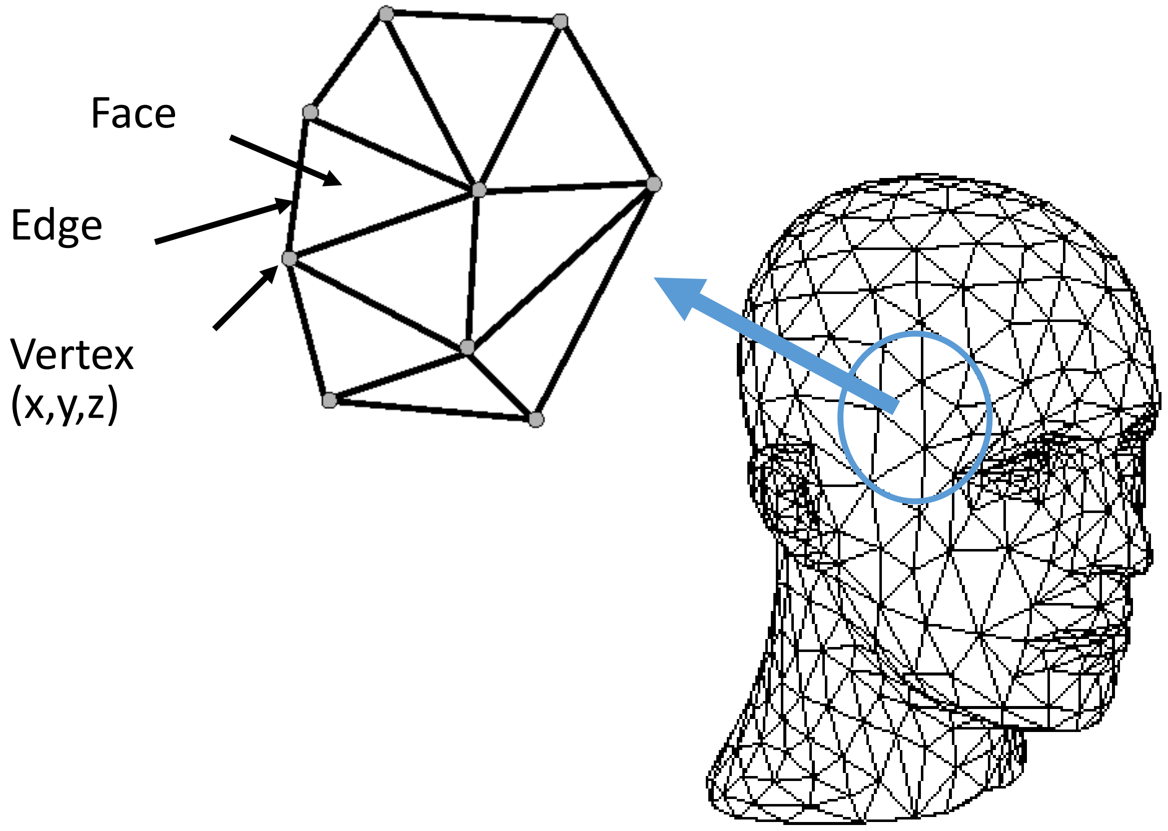
DEADPOOL
20TH CENTURY FOX (2016)

3D Object Representations

- Points
  - Range image
  - Point cloud
- Surfaces
  - ➢ Polygonal mesh
  - Parametric
  - Subdivision
  - Implicit
- Solids
  - Voxels
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific

3D Polygonal Mesh

- Set of polygons representing a 2D surface embedded in 3D



Zorin & Schroeder

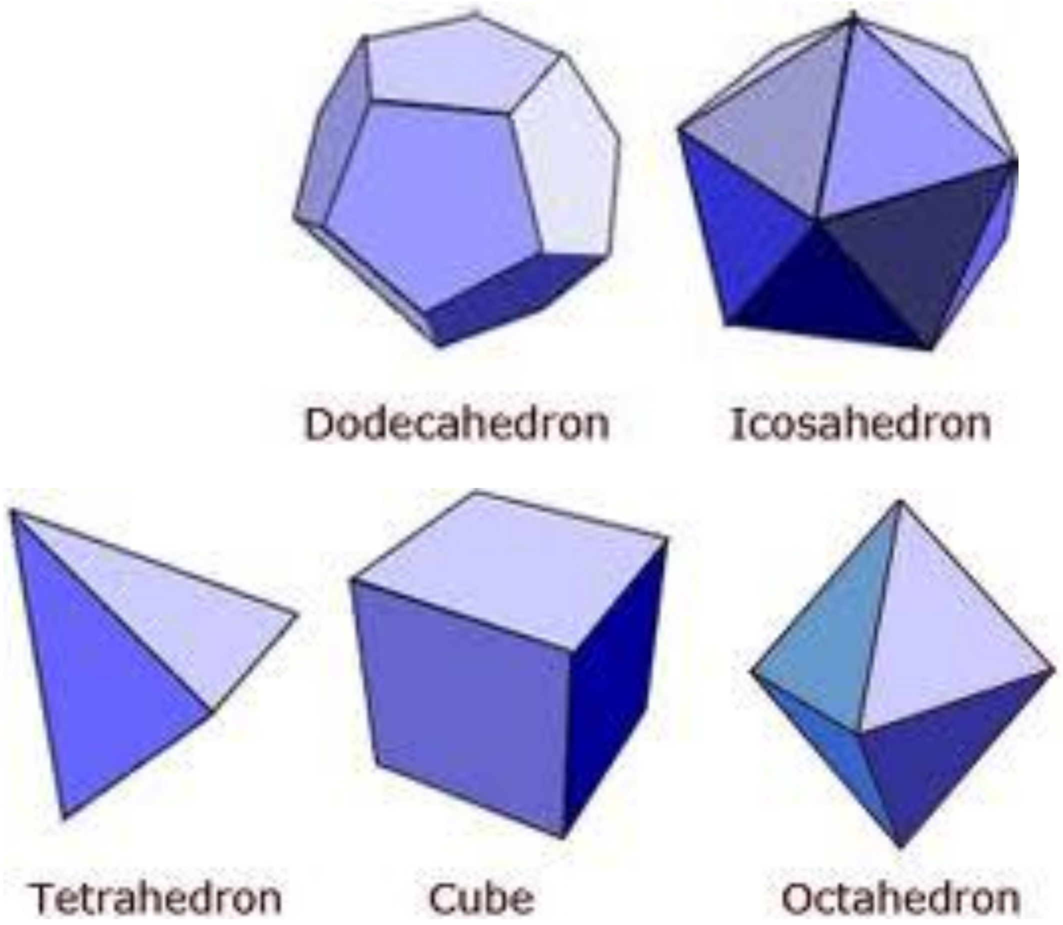
3D Polygonal Mesh

- The power of polygonal meshes

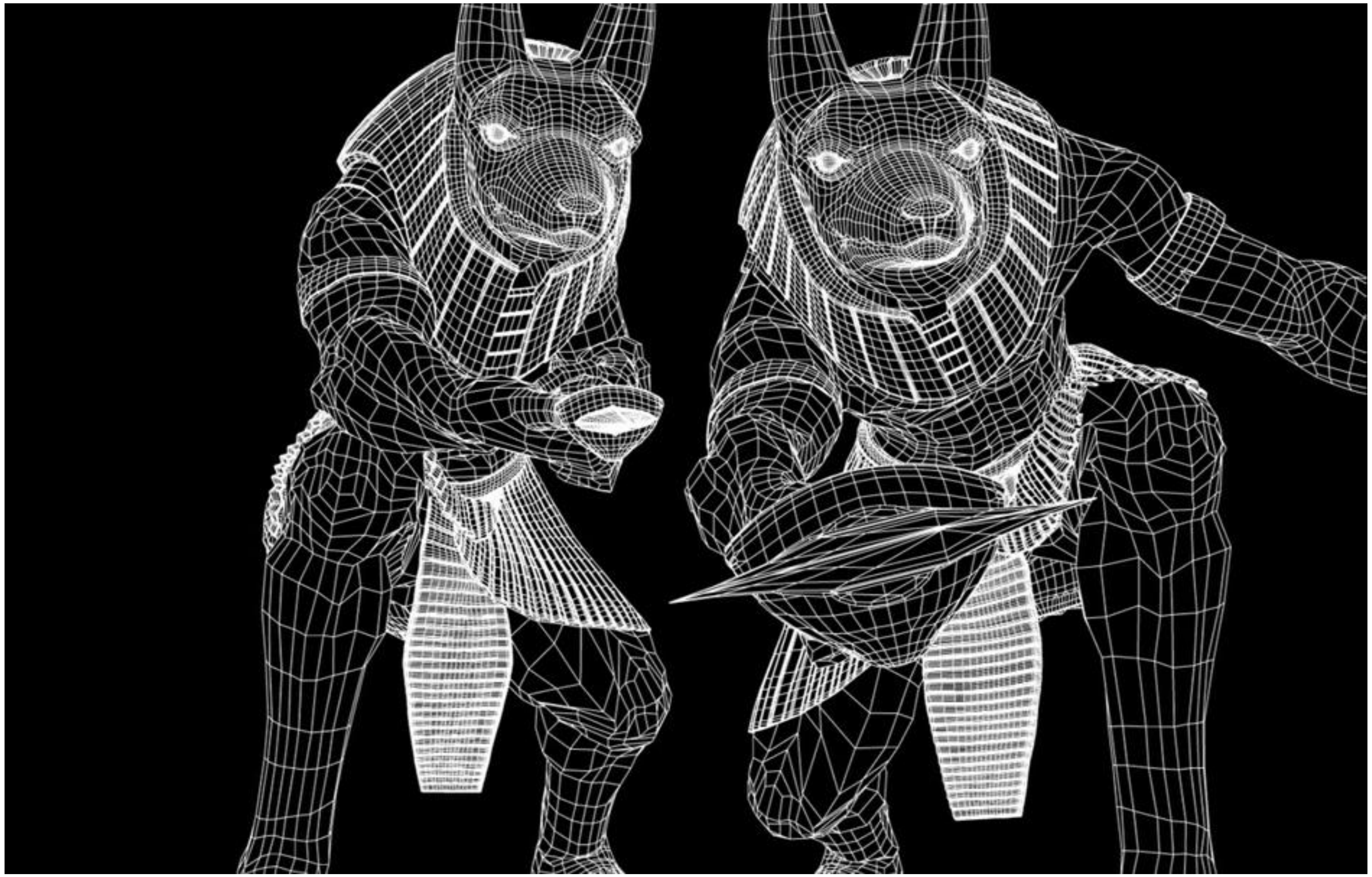
3D Polygonal Mesh

- Set of polygons representing a 2D surface embedded in 3D

Platonic Solids

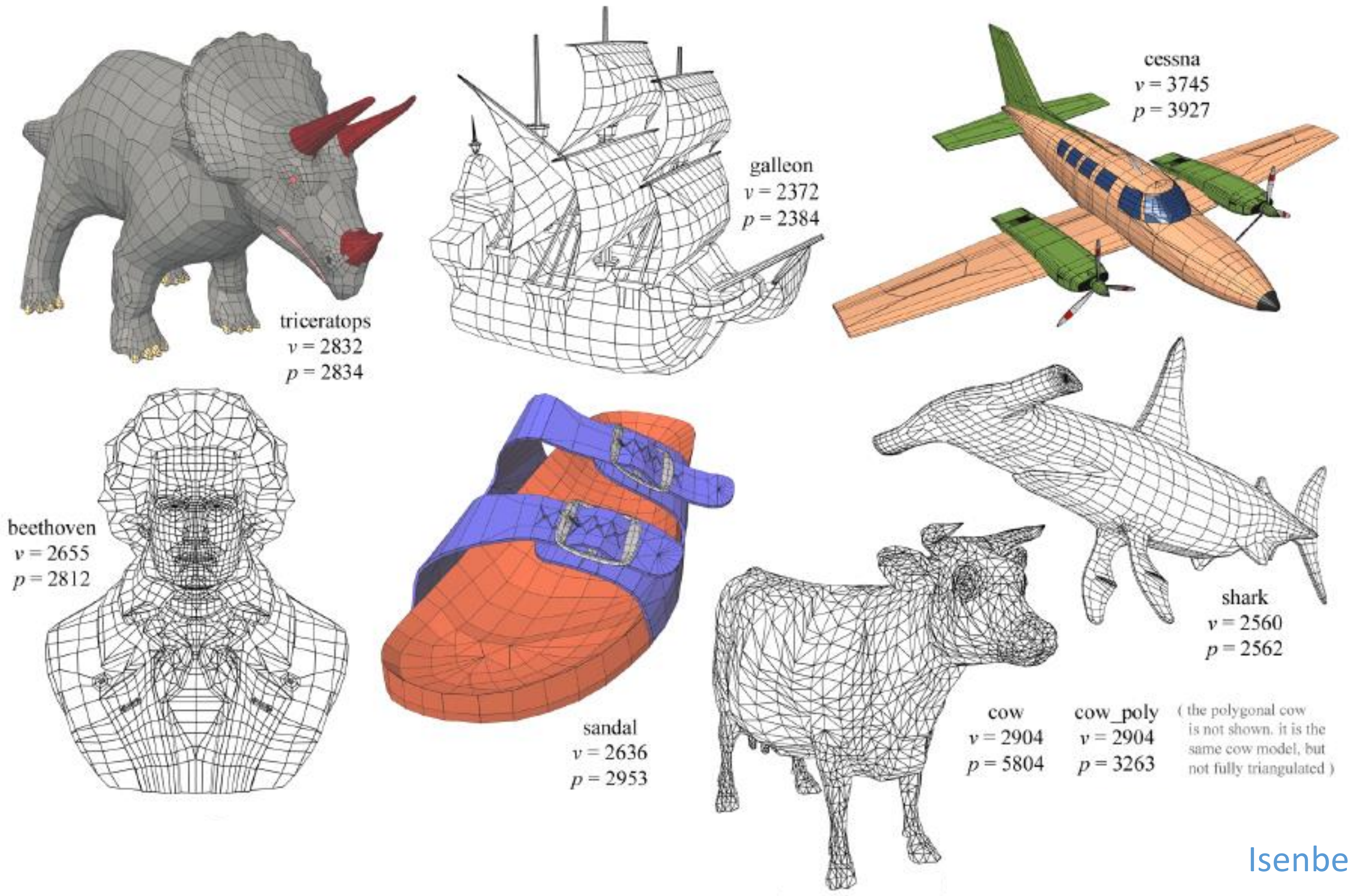


3D Polygonal Mesh



http://www.fxguide.com/featured/Comic_Horrors_Rocks_Statues_and_VanDyke/

3D Polygonal Mesh



Isenberg

3D Polygonal Meshes

- Why are they of interest?
  - Simple, common representation
  - Rendering with hardware support
  - Output of many acquisition tools

Outline

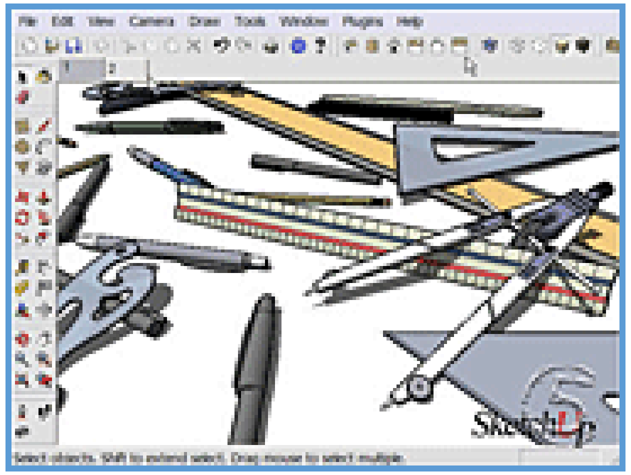
- Acquisition ←
- Representation
- Processing

Polygonal Mesh Acquisition

- Interactive modeling
- Scanners
- Procedural generation
- Conversion
- Simulations

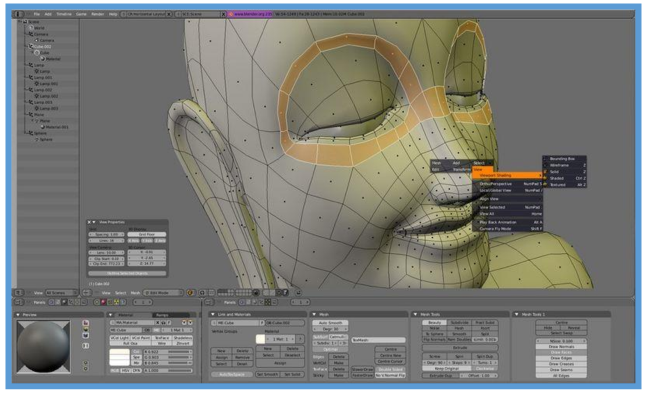
Polygonal Mesh Acquisition

- Interactive modeling
- Scanners
- Procedural generation
- Conversion
- Simulations



Select objects. Shift to extend select. Drag mouse to select multiple.

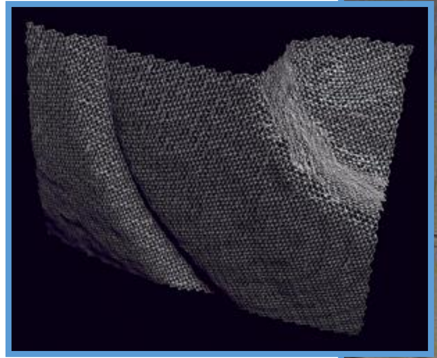
Sketchup

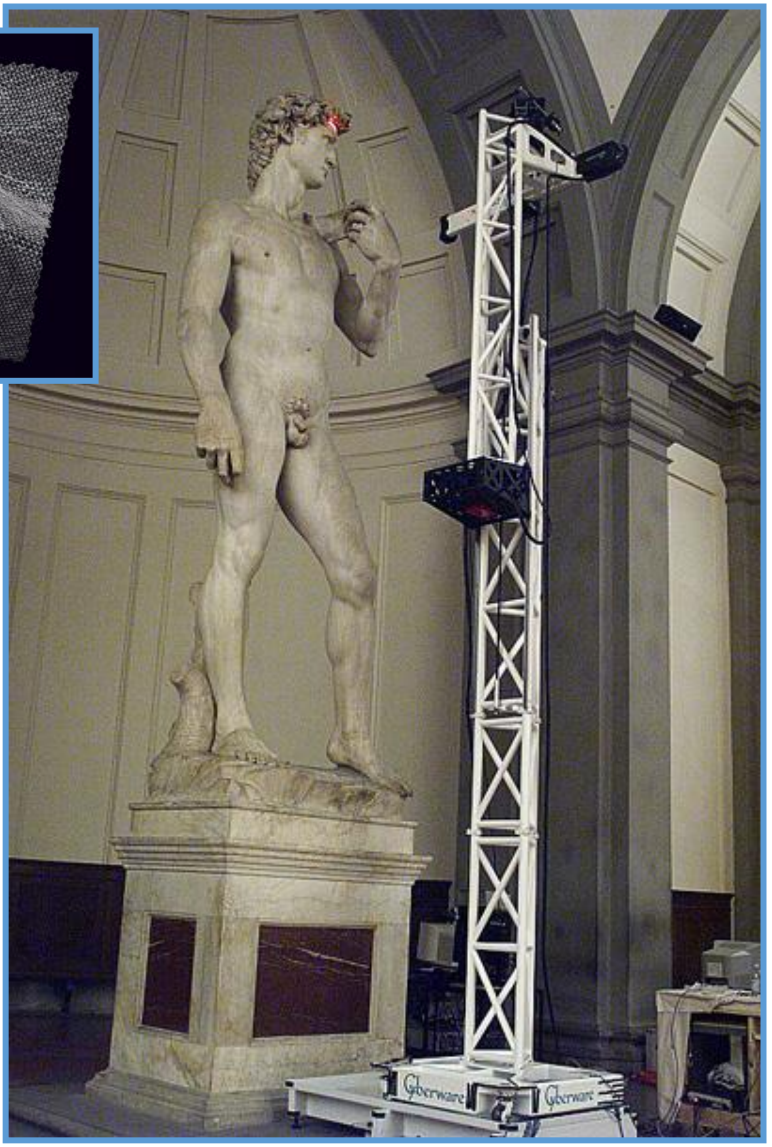


Blender

Polygonal Mesh Acquisition

- Interactive modeling
- Scanners
- Procedural generation
- Conversion
- Simulations

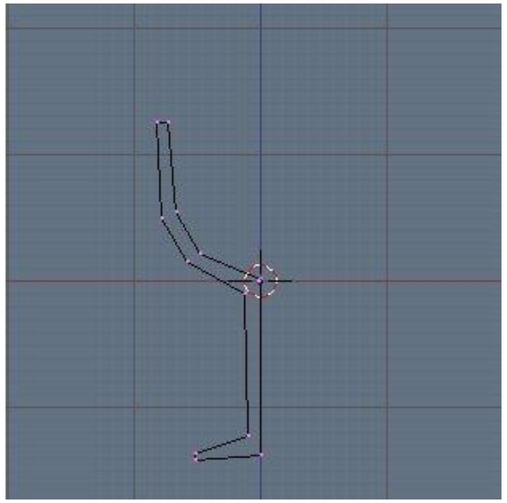


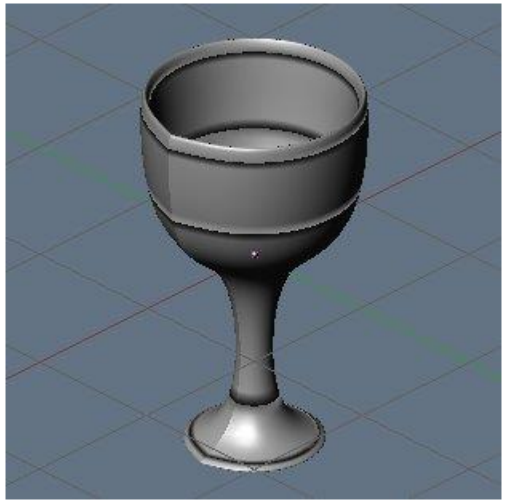


Digital Michelangelo Project
Stanford

Polygonal Mesh Acquisition

- Interactive modeling
- Scanners
- Procedural generation
- Conversion
- Simulations





sphynx.co.uk

Polygonal Mesh Acquisition

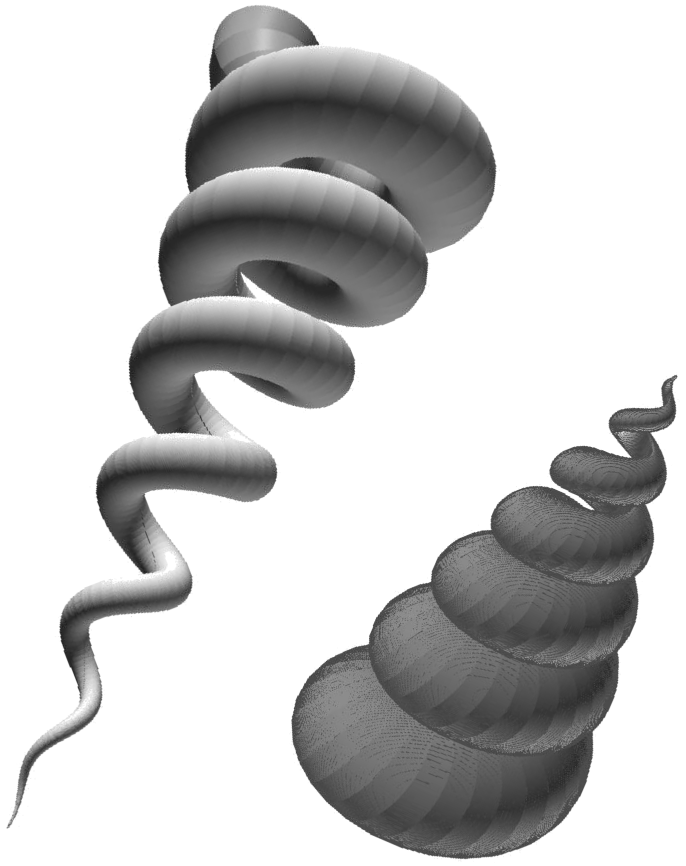
- Interactive modeling
- Scanners
- Procedural generation
- Conversion
- Simulations

Nicky Robinson, COS 426, 2014

Polygonal Mesh Acquisition

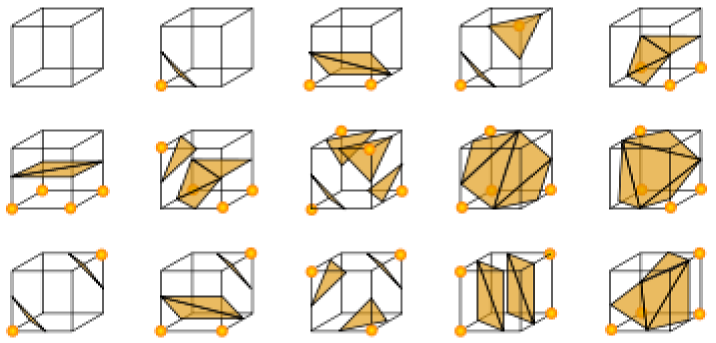
- Interactive modeling
- Scanners
- Procedural generation
- Conversion
- Simulations

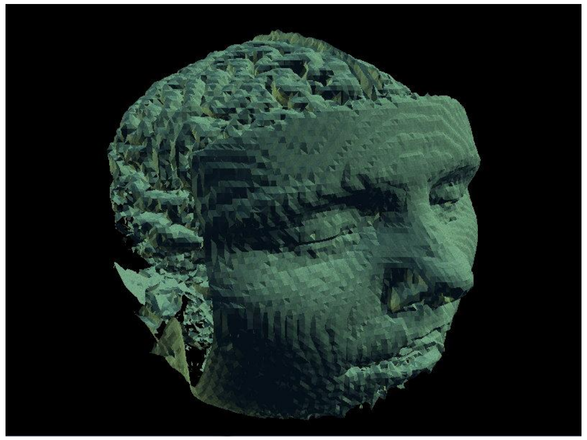
Fowler et al., 1992


Peter Maag, COS 426, 2010

Polygonal Mesh Acquisition

- Interactive modeling
- Scanners
- Procedural generation
- Conversion
- Simulations



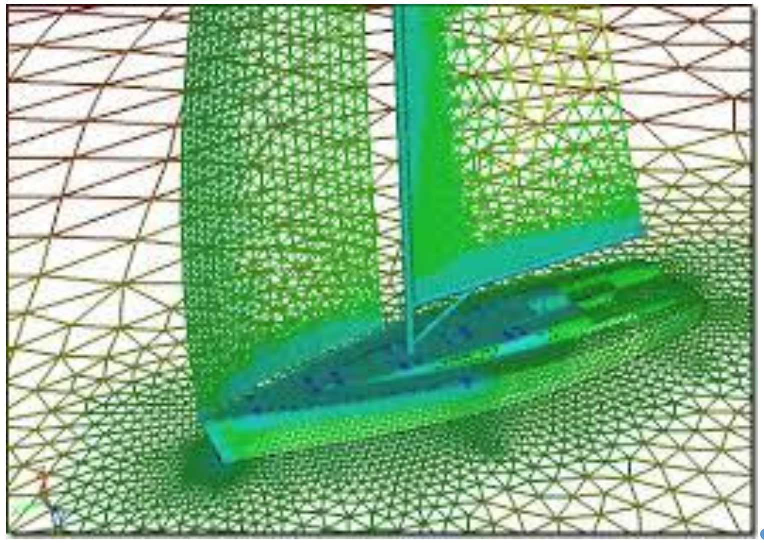


Marching cubes

Jose Maria De Espona

Polygonal Mesh Acquisition

- Interactive modeling
- Scanners
- Procedural generation
- Conversion
- Simulations


symscape

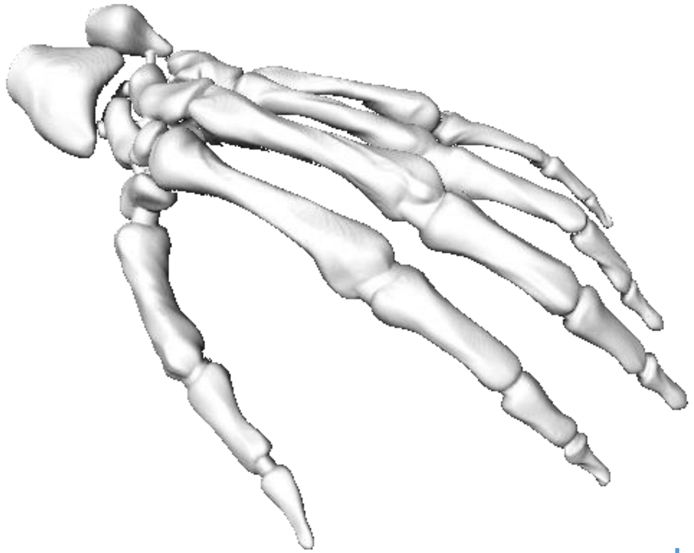
Lee et. al 2010

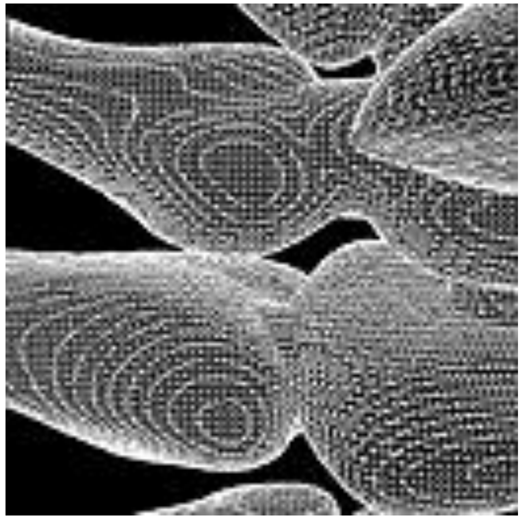
Outline

- Acquisition
- Representation ←
- Processing

Polygon Mesh Representation

- Important properties of mesh representation?
  - Efficient traversal of topology
  - Efficient use of memory
  - Efficient updates

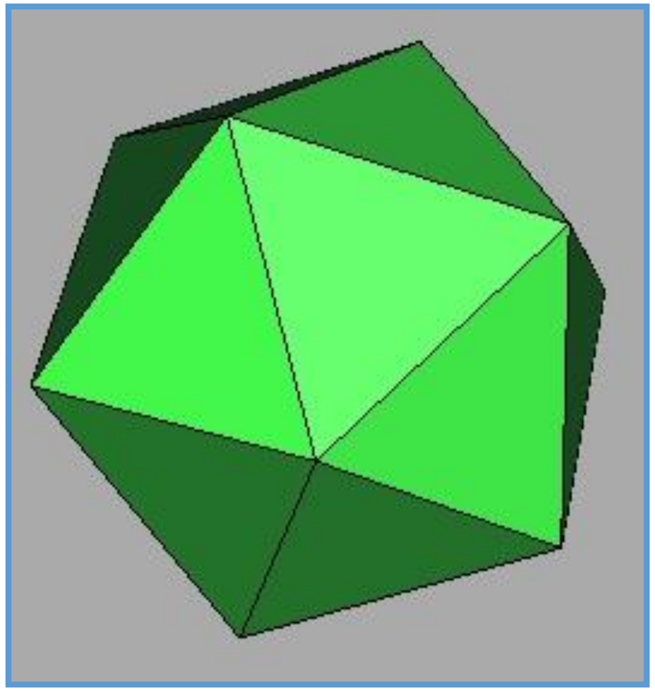




Large Geometric Model Repository
Georgia Tech

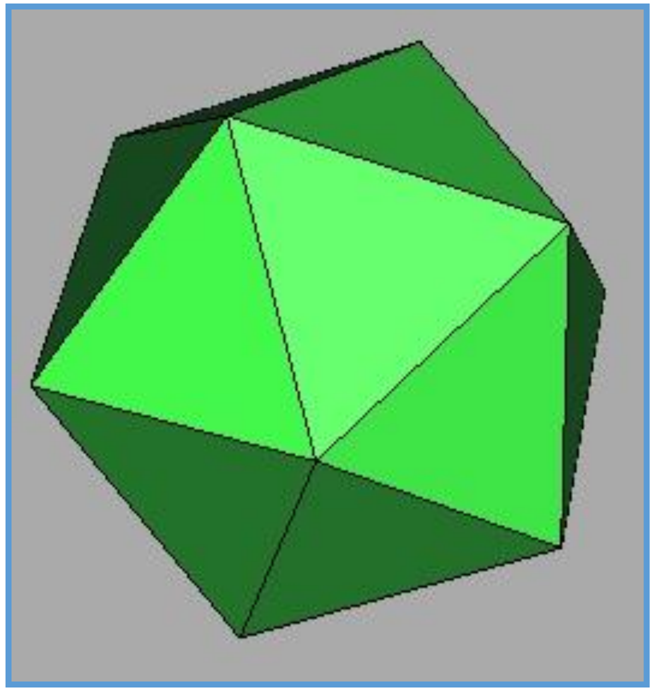
Polygon Mesh Representation

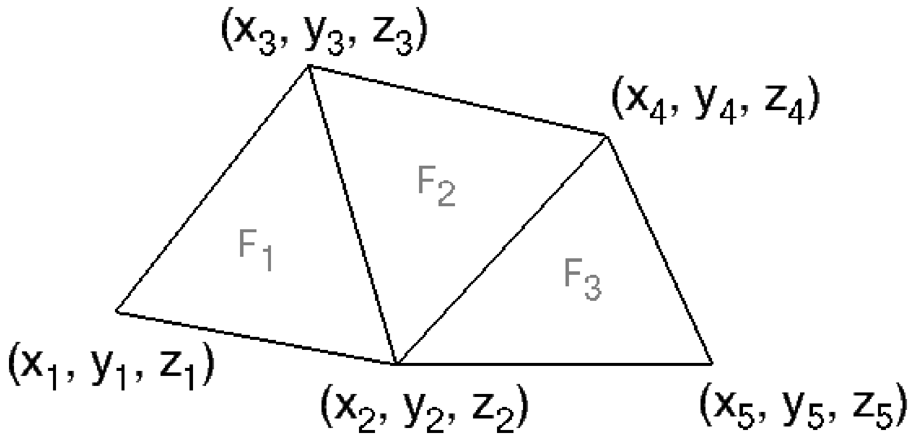
- Possible data structures



Independent Faces

- Each face lists vertex coordinates
  - Redundant vertices
  - No adjacency information



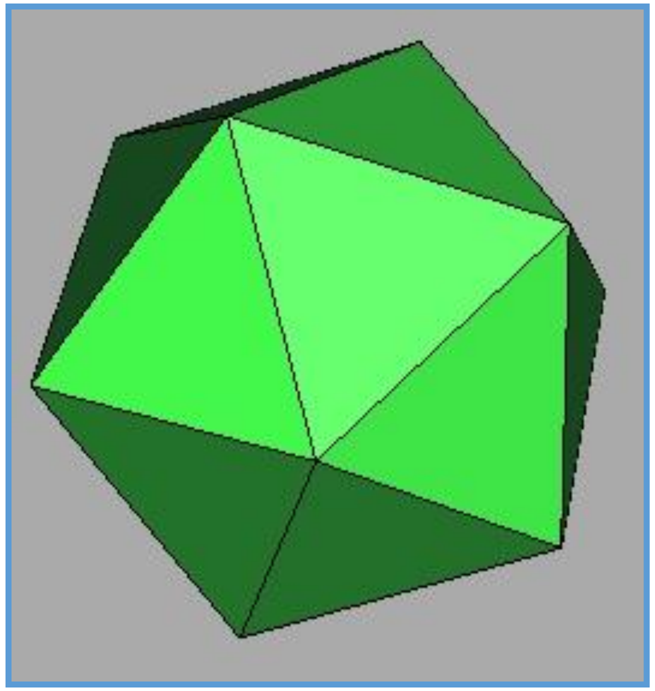


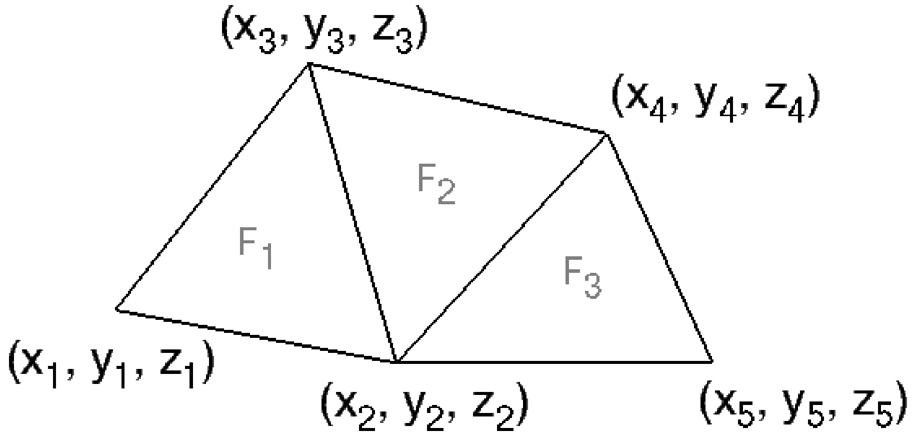
FACE TABLE

F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables (Indexed Vertices)

- Each face lists vertex references
  - Shared vertices
  - Still no adjacency information





VERTEX TABLE

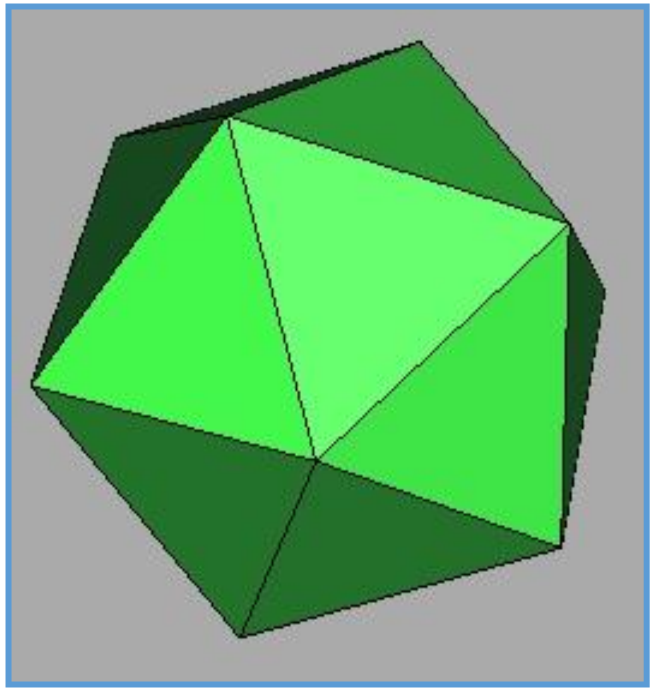
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

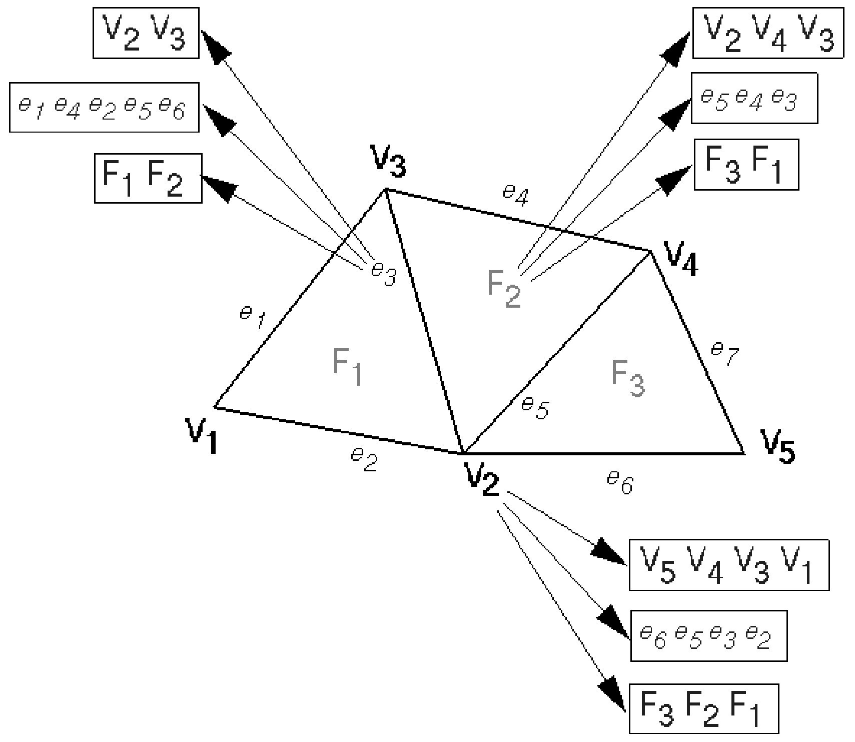
FACE TABLE

F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

Full Adjacency Lists

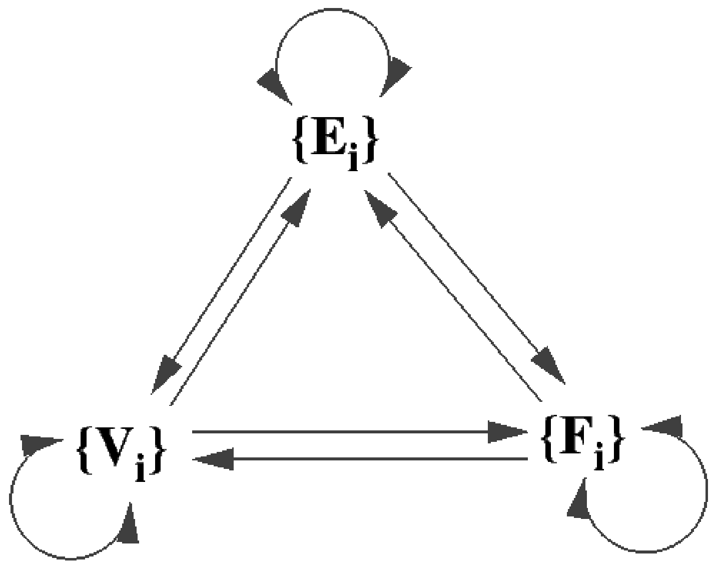
- Store all vertex, edge, and face adjacencies
  - ***Fast direct*** adjacency traversal
  - Extra storage

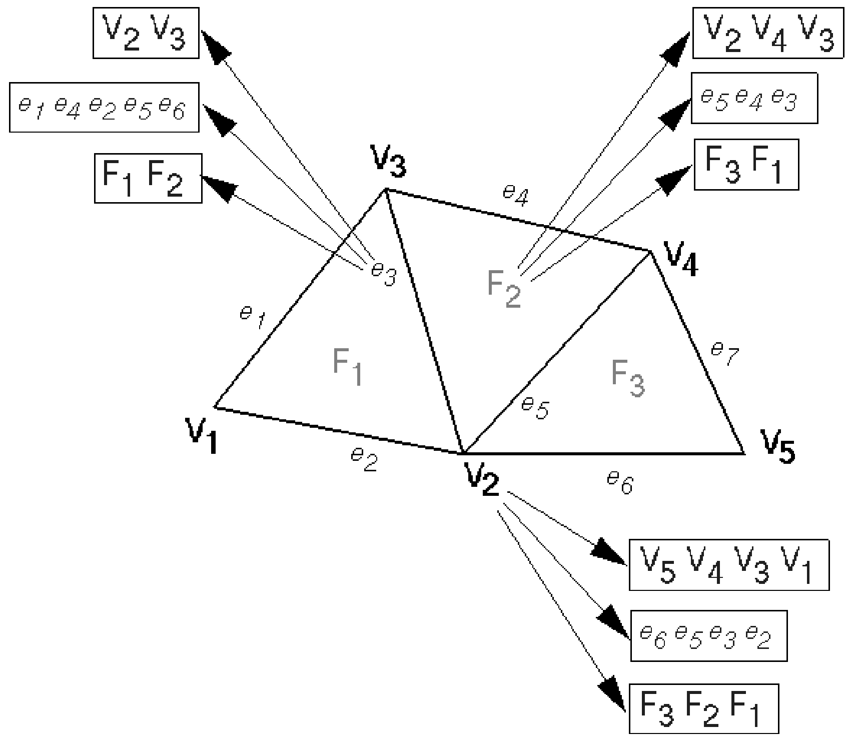




Full Adjacency Lists

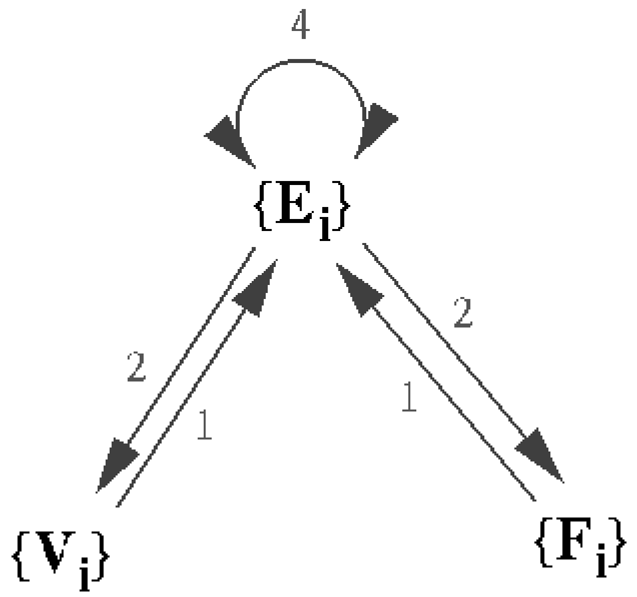
Adjacency relationships visualized:

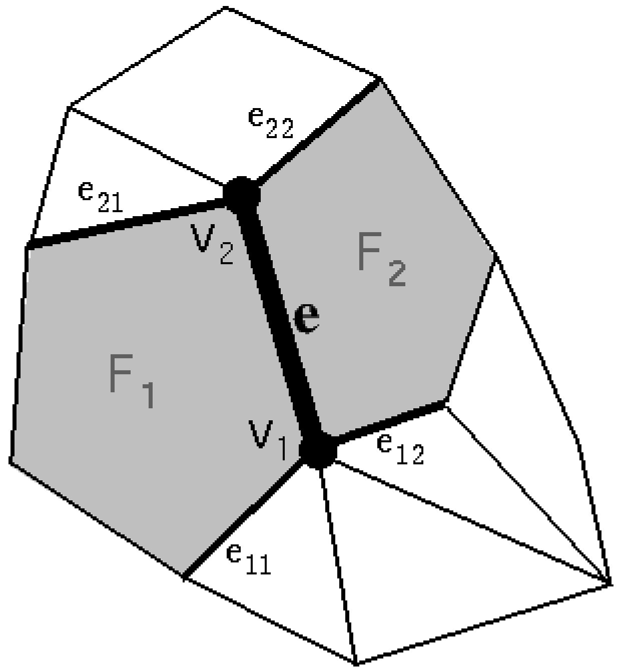


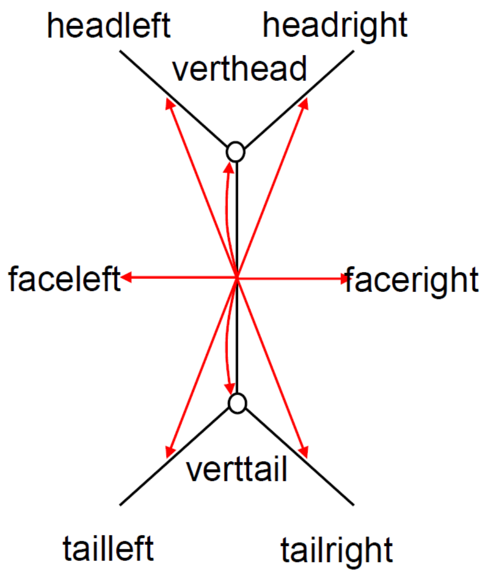


Partial Adjacency - Winged Edge

- Adjacency encoded in **edges**
  - All adjacencies in O(1) time
  - Little extra storage (fixed records)
  - Arbitrary polygons

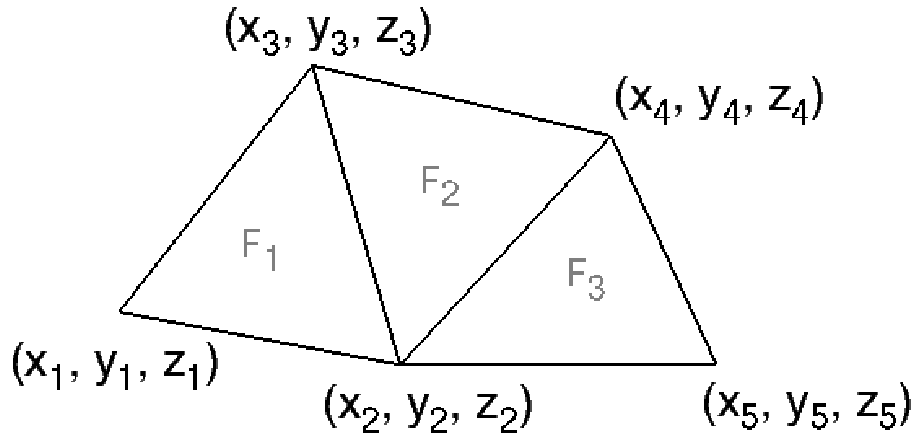






Winged Edge

- Example:



VERTEX TABLE

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE

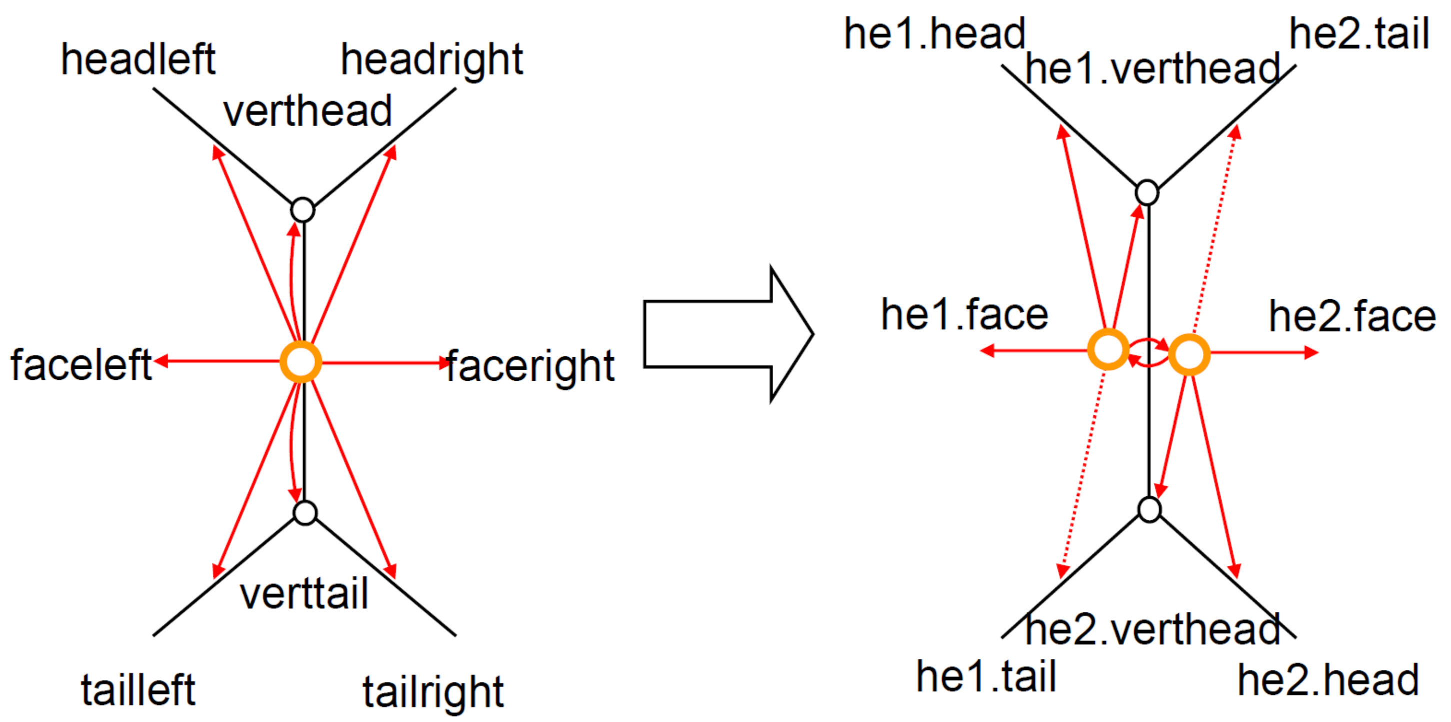
					11	12	21	22
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE

F_1	e_1
F_2	e_3
F_3	e_5

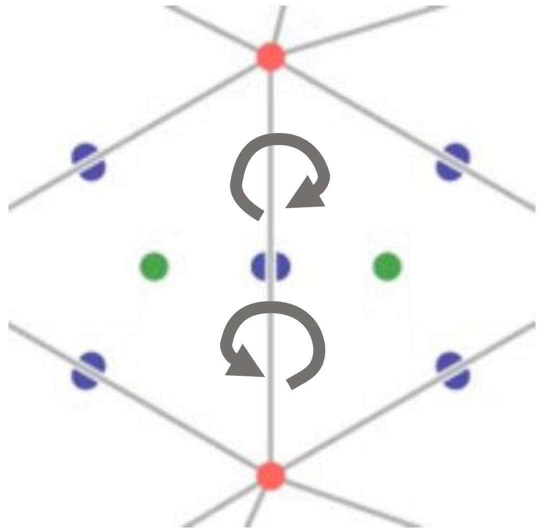
Half Edge

- traversals do not require “ifs” in code
- consistent orientation



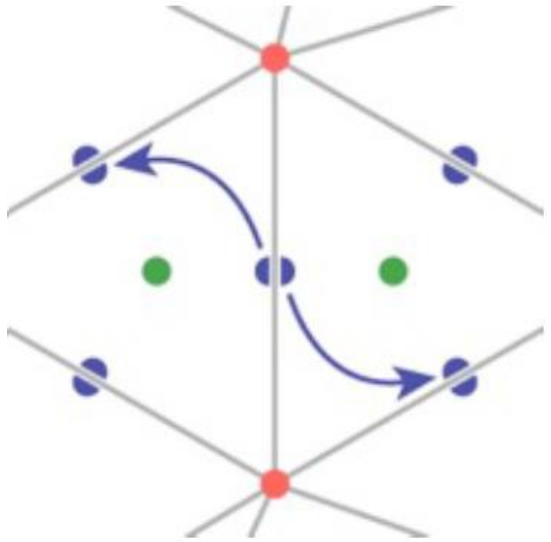
Half Edge … in more detail

- Each **half-**edge stores:
  - Its twin half-edge



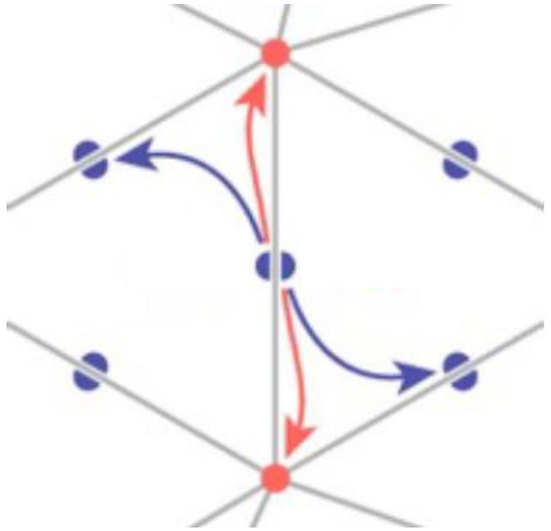
Half Edge

- Each **half-**edge stores:
  - Its twin half-edge
  - The next half-edge



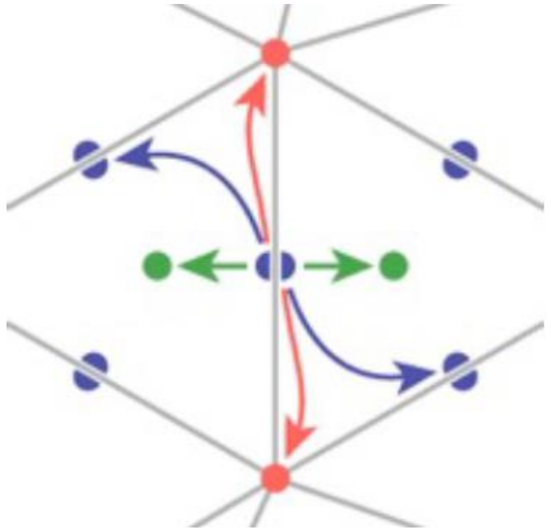
Half Edge

- Each **half-**edge stores:
  - Its twin half-edge
  - The next half-edge
  - The next vertex



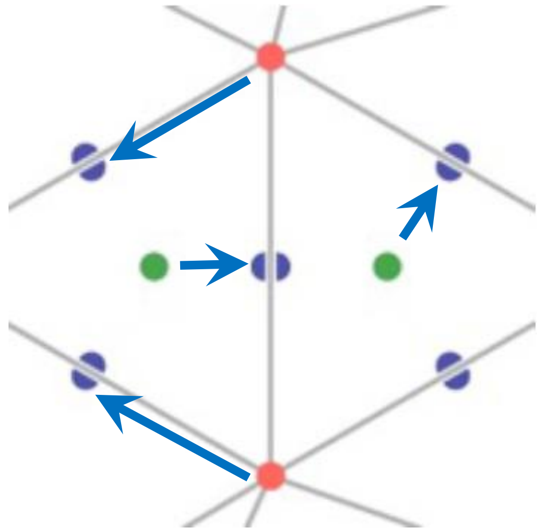
Half Edge

- Each **half-**edge stores:
  - Its twin half-edge
  - The next half-edge
  - The next vertex
  - The incident face



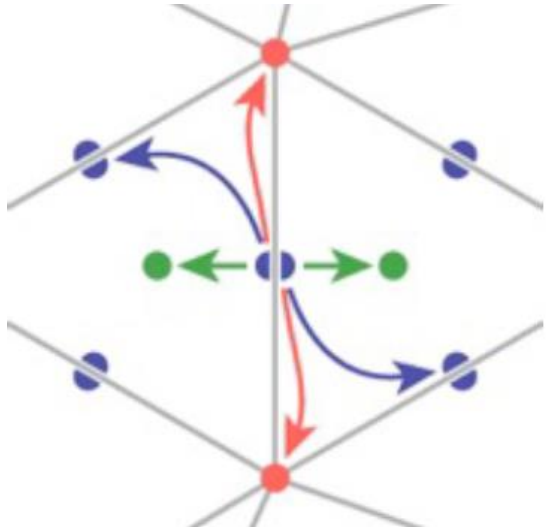
Half Edge

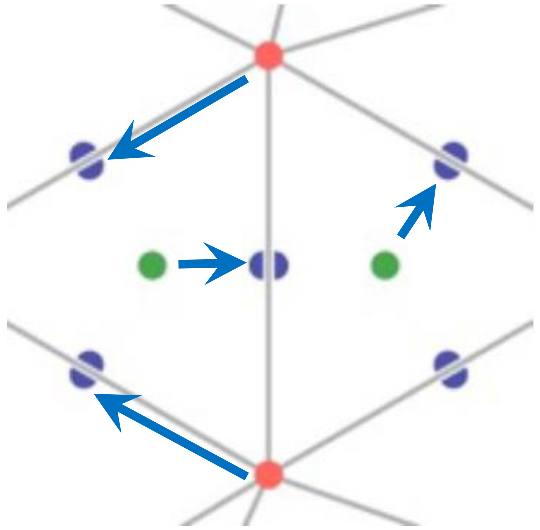
- Each **half-**edge stores:
  - Its twin half-edge
  - The next half-edge
  - The next vertex
  - The incident face
- Each face stores:
  - ***1*** adjacent half-edge
- Each vertex stores:
  - 1 outgoing half-edge

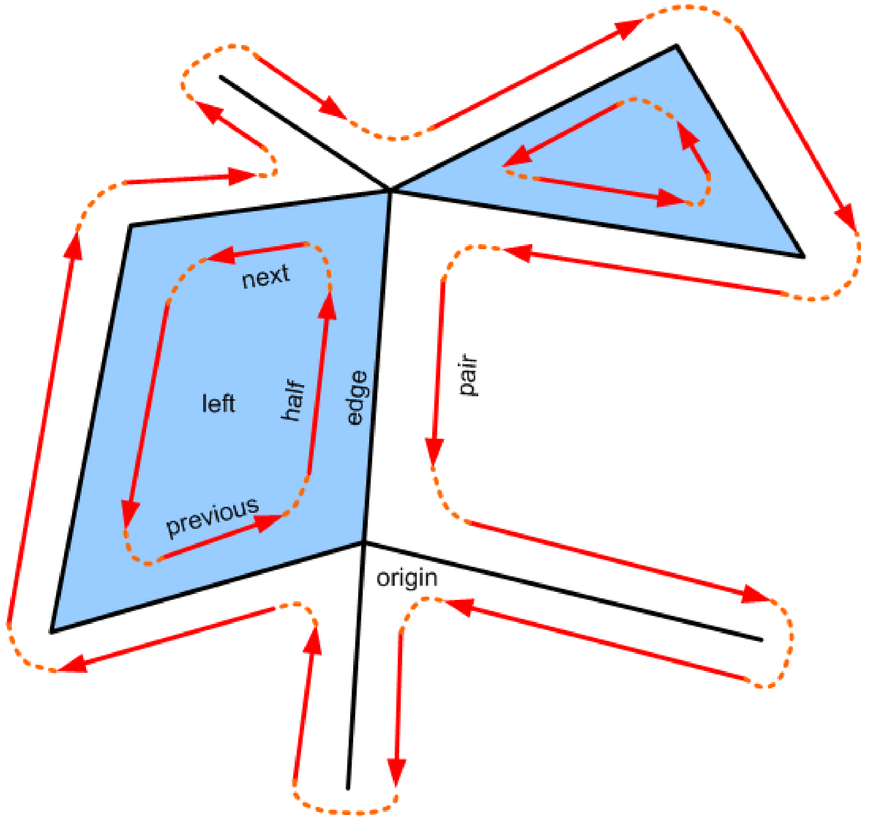


Half Edge

- Queries. How do you find:
  - All faces incident to an edge?
  - All vertices of a face?
  - All faces incident to a face?
  - All vertices incident to a vertex?

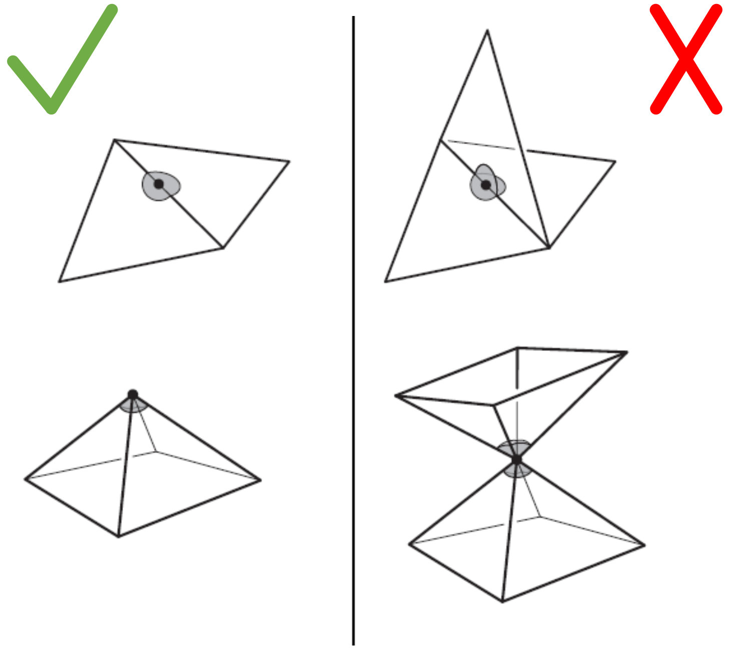


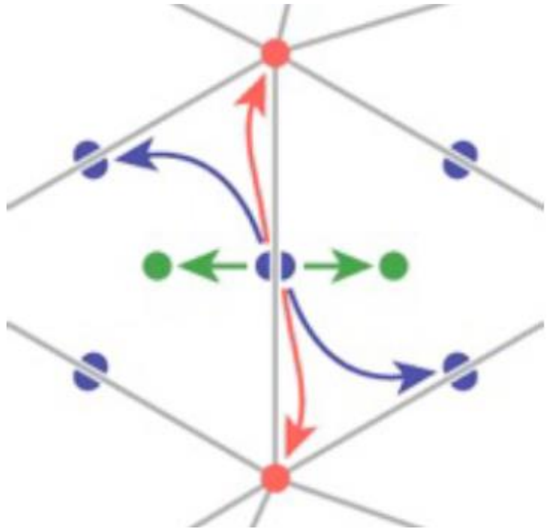




Half Edge

- Adjacency encoded in edges
  - All adjacencies in O(1) time
  - Little extra storage (fixed records)
  - Arbitrary polygons
  - **Assumes 2-Manifold surfaces**





Outline

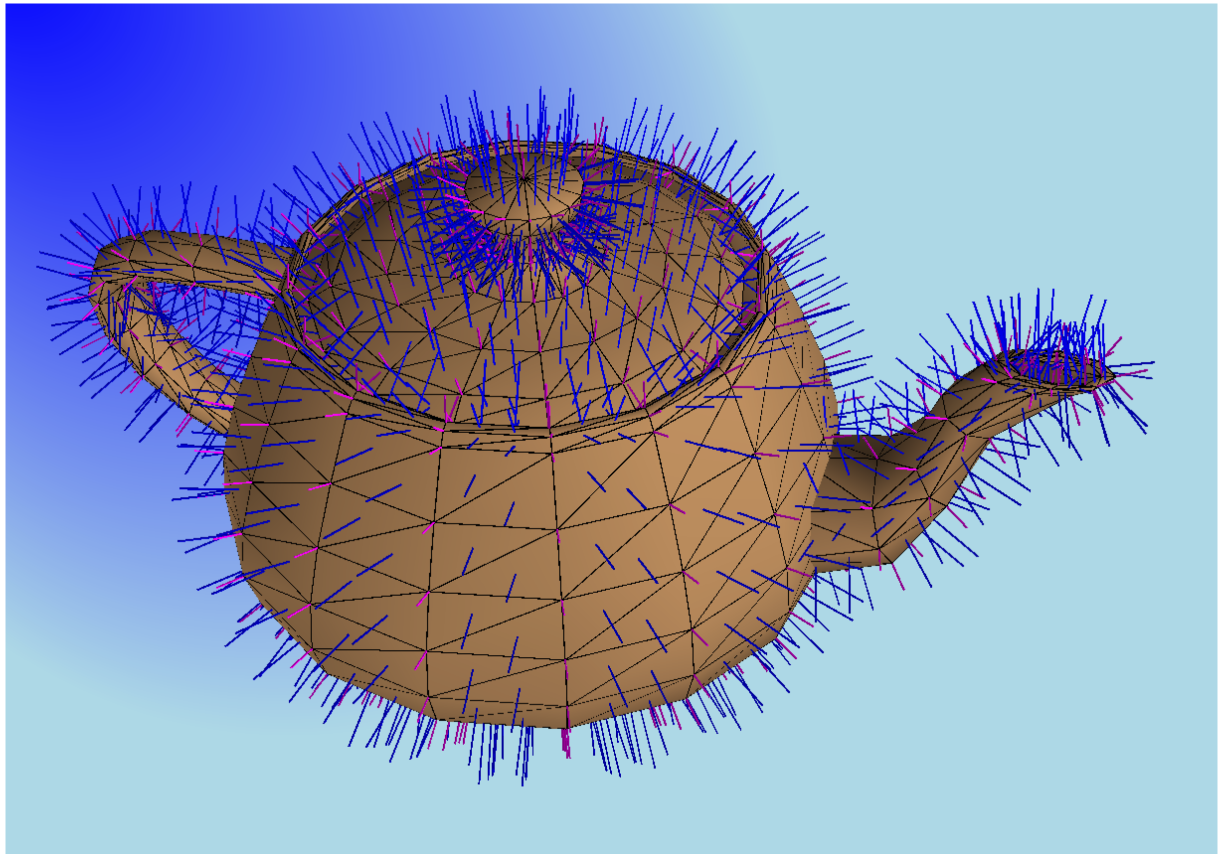
- Acquisition
- Representation
- Processing

Polygonal Mesh Processing

- Analysis
  - Normals
  - Curvature
- Warps
  - Rotate
  - Deform
- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel

Polygonal Mesh Processing

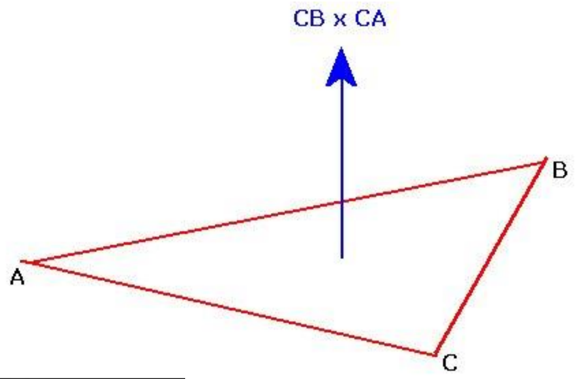
- Analysis
  - ➢Normals
  - Curvature
- Warps
  - Rotate
  - Deform
- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel

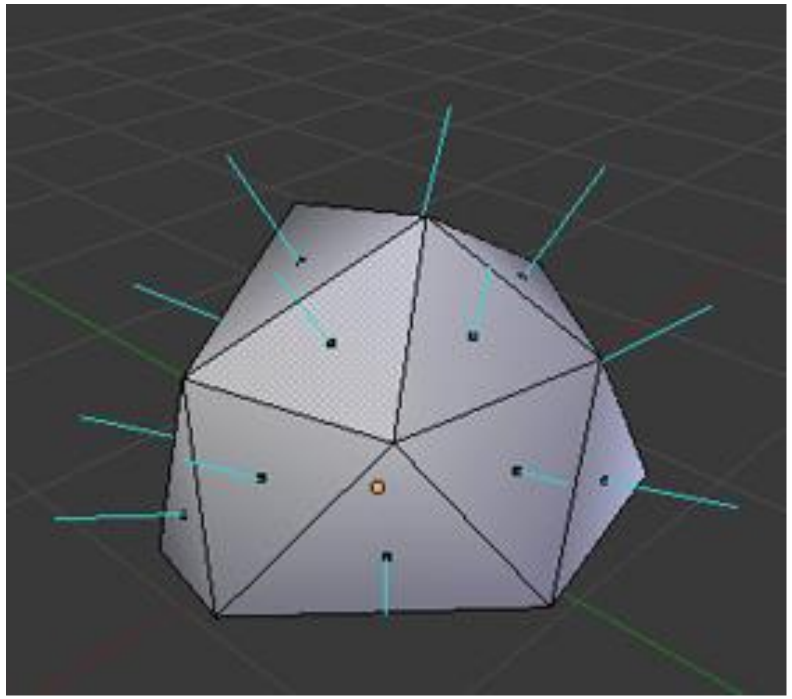


Polygonal Mesh Processing

- Analysis
  - ➢Normals
  - Curvature
- Warps
  - Rotate
  - Deform
- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel

Face normals:
(use cross product)

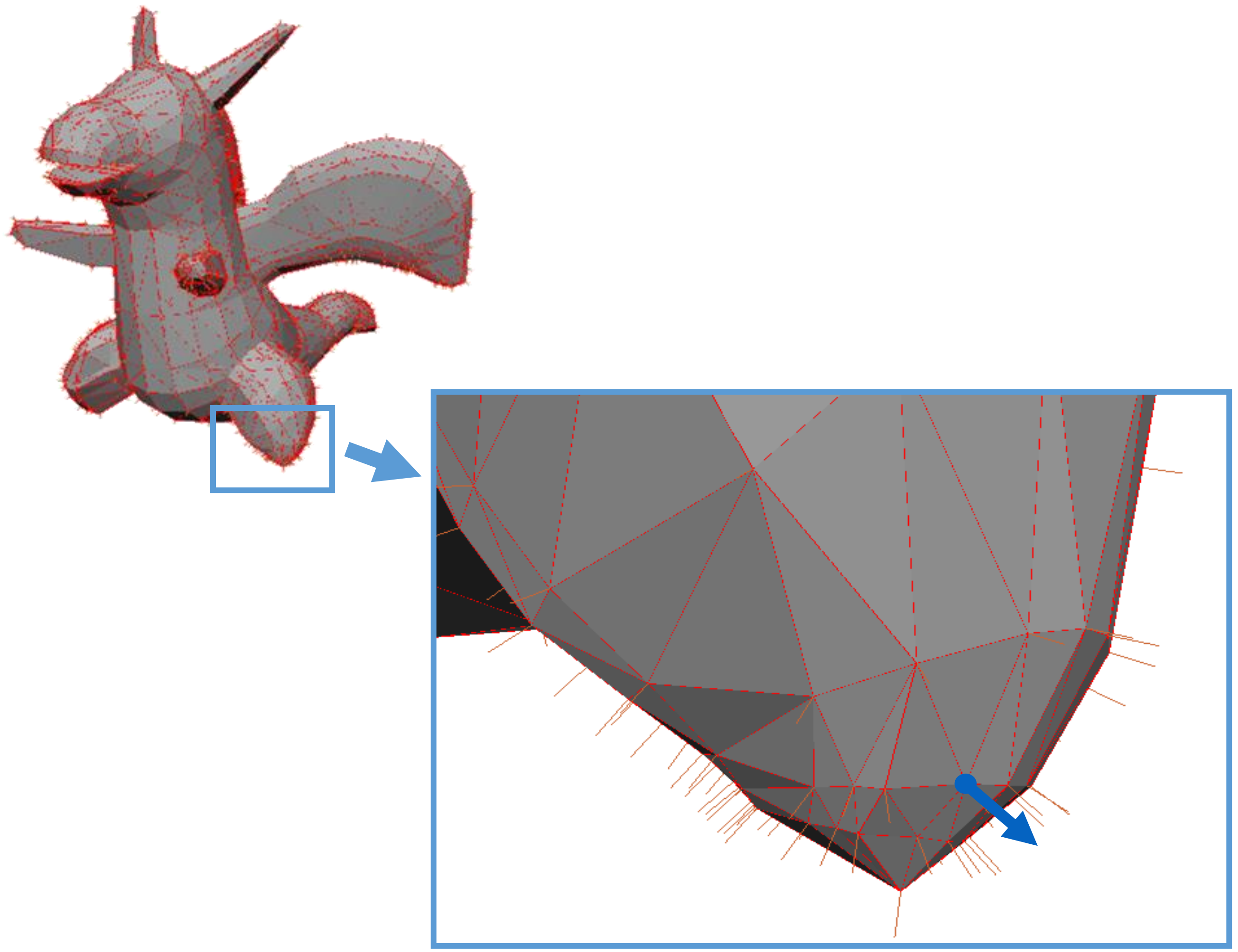




Polygonal Mesh Processing

- Analysis
    - ➢Normals
    - Curvature
- Warps
    - Rotate
    - Deform
- Filters
    - Smooth
    - Sharpen
    - Truncate
    - Bevel

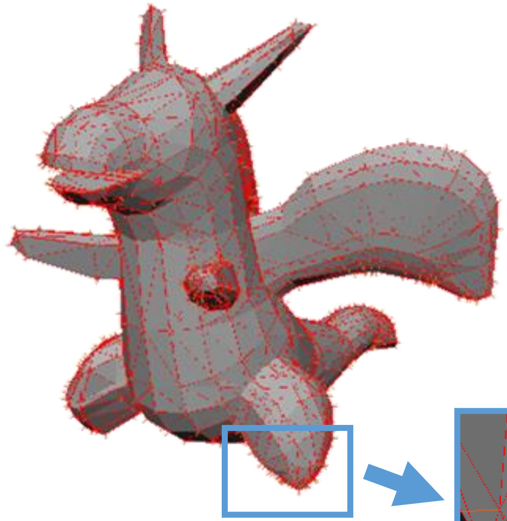
Vertex normals:

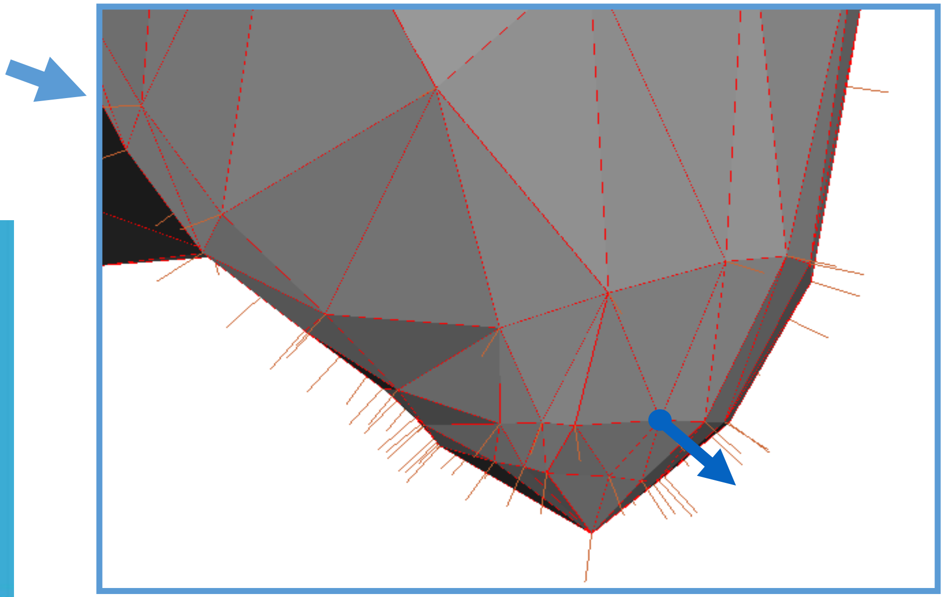


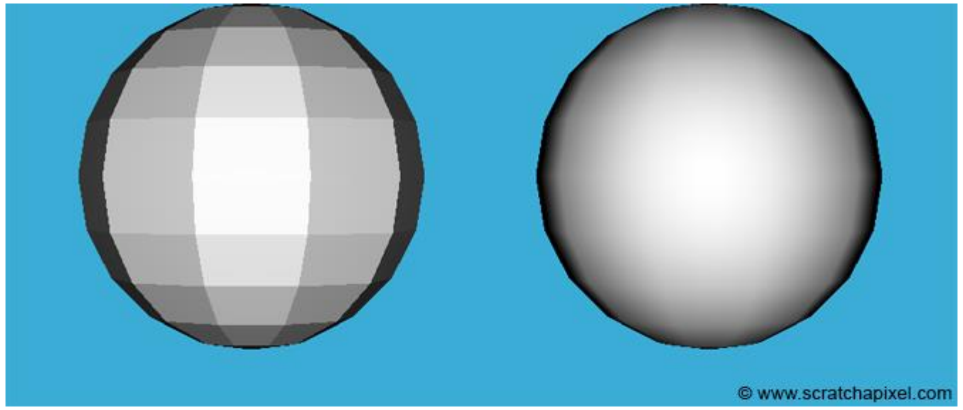
Polygonal Mesh Processing

- Analysis
  - ➤Normals
  - Curvature
- Warps
  - Rotate
  - Deform
- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel

Vertex normals:



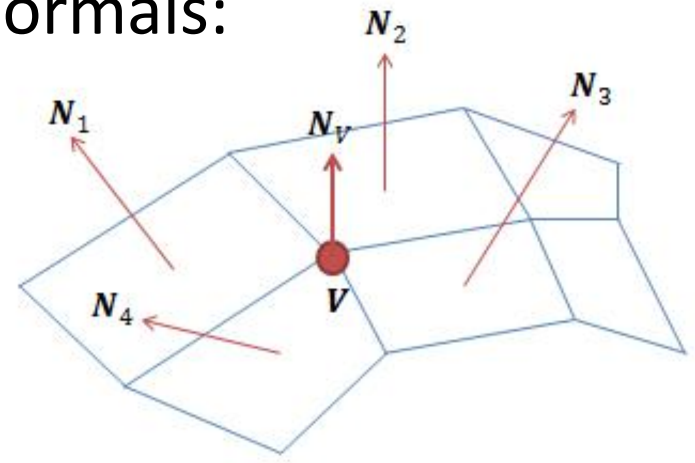




Polygonal Mesh Processing

- Analysis
  - ➢Normals
  - Curvature
- Warps
  - Rotate
  - Deform
- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel

Vertex Normals:

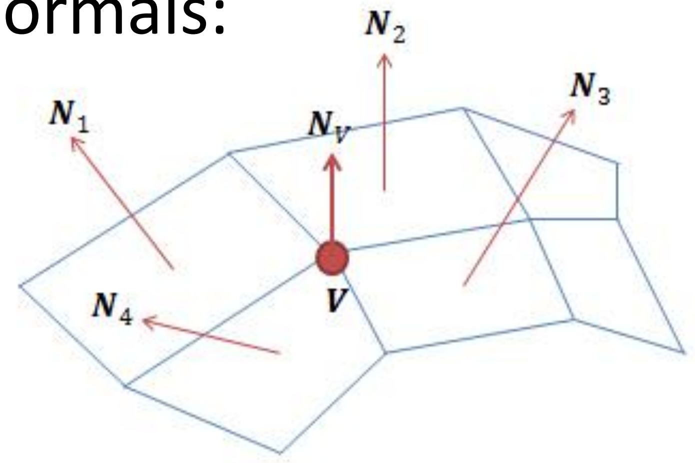


```
for each face
•   calculate face normal
•   add normal to each connected vertex normal
```

Polygonal Mesh Processing

- Analysis
  - ➢Normals
  - Curvature
- Warps
  - Rotate
  - Deform
- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel

Vertex Normals:



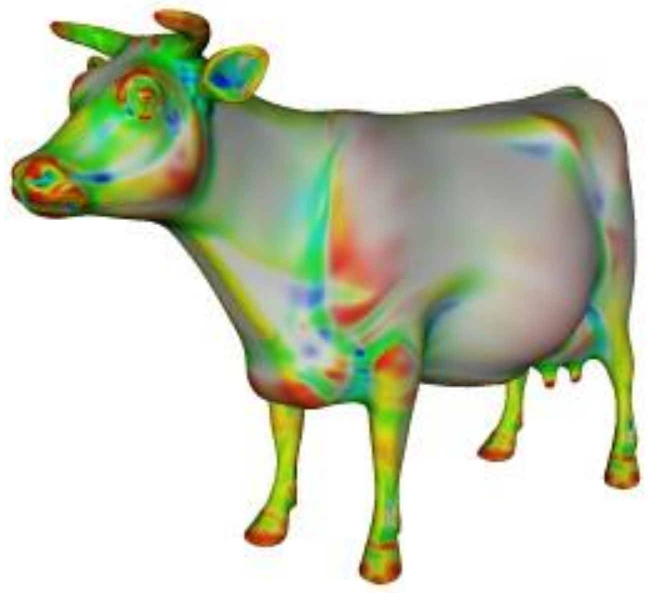
$$N_V = \frac{\sum_{k=1}^{n} N_k}{\left|\sum_{k=1}^{n} N_k\right|}$$

```
for each face
•   calculate face normal
•   add normal to each connected vertex normal


for each vertex normal
•   normalize
```

Polygonal Mesh Processing

- Analysis
  - Normals
  - ➢Curvature
- Warps
  - Rotate
  - Deform
- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel



color-coded curvature
(red → higher curvature)

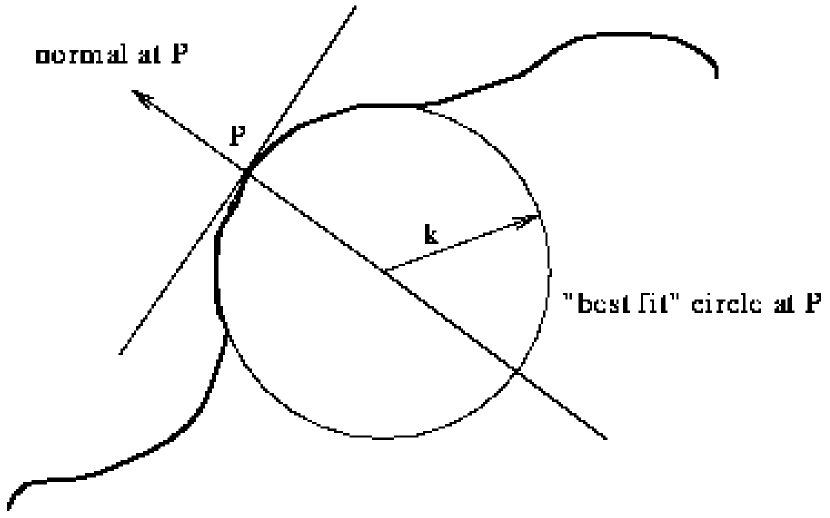
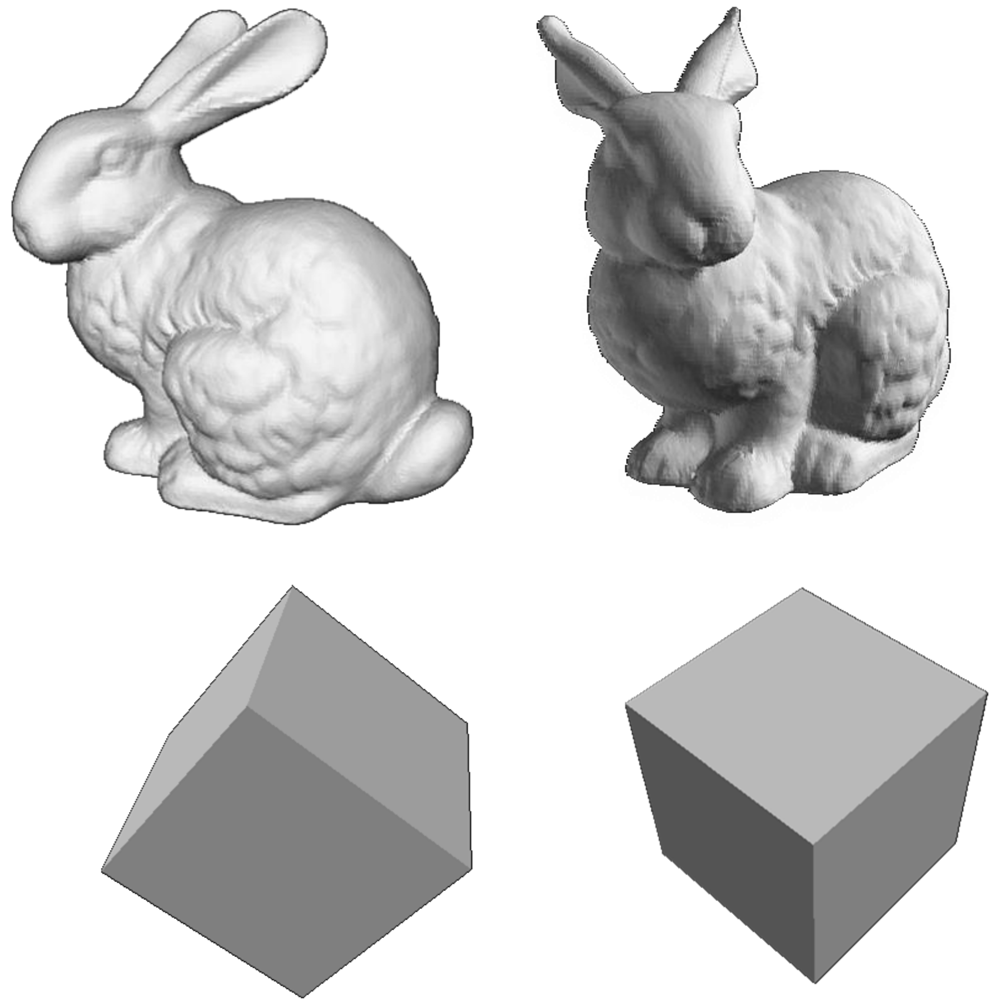


Figure 32: curvature of curve at P is $1/k$

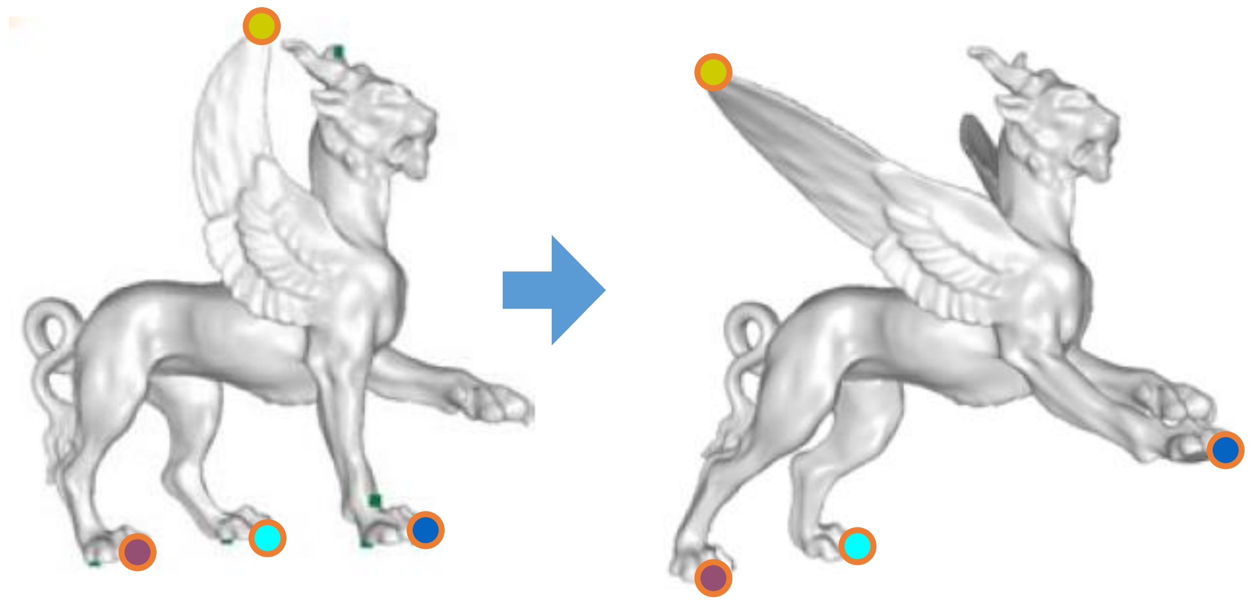
Polygonal Mesh Processing

- Analysis
  - Normals
  - Curvature
- Warps
  - ➤Rotate
  - Deform
- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel



Polygonal Mesh Processing

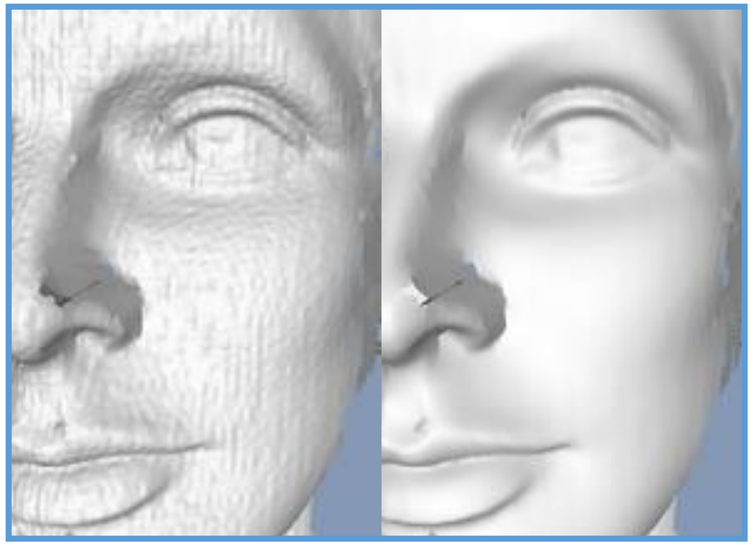
- Analysis
  - Normals
  - Curvature
- Warps
  - Rotate
  - ➢Deform
- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel



Sheffer

Polygonal Mesh Processing

- Analysis
  - Normals
  - Curvature
- Warps
  - Rotate
  - Deform
- Filters
  - ➢Smooth
  - Sharpen
  - Truncate
  - Bevel


Thouis "Ray" Jones

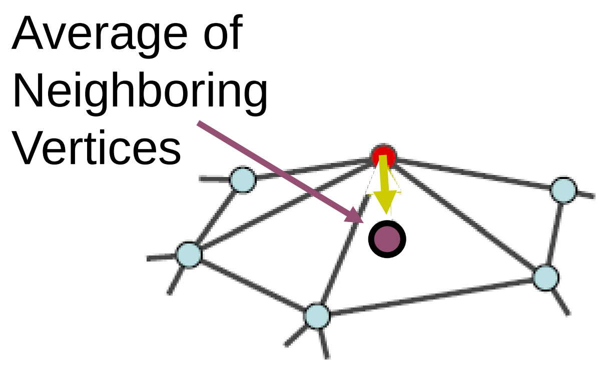
How?

The Laplacian Operator

- Mesh formulation:

$$\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} (\mathbf{v}_i - \mathbf{v}_j)$$

$d_i = |N(i)|$ is the number of neighbors.



Olga Sorkine

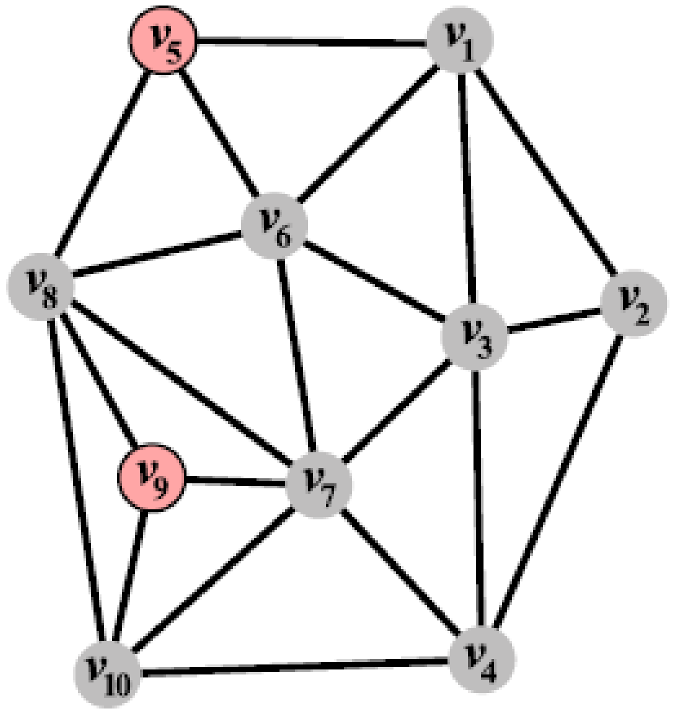
The Laplacian Operator

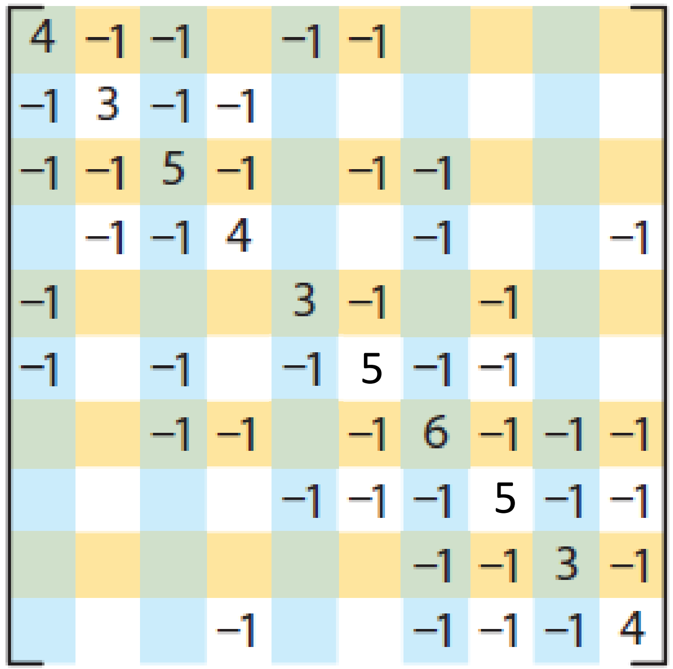
- The Laplacian operator Δ

$$L(v_i) = \Delta(v_i) = \frac{\sum_{j \in 1_{ring_i}} v_j - v_i}{\#1_{ring_i}}$$

- In matrix form:

$$L_{ij} = \begin{cases} -w_{ij} & i \neq j \\ \sum_{j \in 1_{ring_i}} w_{ij} & i = j \\ 0 & else \end{cases}$$





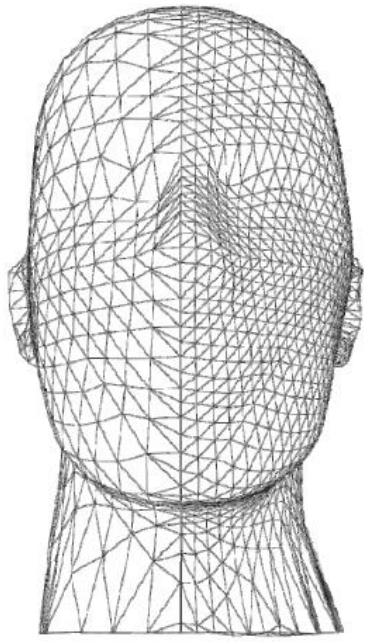
4	−1	−1		−1	−1				
−1	3	−1	−1						
−1	−1	5	−1		−1	−1			
	−1	−1	4			−1			−1
−1				3	−1		−1		
−1		−1		−1	5	−1	−1		
		−1	−1		−1	6	−1	−1	−1
				−1	−1	−1	5	−1	−1
						−1	−1	3	−1
			−1			−1	−1	−1	4

The Laplacian Operator

- The Laplacian operator Δ

$$L(v_i) = \Delta(v_i) = \frac{\Sigma_{j \in 1_{ring_i}} v_j - v_i}{\# 1_{ring_i}}$$

- However, Meshes are irregular



The Laplacian Operator

- The Laplacian operator Δ

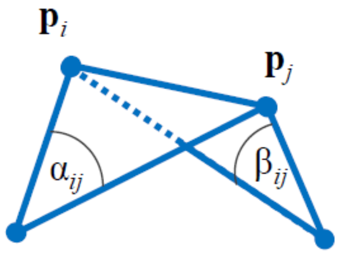
$$L(v_i) = \Delta(v_i) = \frac{\sum_{j \in 1_{ring_i}} v_j - v_i}{\#1_{ring_i}}$$

- However, Meshes are irregular

  - Cotangent weights:

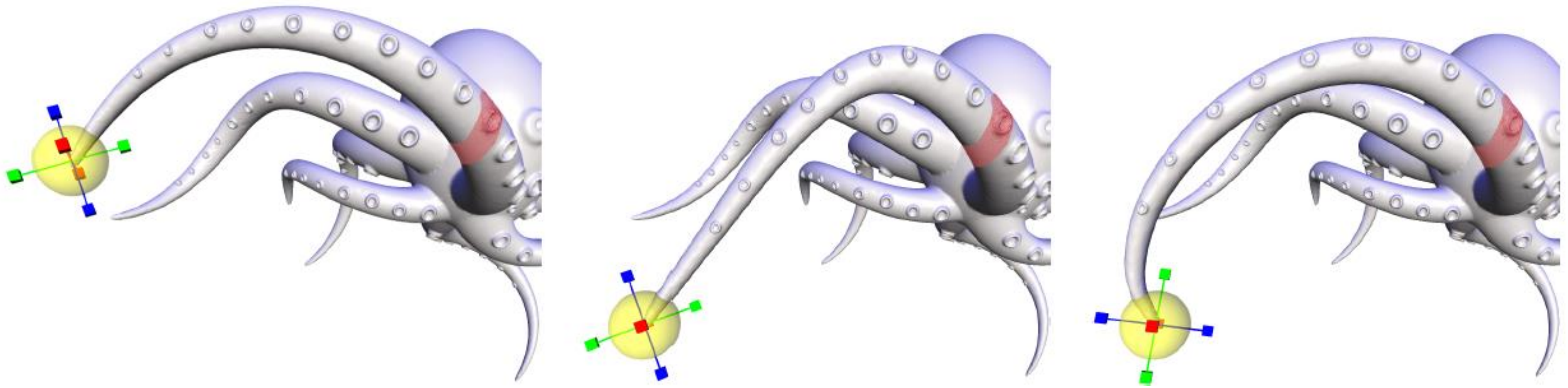
$$L(p_i) = \frac{\sum_{j \in 1_{ring_i}} w_{ij} \cdot p_j}{\sum_{j \in 1_{ring_i}} w_{ij}} - p_i$$

$$w_{ij} = \frac{\cot(\alpha_{ij}) + \cot(\beta_{ij})}{2}$$



Solve Constrained Laplacian Optimization

- Applicable to:
  - Deformation, by adding constraints



Solve Constrained Laplacian Optimization

- The Laplacian operator Δ

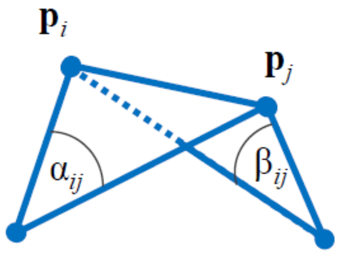
$$L(v_i) = \Delta(v_i) = \frac{\sum_{j \in 1ring_i} v_j - v_i}{\#1ring_i}$$

- However, Meshes are irregular

  - Cotangent weights:

$$L(p_i) = \frac{\sum_{j \in 1ring_i} w_{ij} \cdot p_j}{\sum_{j \in 1ring_i} w_{ij}} - p_i$$

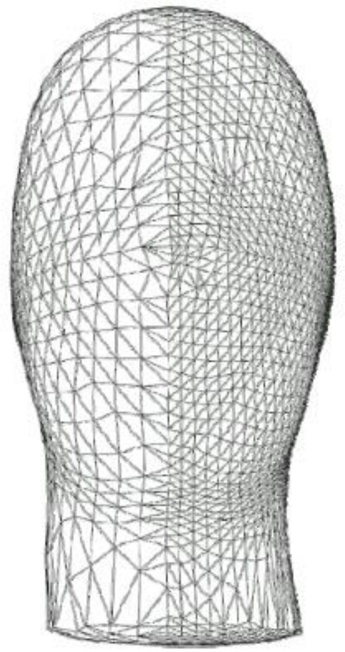
$$w_{ij} = \frac{\cot(\alpha_{ij}) + \cot(\beta_{ij})}{2}$$



Solve:

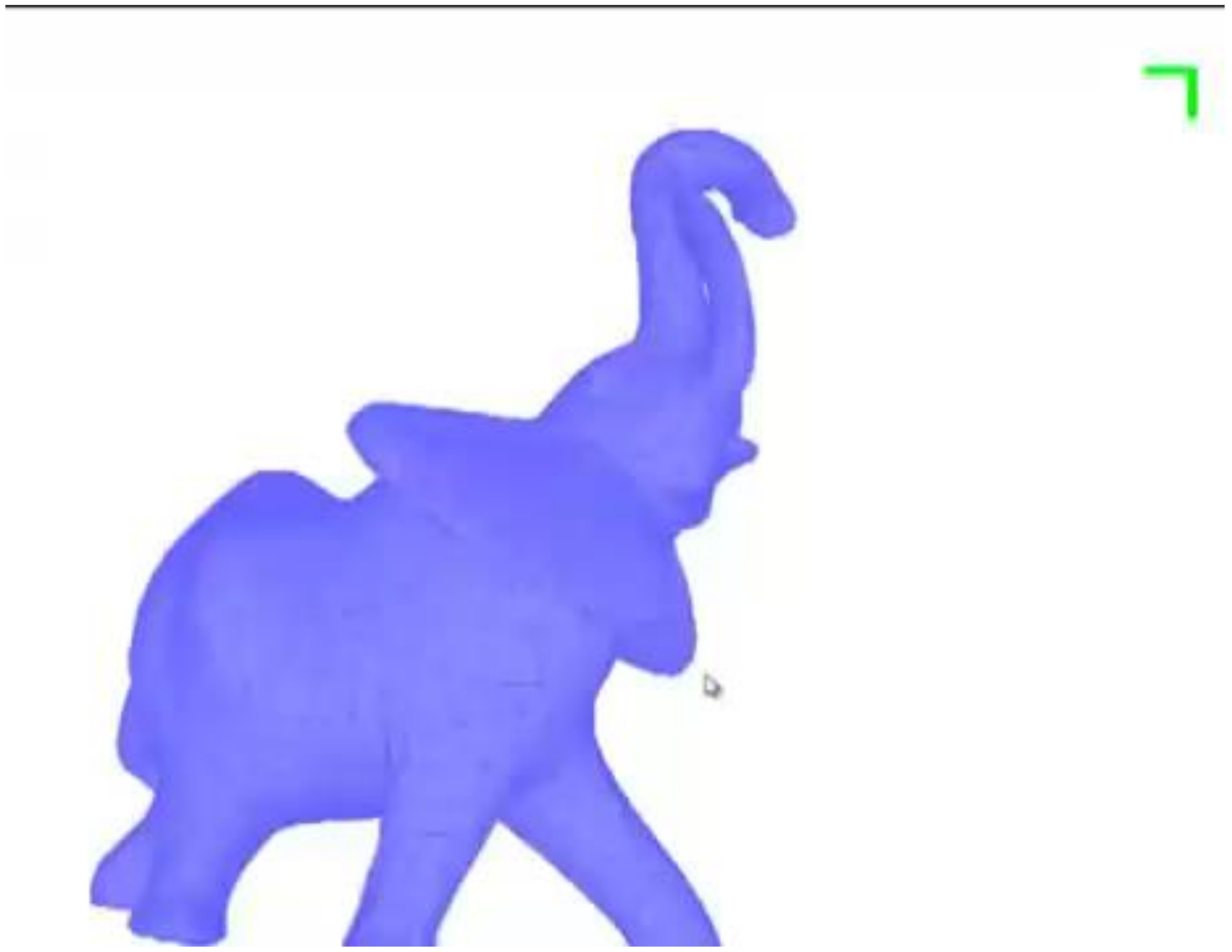
$$\left(\begin{array}{c|c} \multicolumn{2}{c}{L} \\ \hline \omega I_{m\times m} & 0 \end{array} \right) \mathbf{x} = \begin{pmatrix} \delta^{(x)} \\ \omega c_{1:m} \end{pmatrix}$$

$$\tilde{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(\|L\mathbf{x} - \delta^{(x)}\|^2 + \sum_{j \in C} \omega^2 |x_j - c_j|^2 \right)$$



Polygonal Mesh Processing

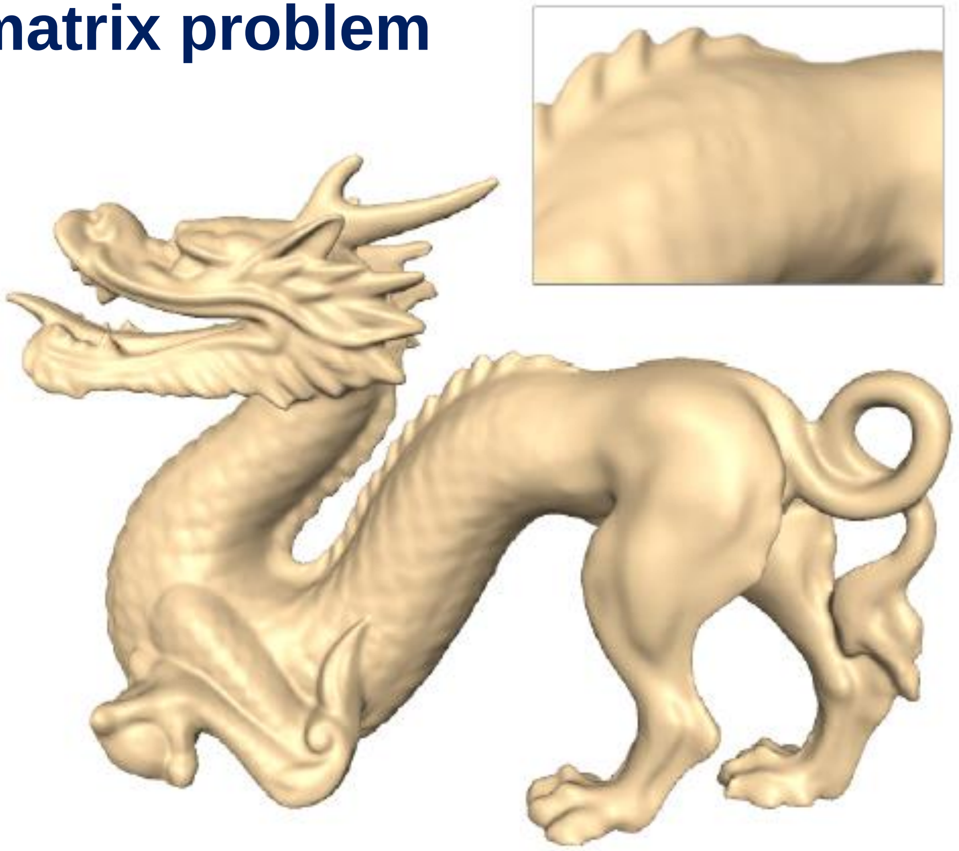
Deformation



Sorkine

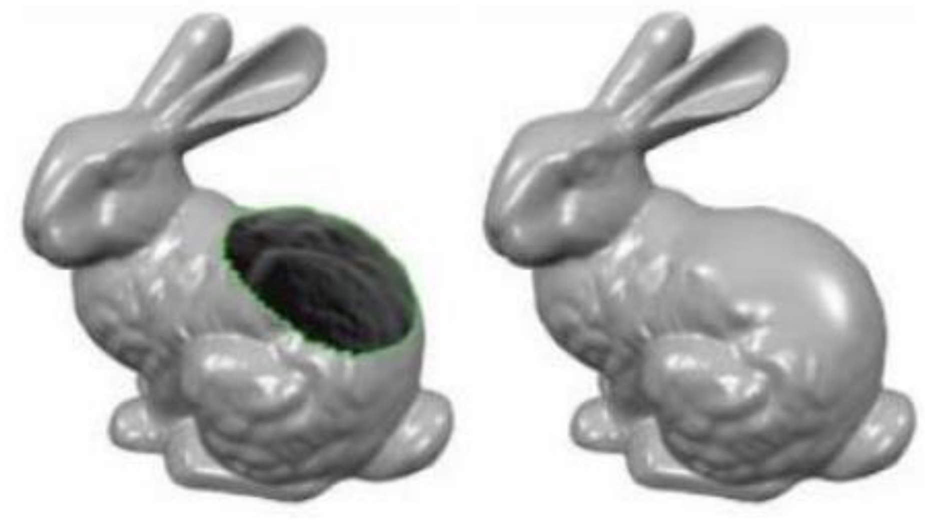
The Laplacian Operator

- Applicable to:
  - Deformation, by adding constraints
  - Blending, by **concatenating rows in matrix problem**



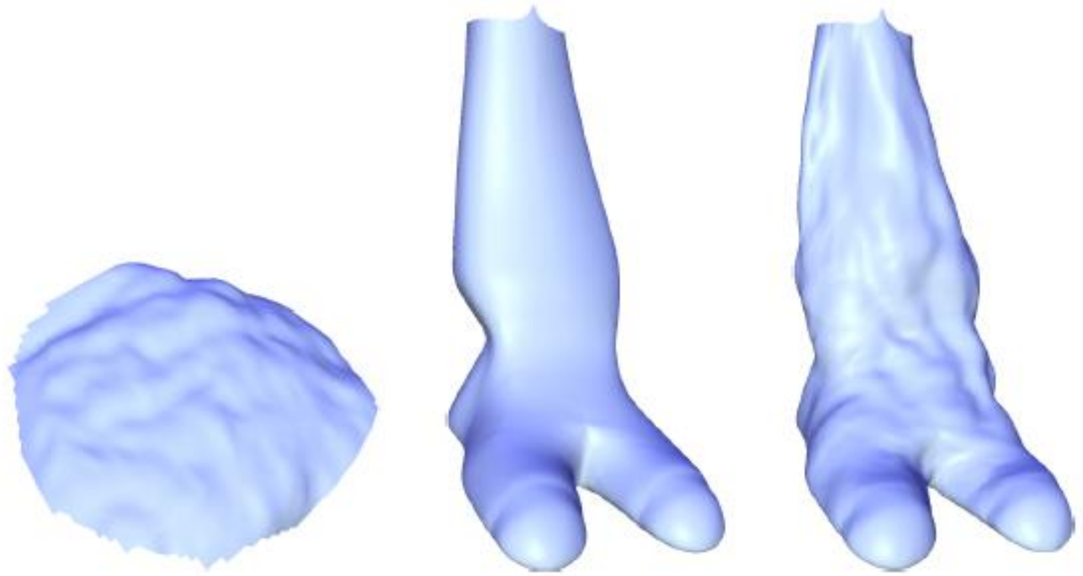
The Laplacian Operator

- Applicable to:
  - Deformation, by adding constraints
  - Blending, by concatenating rows
  - Hole filling, by 0’s on the RHS



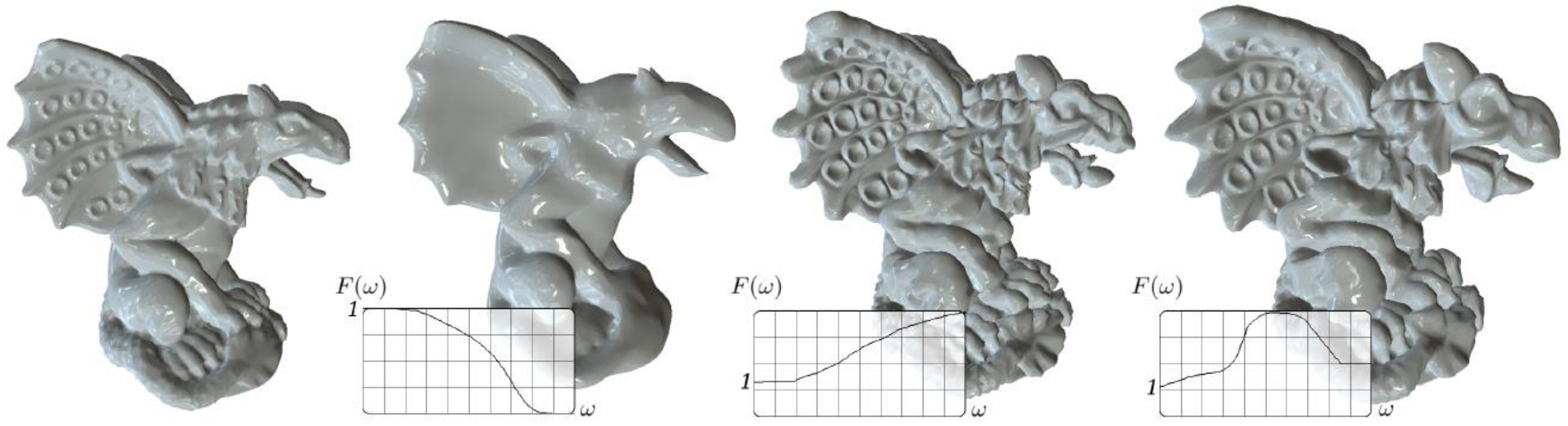
The Laplacian Operator

- Applicable to:
  - Deformation, by adding constraints
  - Blending, by concatenating rows
  - Hole filling, by 0’s on the RHS
  - Coating (or detail transfer), by copying RHS values (after filtering)



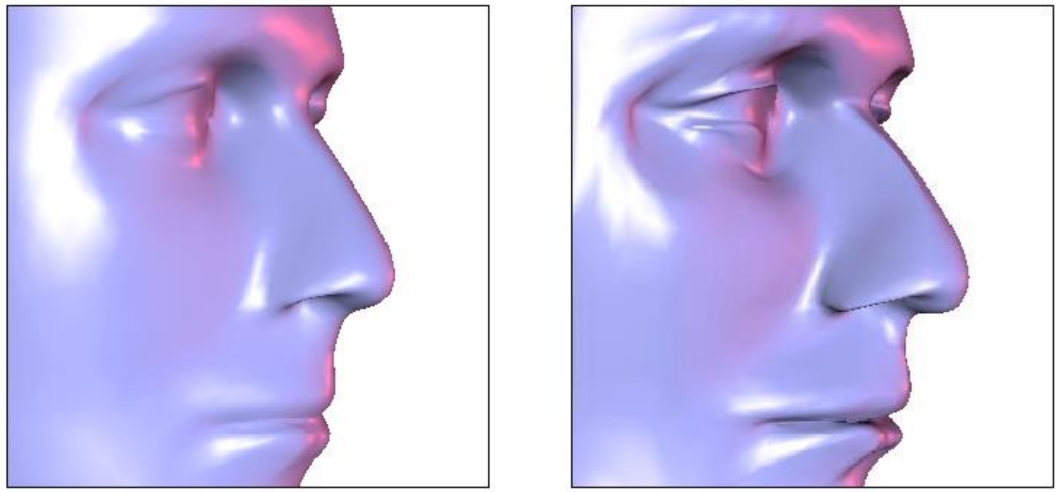
The Laplacian Operator

- Applicable to:
  - Deformation, by adding constraints
  - Blending, by concatenating rows
  - Hole filling, by 0's on the RHS
  - Coating (or detail transfer), by copying RHS values (after filtering)
  - Spectral mesh processing, through eigen analysis

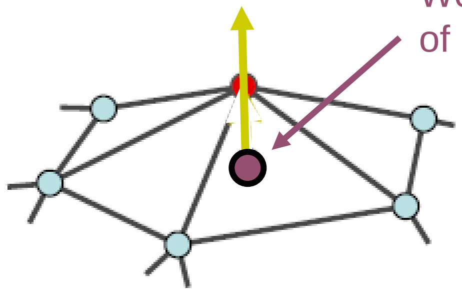


Polygonal Mesh Processing

- Analysis
  - Normals
  - Curvature
- Warps
  - Rotate
  - Deform
- Filters
  - Smooth
  - ➢Sharpen
  - Truncate



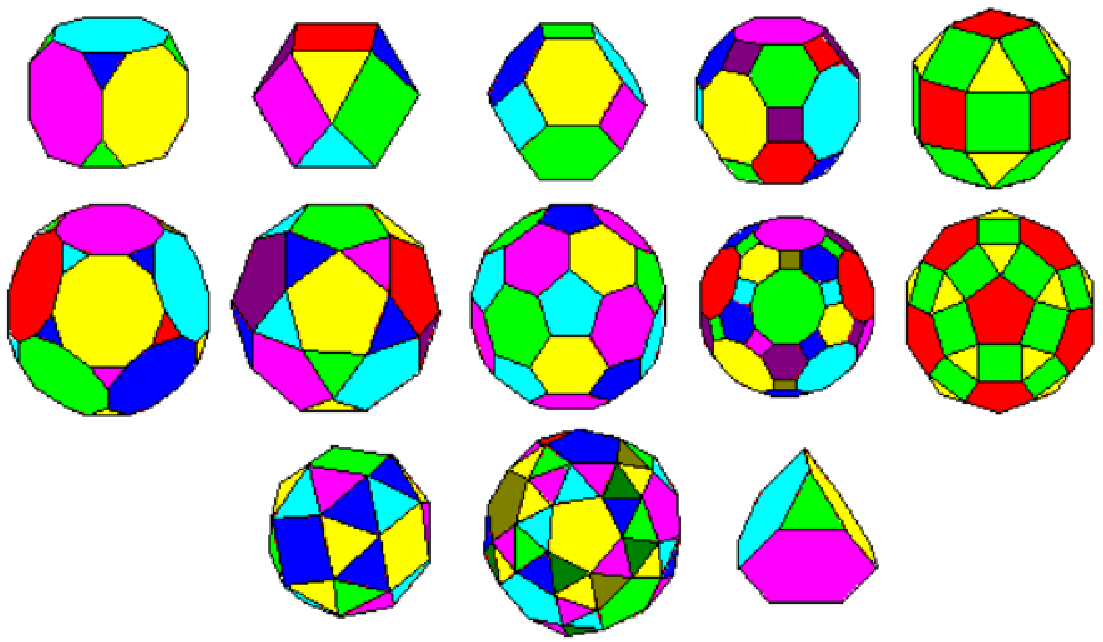
Desbrun



Olga Sorkine

Polygonal Mesh Processing

- Analysis
  - Normals
  - Curvature
- Warps
  - Rotate
  - Deform
- Filters
  - Smooth
  - Sharpen
  - ➢Truncate



Archimedean Polyhedra

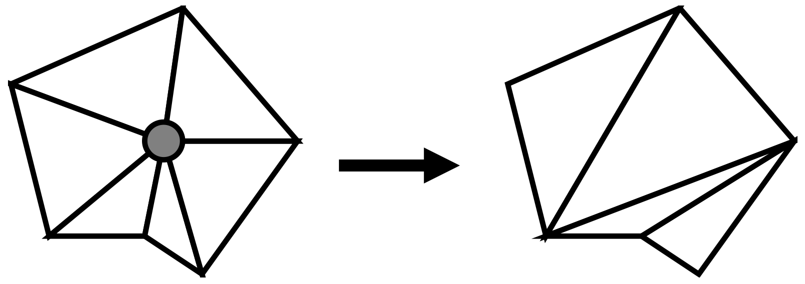
http://www.uwgb.edu/dutchs/symmetry/archpol.htm

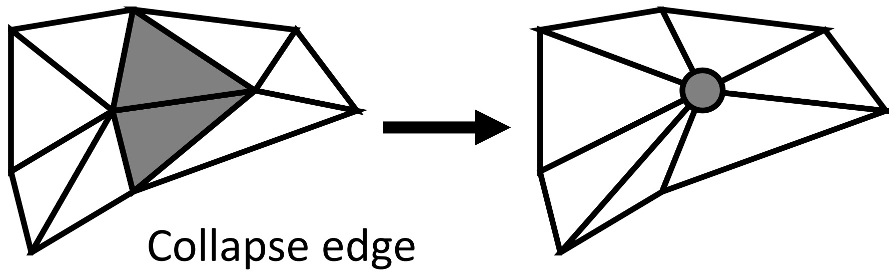
Polygonal Mesh Processing

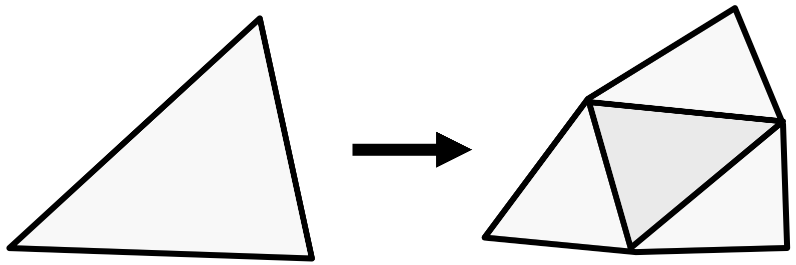
- Remeshing
  - Subdivide
  - Resample
  - Simplify
- Topological fixup
  - Fill holes
  - Fix self-intersections
- Boolean operations
  - Crop
  - Subtract

Polygonal Mesh Processing

- Remeshing
  - Subdivide
  - Resample
  - Simplify
- Topological fixup
  - Fill holes
  - Fix self-intersections
- Boolean operations
  - Crop
  - Subtract

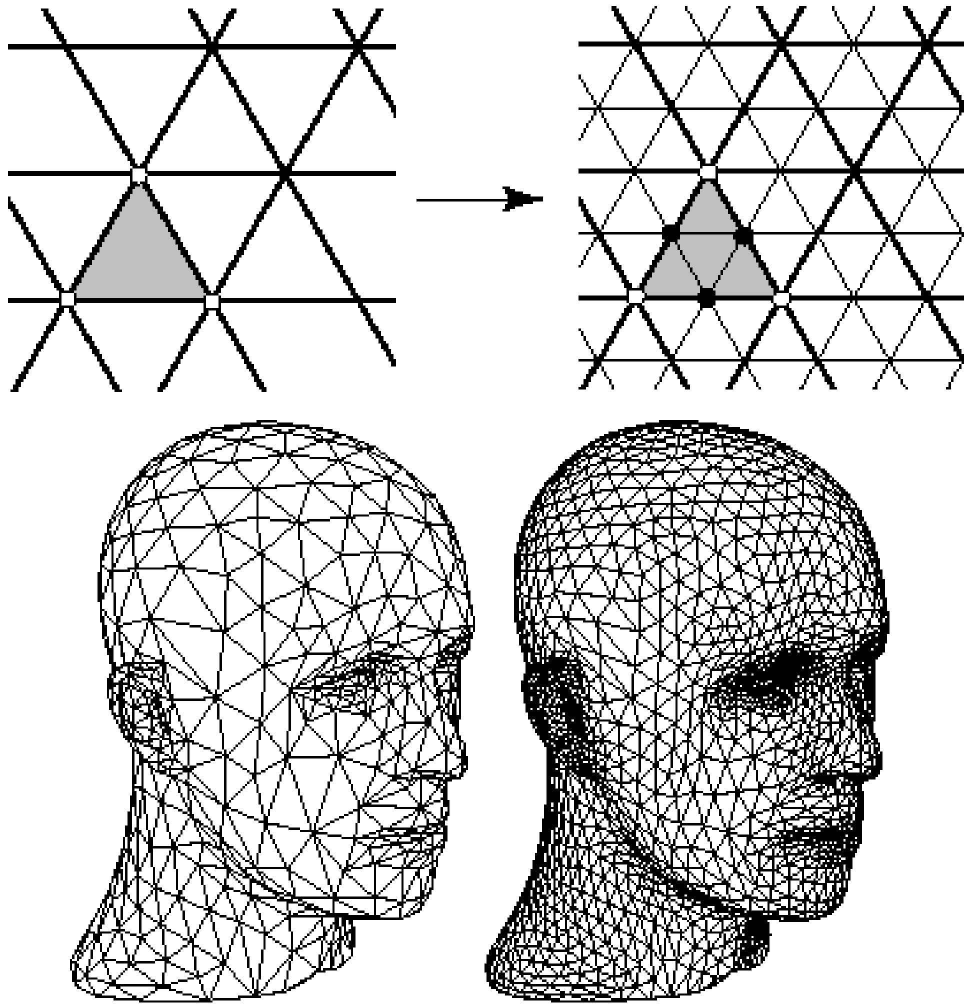

Remove Vertex


Collapse edge


Subdivide face

Polygonal Mesh Processing

- Remeshing
  - ➤Subdivide
  - Resample
  - Simplify
- Topological fixup
  - Fill holes
  - Fix self-intersections
- Boolean operations
  - Crop
  - Subtract



Zorin & Schroeder

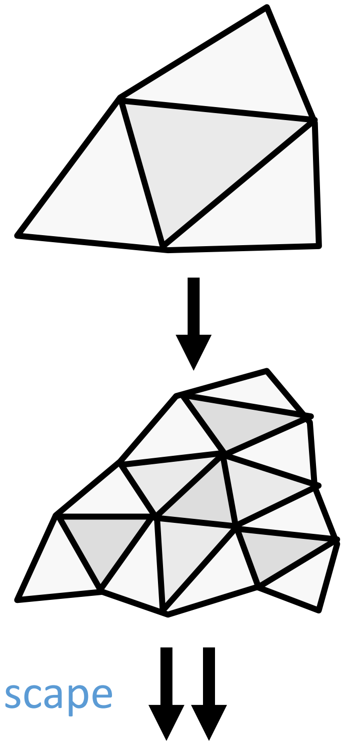
Polygonal Mesh Processing

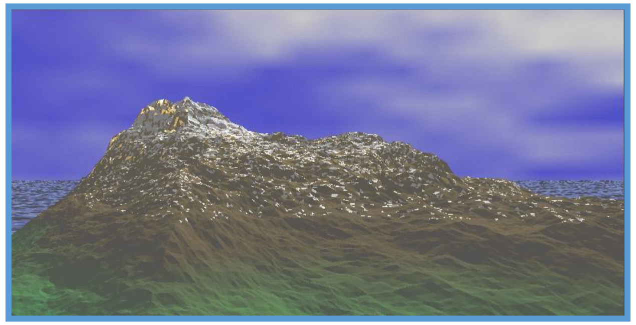
- Remeshing
  - ➢Subdivide
  - Resample
  - Simplify
- Topological fixup
  - Fill holes
  - Fix self-intersections
- Boolean operations
  - Crop
  - Subtract

Matt Matl, COS 426, 2014

Polygonal Mesh Processing

- Remeshing
  - ➢ Subdivide
  - Resample
  - Simplify
- Topological fixup
  - Fill holes
  - Fix self-intersections
- Boolean operations
  - Crop
  - Subtract

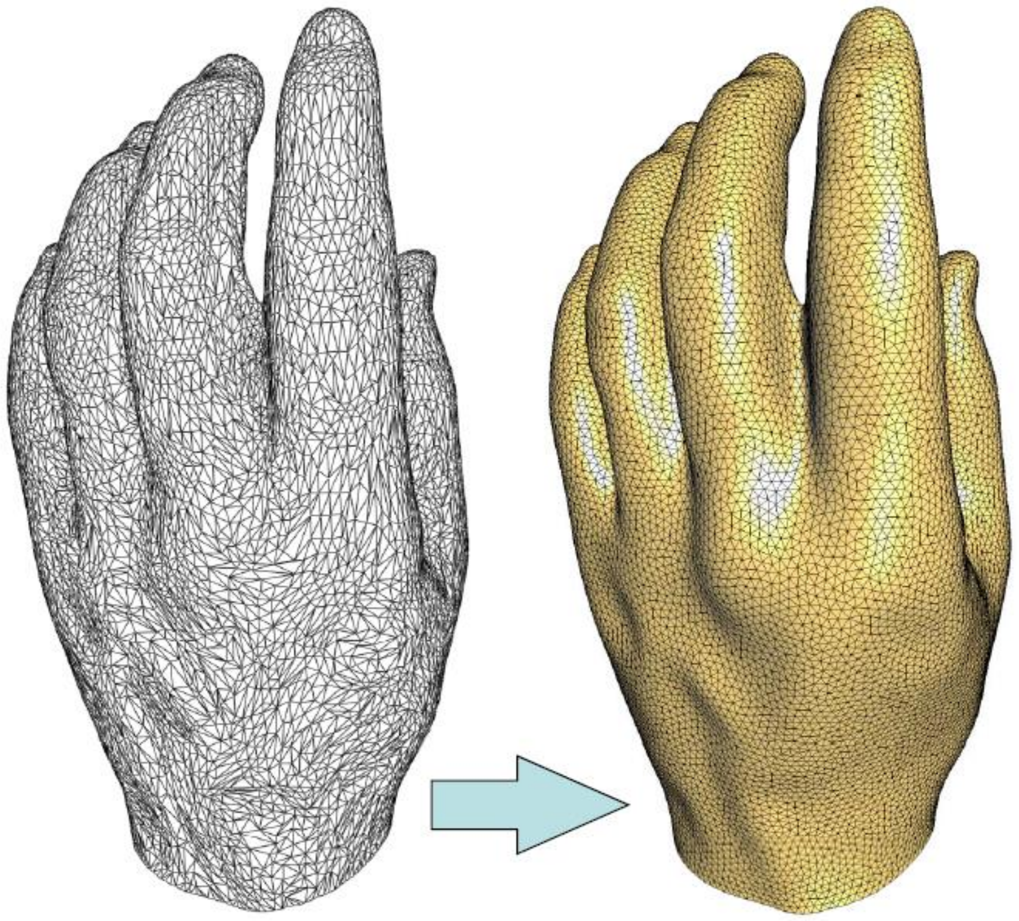




Dirk Balfanz, Igor Guskov,
Sanjeev Kumar, & Rudro Samanta,

Polygonal Mesh Processing

- Remeshing
  - Subdivide
  - ➢Resample
  - Simplify
- Topological fixup
  - Fill holes
  - Fix self-intersections
- Boolean operations
  - Crop
  - Subtract

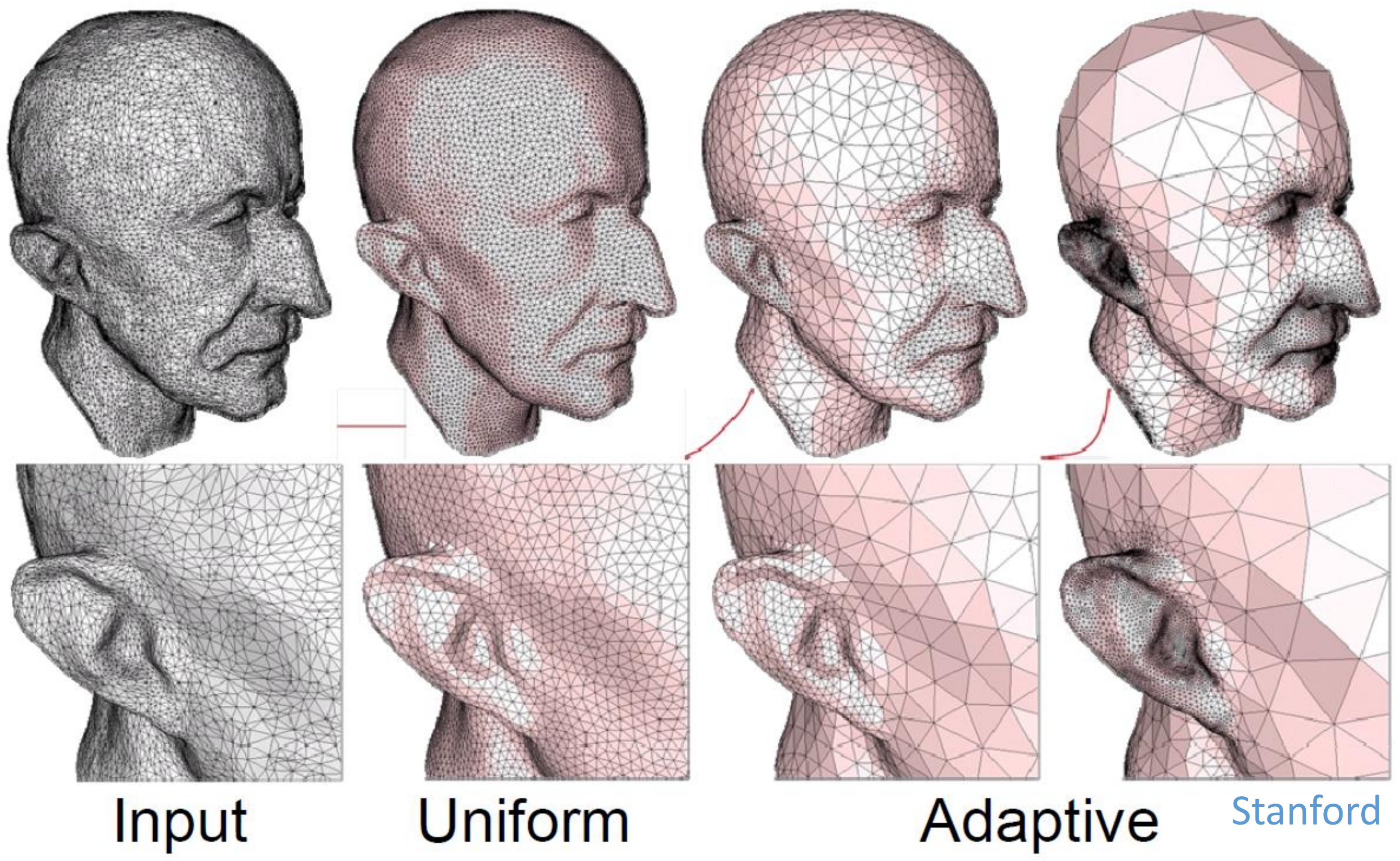


Stanford

- more uniform distribution
- triangles with nicer aspect

Polygonal Mesh Processing

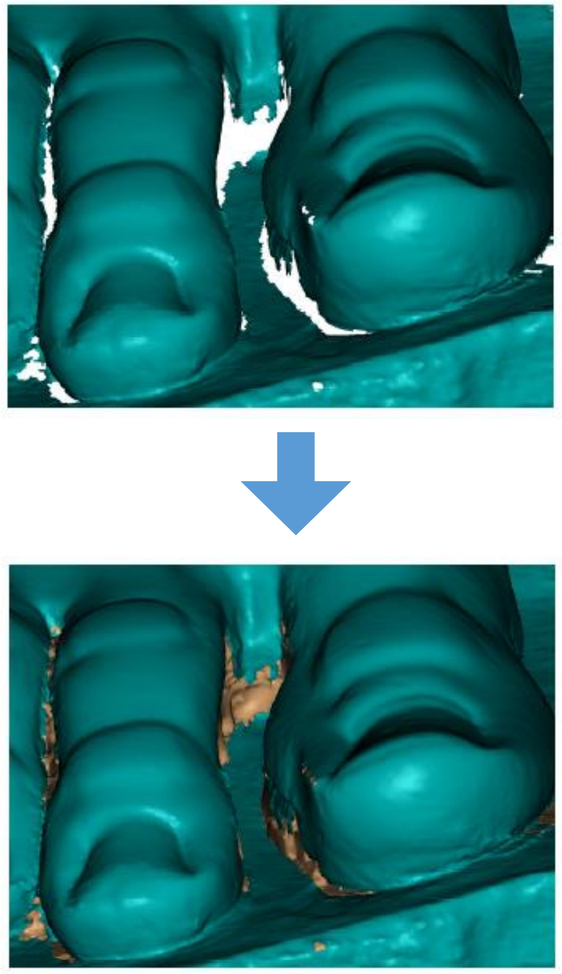
- Remeshing
  - Subdivide
  - Resample
  - ➢Simplify
- Topological fixup
  - Fill holes
  - Fix self-intersections
- Boolean operations
  - Crop
  - Subtract



Stanford

Polygonal Mesh Processing

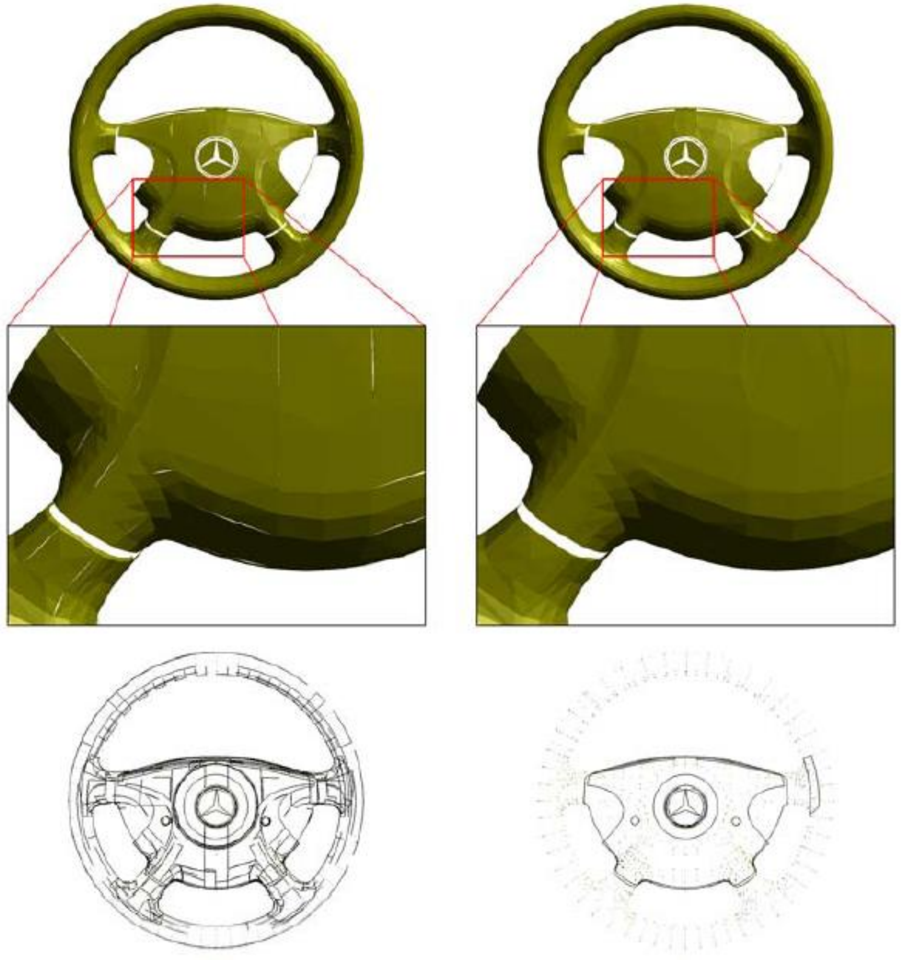
- Remeshing
  - Subdivide
  - Resample
  - Simplify
- Topological fixup
  - ➢Fill holes
  - Fix self-intersections
- Boolean operations
  - Crop
  - Subtract



Podolak

Polygonal Mesh Processing

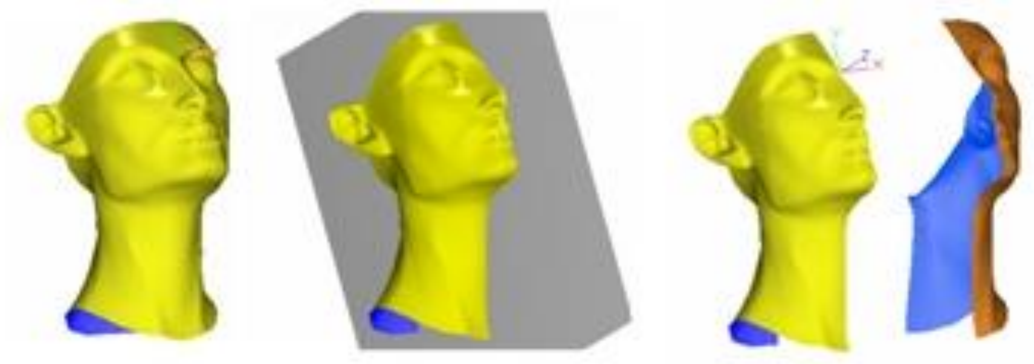
- Remeshing
  - Subdivide
  - Resample
  - Simplify
- Topological fixup
  - Fill holes
  - ➢Fix self-intersections
- Boolean operations
  - Crop
  - Subtract



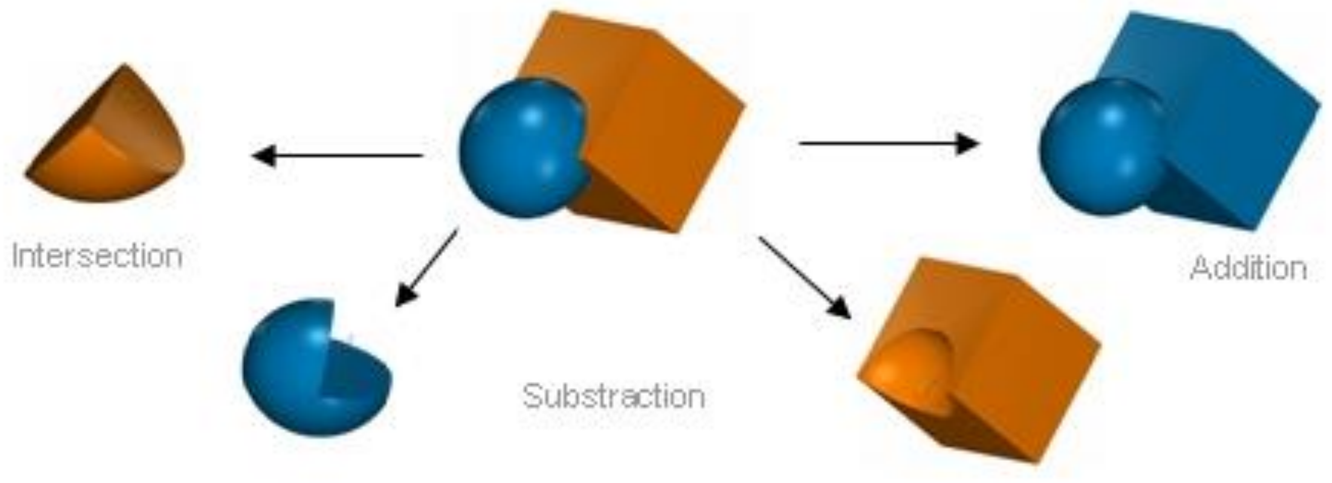
Borodin

Polygonal Mesh Processing

- Remeshing
  - Subdivide
  - Resample
  - Simplify
- Topological fixup
  - Fill holes
  - Fix self-intersections
- Boolean operations
  - ➢Crop
  - ➢Subtract
  - ➢Etc.



Mesh separation processed by a boolean operation.



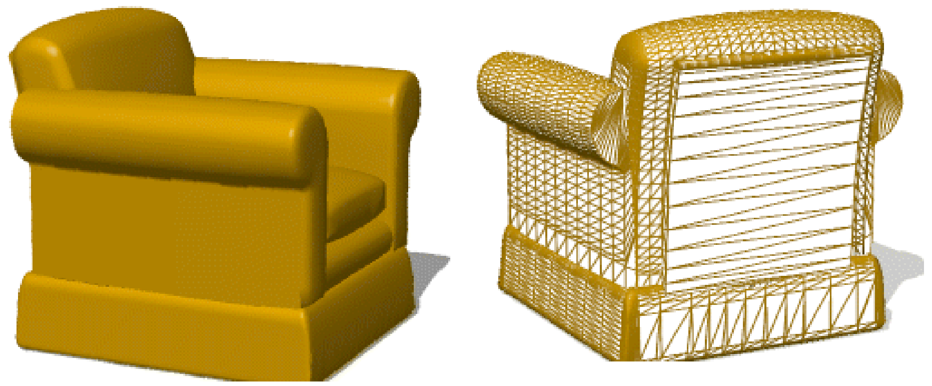
Several Boolean operations with 3DReshaper®

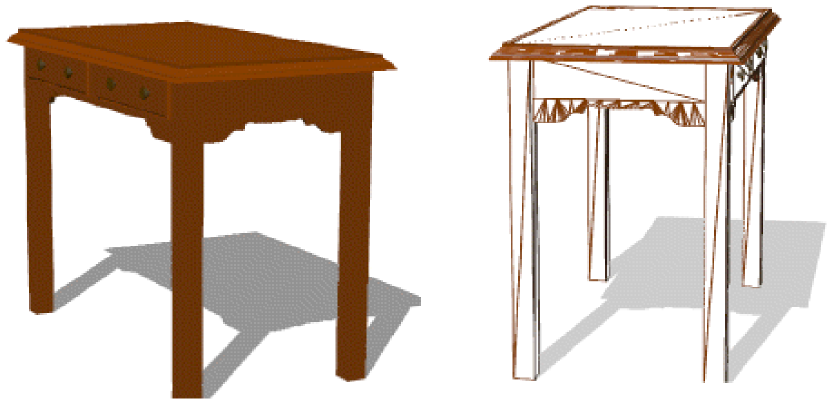
Summary

- Polygonal meshes
  - Most common surface representation
  - Fast rendering
- Processing operations
  - Must consider irregular vertex sampling
  - Must handle/avoid topological degeneracies
- Representation
  - Which adjacency relationships to store
    depend on which operations must be efficient

3D Polygonal Meshes

- Properties
  - ? Efficient display
  - ? Easy acquisition
  - ? Accurate
  - ? Concise
  - ? Intuitive editing
  - ? Efficient editing
  - ? Efficient intersections
  - ? Guaranteed validity
  - ? Guaranteed smoothness
  - ? etc.

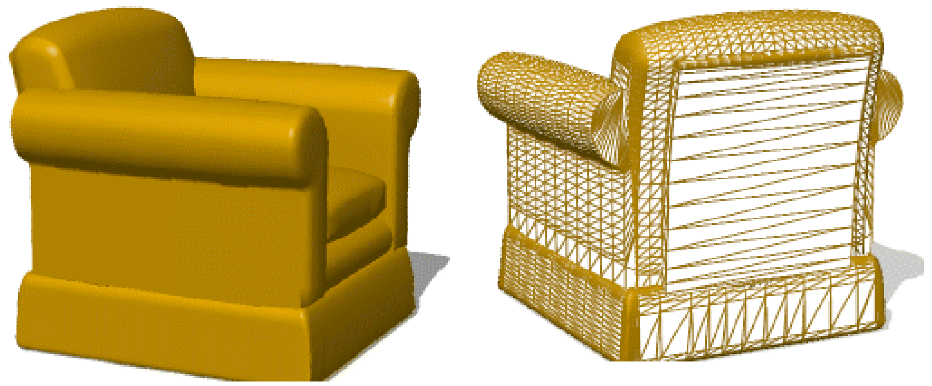
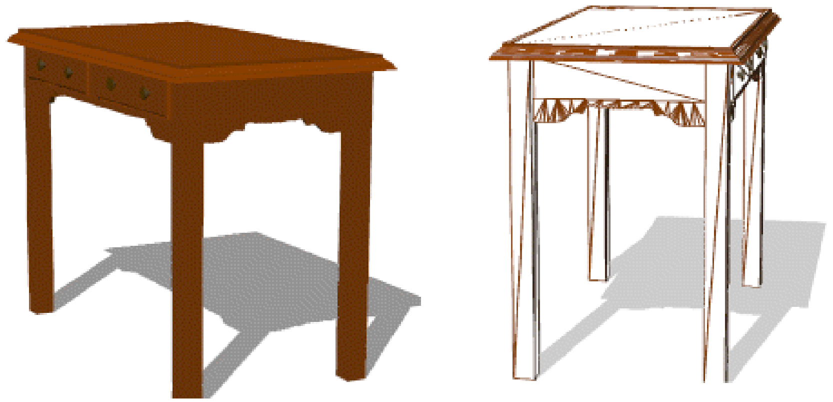




Viewpoint

3D Polygonal Meshes

- Properties
  - ☺Efficient display
  - ☺Easy acquisition
  - ☹Accurate
  - ☹Concise
  - ☹Intuitive editing
  - ☹Efficient editing
  - ☹Efficient intersections
  - ☹Guaranteed validity
  - ☹Guaranteed smoothness

Viewpoint