Polygonal Meshes

COS 426

3D Object Representations

Points
- Range image
- Point cloud

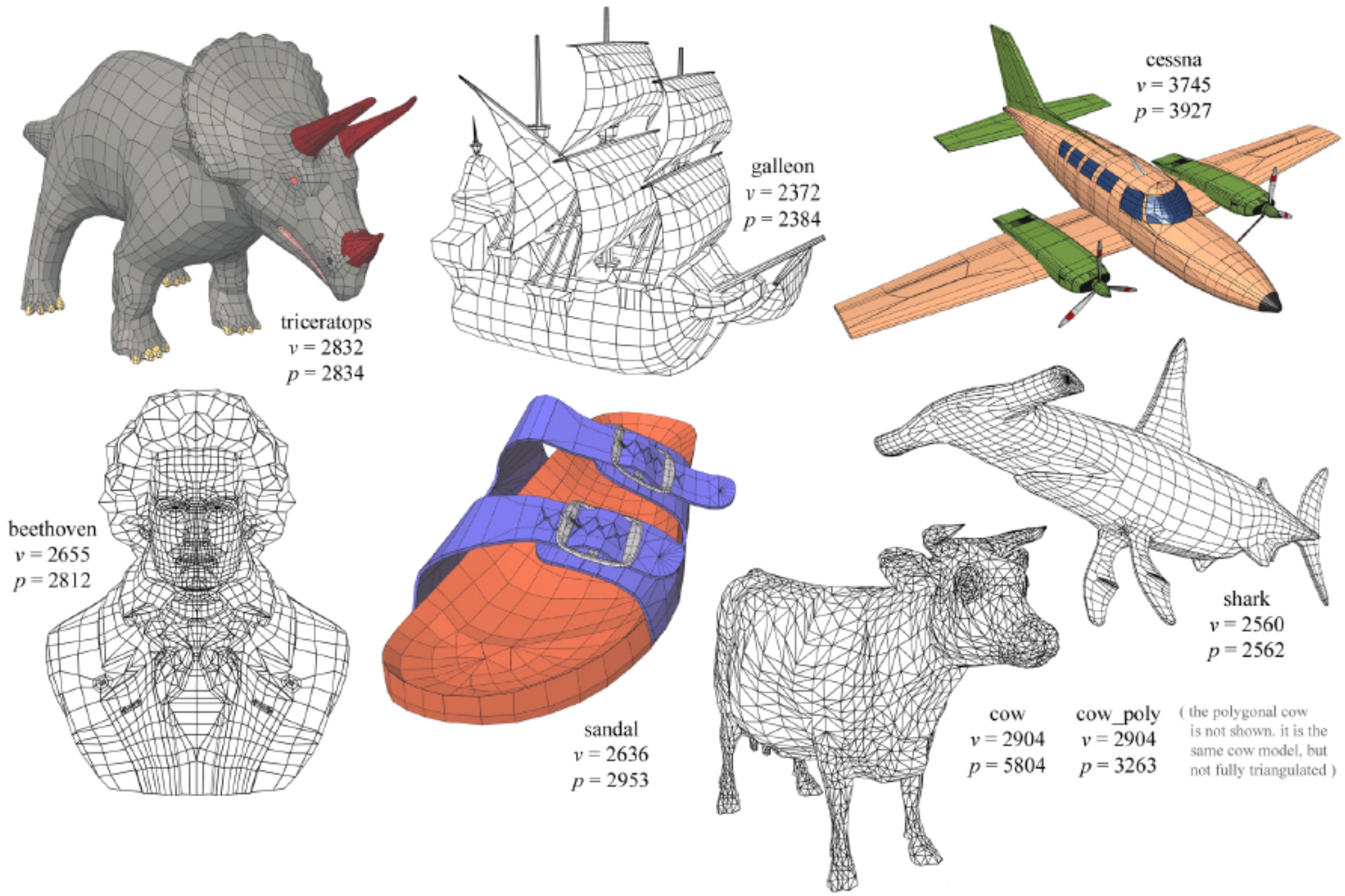
Surfaces
- ➢ Polygonal mesh
- Subdivision
- Parametric
- Implicit

Solids
- Voxels
- BSP tree
- CSG
- Sweep

High-level structures
- Scene graph
- Application specific

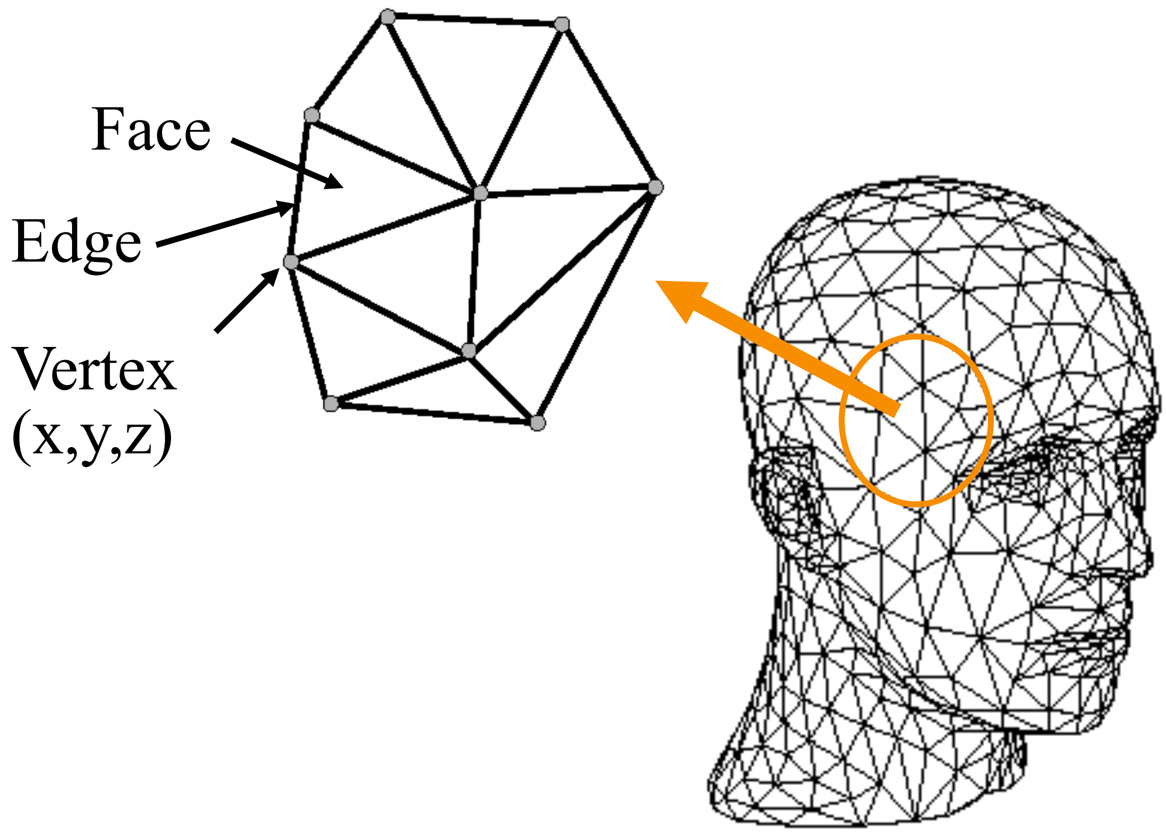
3D Polygonal Mesh

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



3D Polygonal Mesh

Geometry & topology



Zorin & Schroeder

Geometry background

Scene is usually approximated by 3D primitives

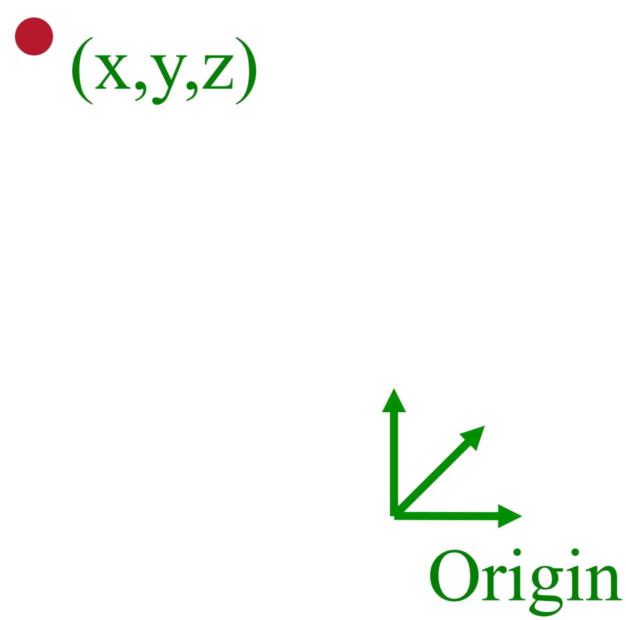
- Point
- Vector
- Line segment
- Ray
- Line
- Plane
- Polygon

3D Point

Specifies a location

- Represented by three coordinates
- Infinitely small

```
struct Point {
    Coordinate x;
    Coordinate y;
    Coordinate z;
};
```

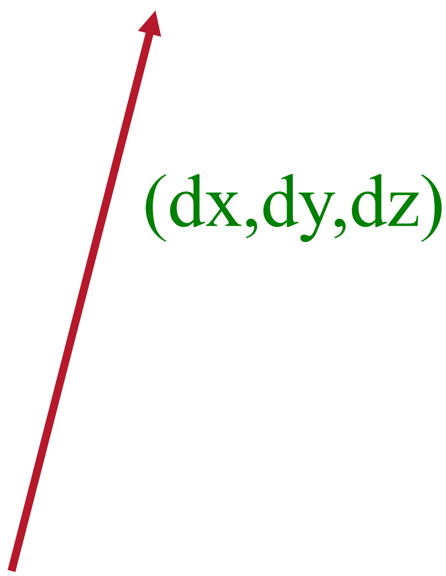


3D Vector

Specifies a direction and a magnitude

- Represented by three coordinates
- Magnitude ||V|| = sqrt(dx*dx + dy*dy + dz*dz)
- Has no location

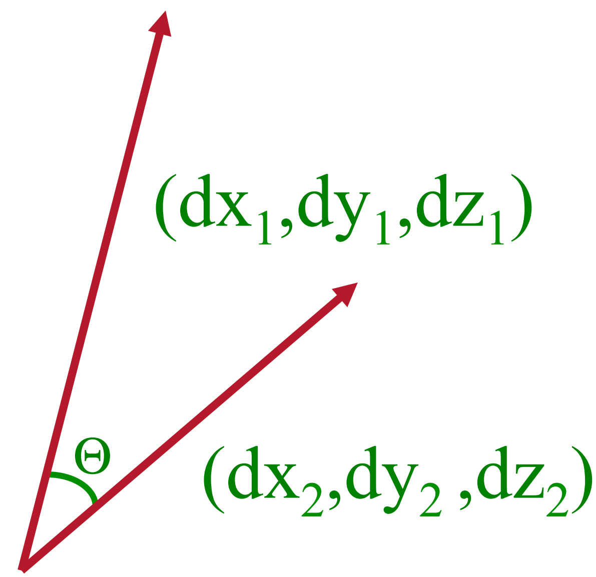
```
struct Vector {
    Coordinate dx;
    Coordinate dy;
    Coordinate dz;
};
```



3D Vector

Scalar / dot product of two 3D vectors

$$V_1 \cdot V_2 = dx_1 * dx_2 + dy_1 * dy_2 + dz_1 * dz_2 = ||V_1|| \, ||V_2|| \cos(\Theta)$$

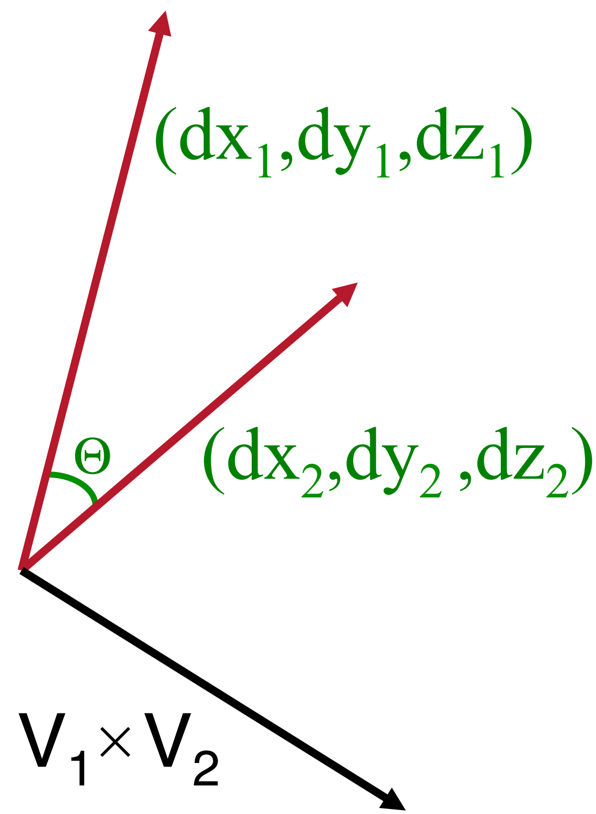


3D Vector

Cross product of two 3D vectors

$V_1 \times V_2 = (dy_1dx_2 - dz_1dy_2, dz_1dx_2 - dx_1dz_2, dx_1dy_2 - dy_1dx_2)$

- vector perpendicular to both V_1 and V_2
- magnitude is $||V_1|| \, ||V_2|| \sin(\Theta)$

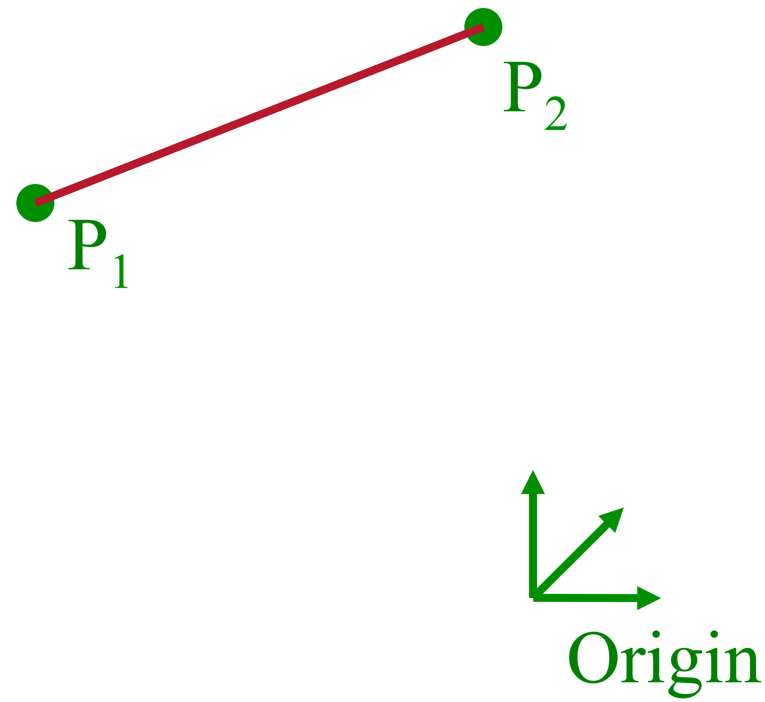


3D Line Segment

Linear path between two points

- Parametric representation:
  - » $p = P_1 + t\,(P_2 - P_1), \quad (0 \le t \le 1)$

```
struct Segment {
    Point P1;
    Point P2;
};
```

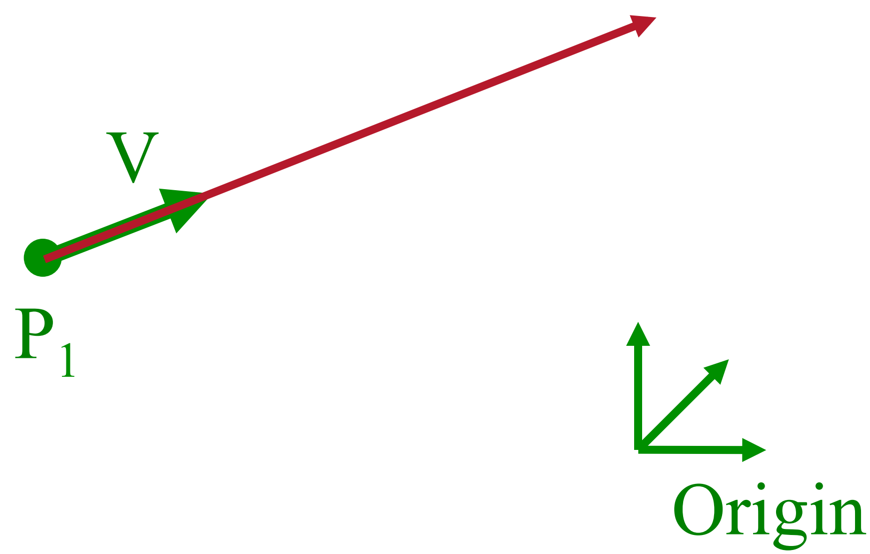


3D Ray

Line segment with one endpoint at infinity

- Parametric representation:
  - » $p = P_1 + t\,V, \quad (0 \leq t < \infty)$

```
struct Ray {
    Point P1;
    Vector V;
};
```

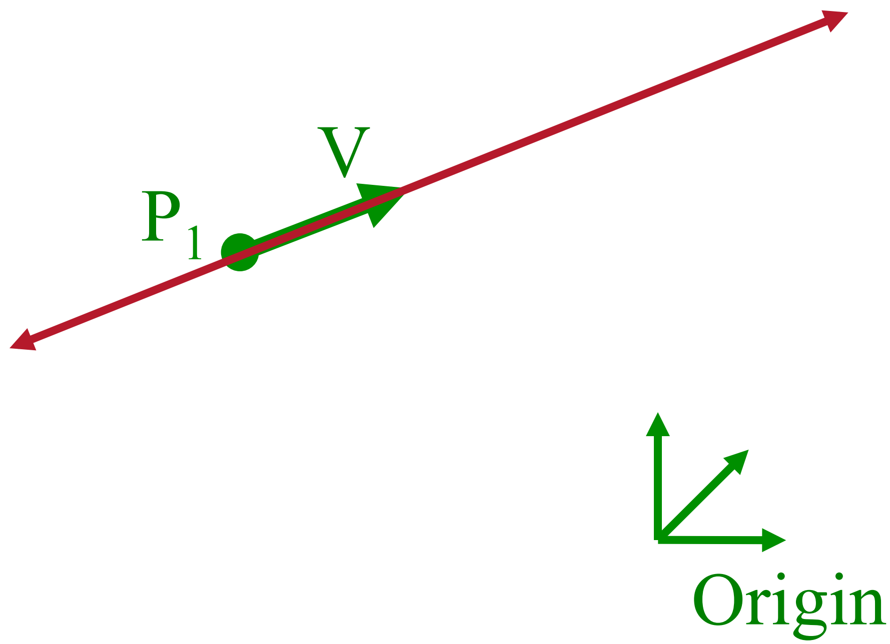


3D Line

Line segment with both endpoints at infinity

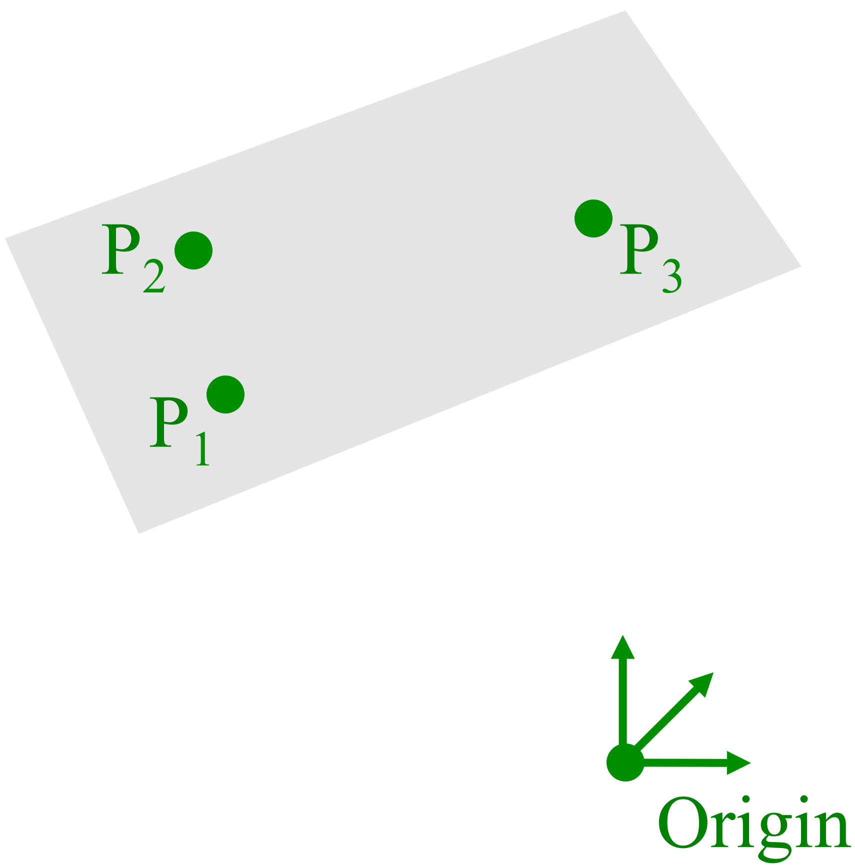
- Parametric representation:
  - » $p = P_1 + t\,V, \quad (-\infty < t < \infty)$

```
struct Line {
    Point P1;
    Vector V;
};
```



3D Plane

Defined by three points



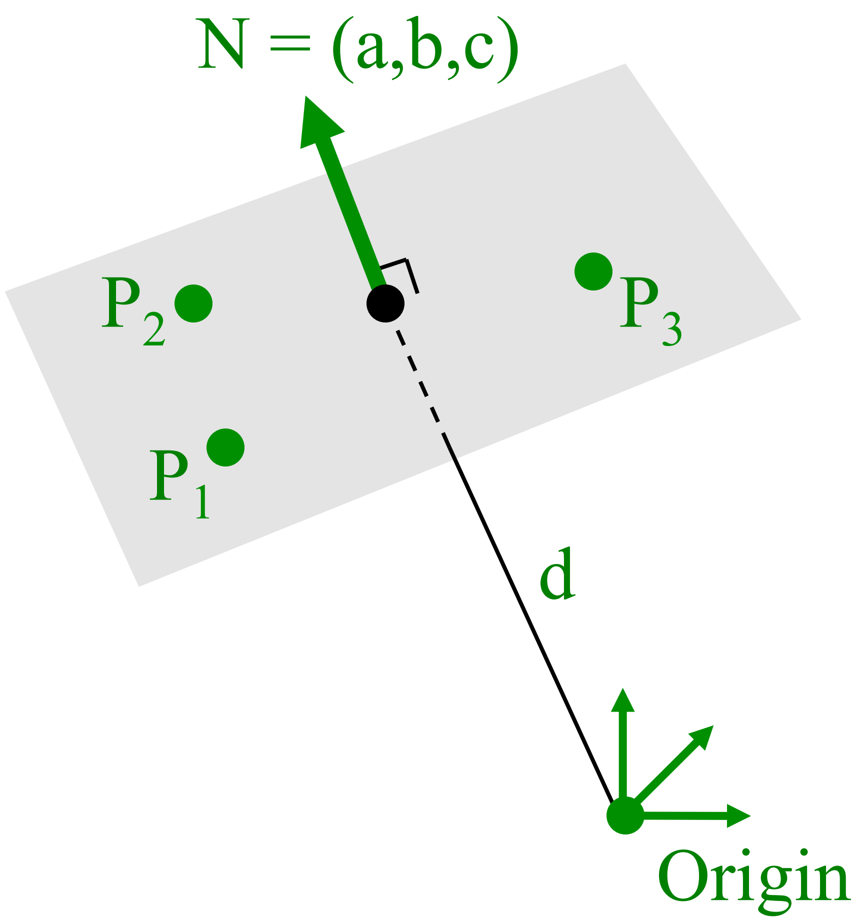
3D Plane

Defined by three points

- Implicit representation:
  - » $ax + by + cz + d = 0$

    OR
  - » $P \cdot N + d = 0$

```
struct Plane {
    Vector N;
    float d;
};
```

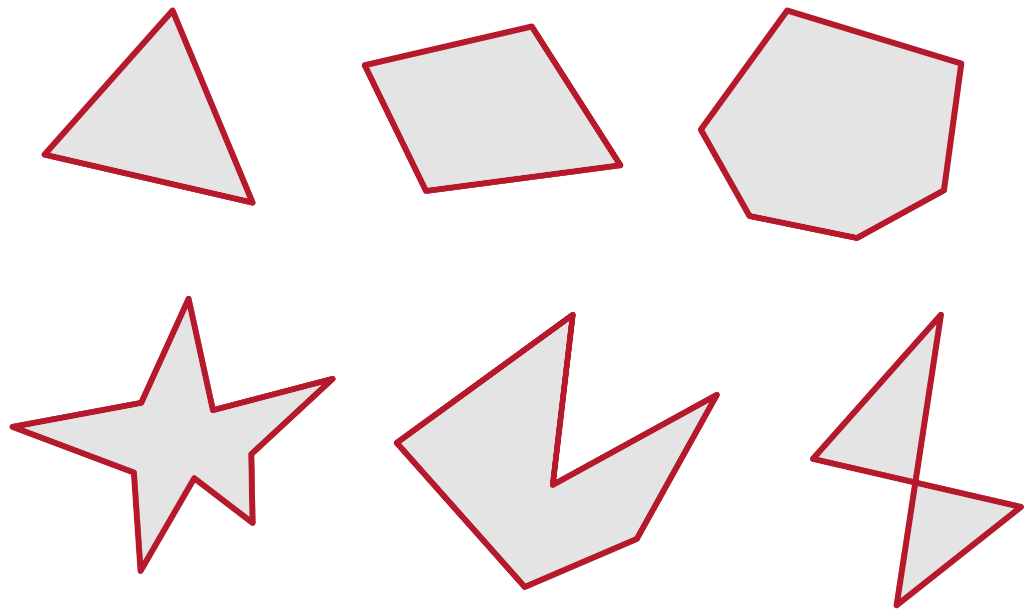


- N is the plane normal
  - » Unit-length
  - » Perpendicular to plane

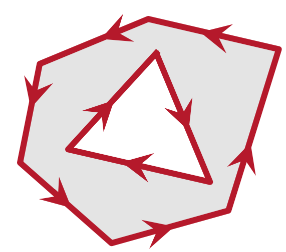
3D Polygon

Region “inside” a sequence of coplanar points

```
struct Polygon {
      vector<Point> points;
};
```



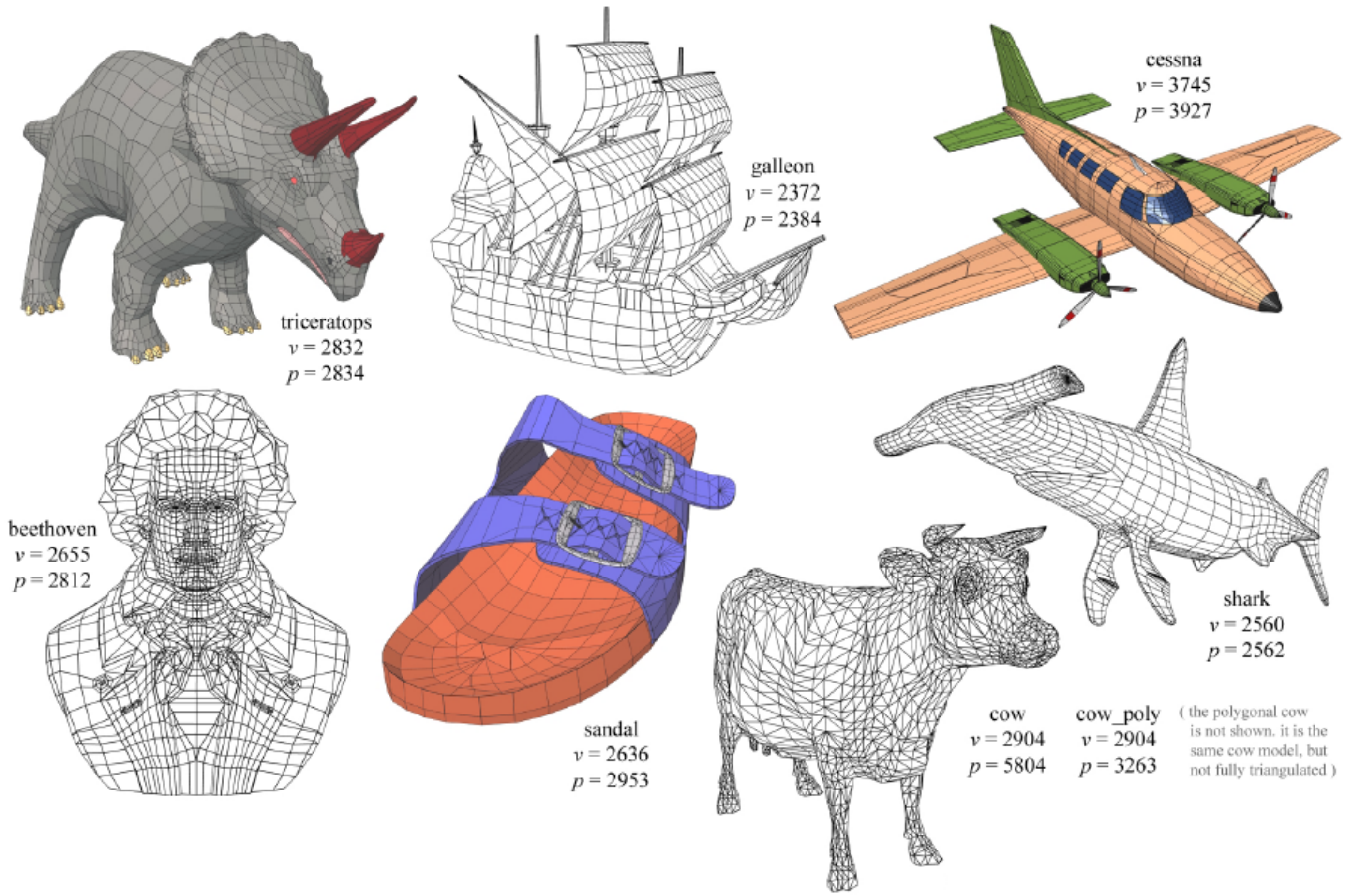
- Points in counter-clockwise order (defines normal)



- Winding rule determines inside/outside

3D Polygonal Mesh

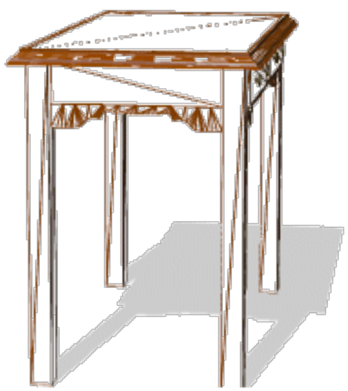
Set of polygons representing a 2D surface embedded in 3D

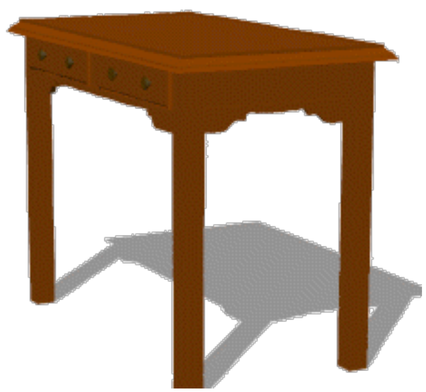


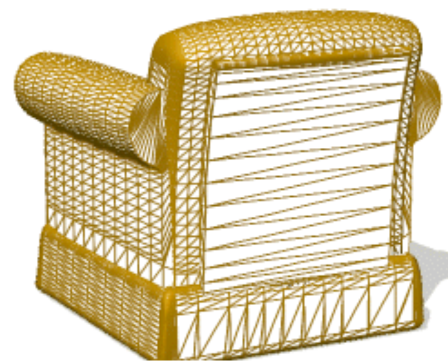
3D Polygonal Meshes

Why are they of interest?

- Simple, common representation
- Rendering with hardware support
- Output of many acquisition tools
- Input to many simulation/analysis tools





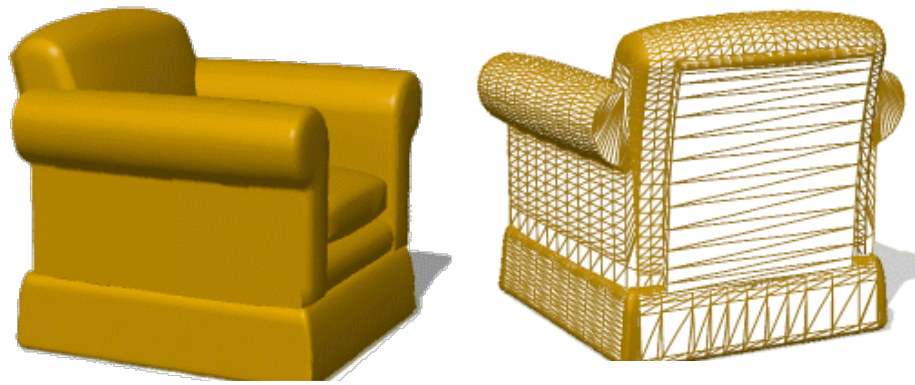


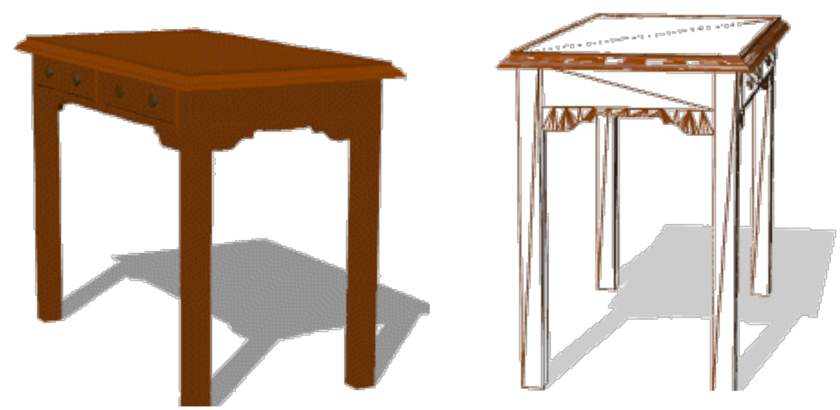
Viewpoint

3D Polygonal Meshes

Properties

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





Viewpoint

Outline

Acquisition ←

Processing

Representation

Polygonal Mesh Acquisition

Interactive modeling

- Polygon editors
- Interchange formats

Scanners

- Laser range scanners
- Geological survey
- CAT, MRI, etc. (isosurfaces)

Simulations

- Physical processes

Polygonal Mesh Acquisition

Interactive modeling

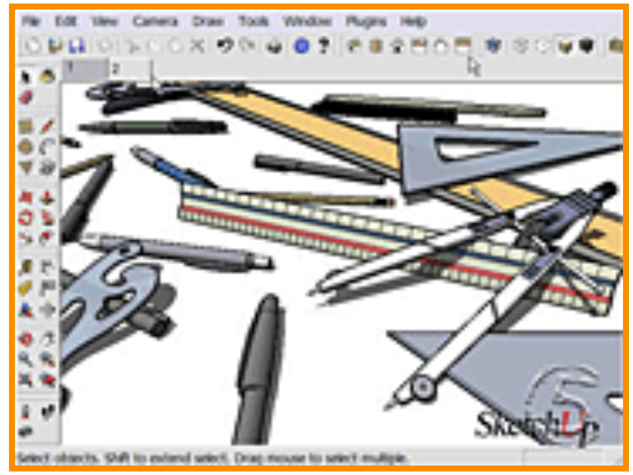
- ➢ Polygon editors
- ◦ Interchange formats

Scanners

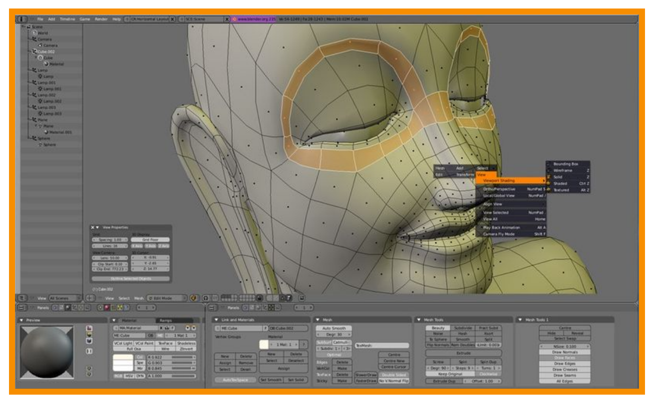
- ◦ Laser range scanners
- ◦ Geological survey
- ◦ CAT, MRI, etc. (isosurfaces)

Simulations

- ◦ Physical processes



Sketchup



Blender

Polygonal Mesh Acquisition

Interactive modeling
- Polygon editors
- ➢ Interchange formats

Scanners
- Laser range scanners
- Geological survey
- CAT, MRI, etc.

Simulations
- Physical processes




Jose Maria De Espona

Polygonal Mesh Acquisition

Interactive modeling

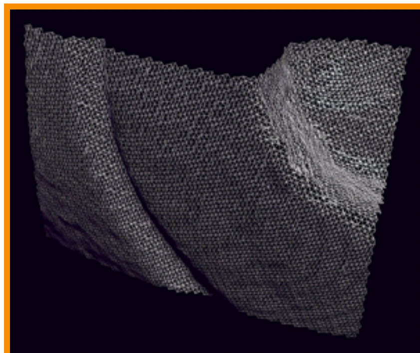
- Polygon editors
- Interchange formats

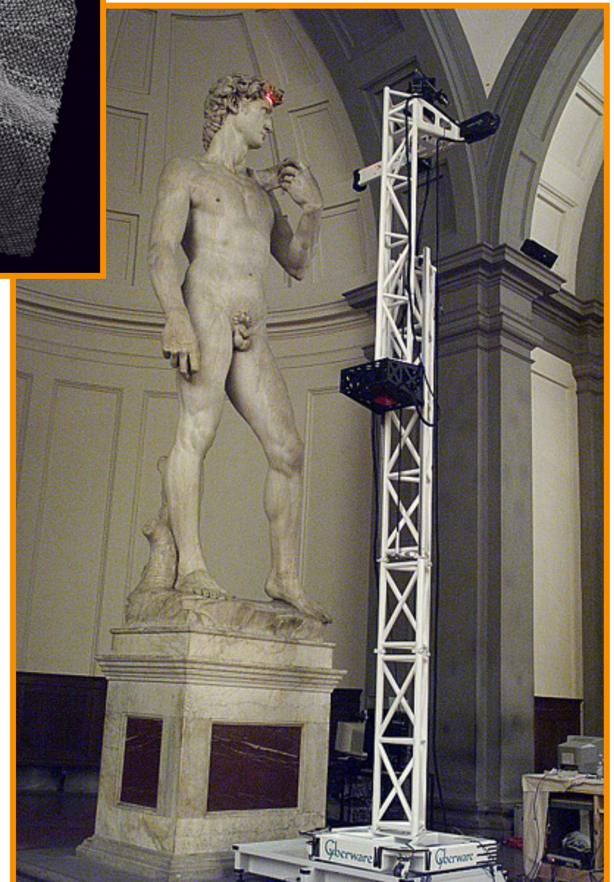
Scanners

- ➢ Laser range scanners
- Geological survey
- CAT, MRI, etc. (isosurfaces)

Simulations

- Physical processes





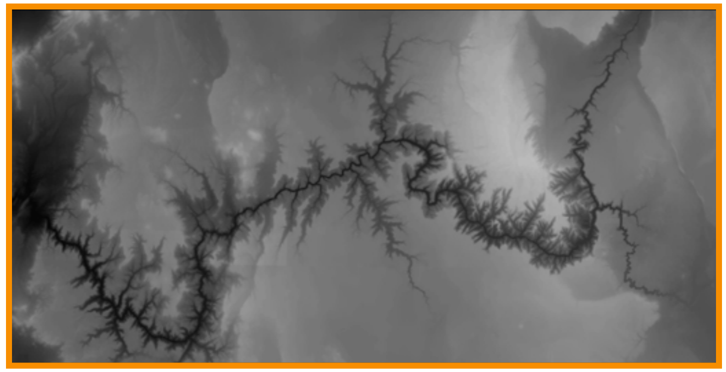
Digital Michelangelo Project
Stanford

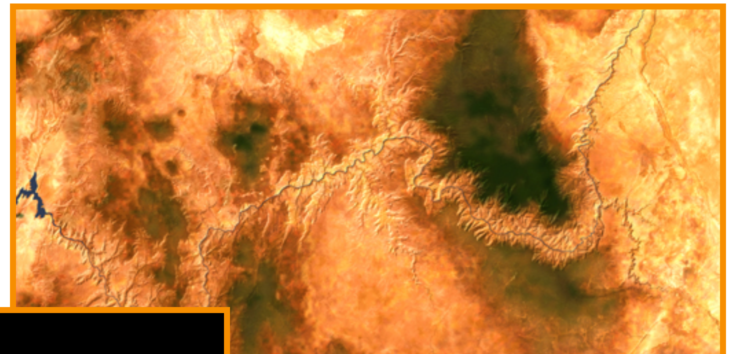
Polygonal Mesh Acquisition

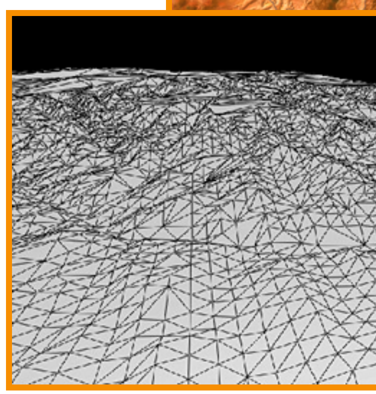
Interactive modeling
- Polygon editors
- Interchange formats

Scanners
- Laser range scanners
- ➢ Geological survey
- CAT, MRI, etc. (isosurfaces)

Simulations
- Physical processes







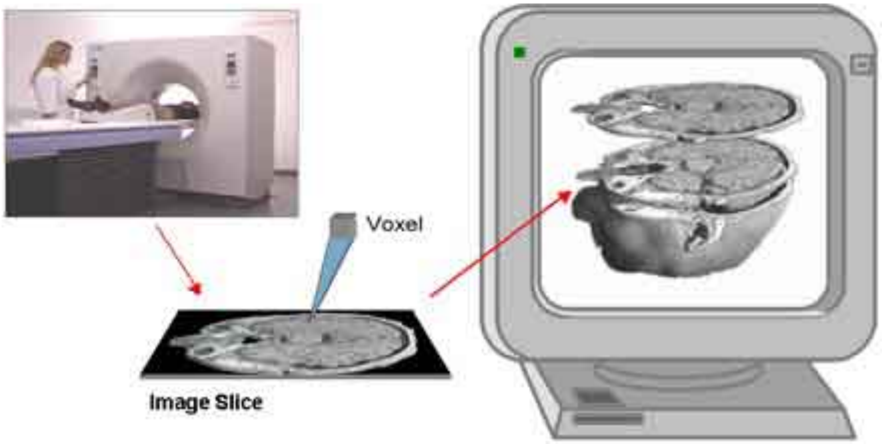
Large Geometric
Model Repository
Georgia Tech

Polygonal Mesh Acquisition

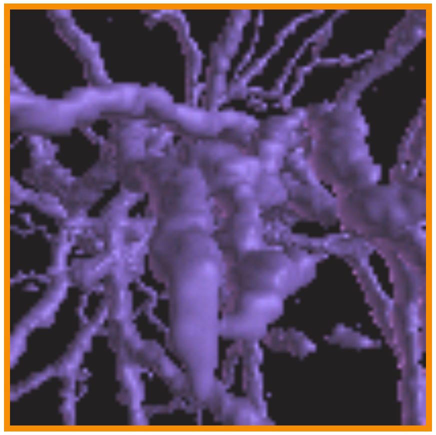
Interactive modeling
- Polygon editors
- Interchange formats

Scanners
- Laser range scanners
- Geological survey
- ➤ CAT, MRI, etc. (isosurfaces)

Simulations
- Physical processes



www.volumegraphics.com



SUNY Stony Brook

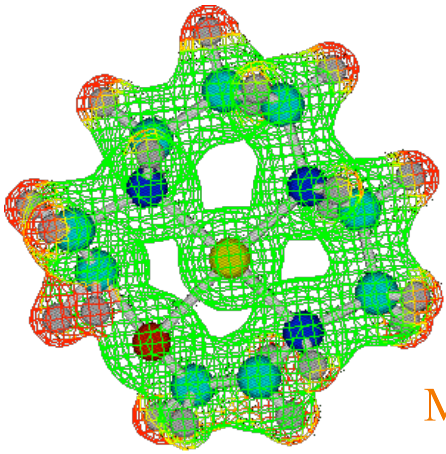
Polygonal Mesh Acquisition

Interactive modeling
- Polygon editors
- Interchange formats

Scanners
- Laser range scanners
- Geological survey
- CAT, MRI, etc. (isosurfaces)

Simulations
- Physical processes

SGI



MIT

Outline

Acquisition

Processing ←

Representation

Polygonal Mesh Processing

Analysis
- Normals
- Curvature

Warps
- Rotate
- Deform

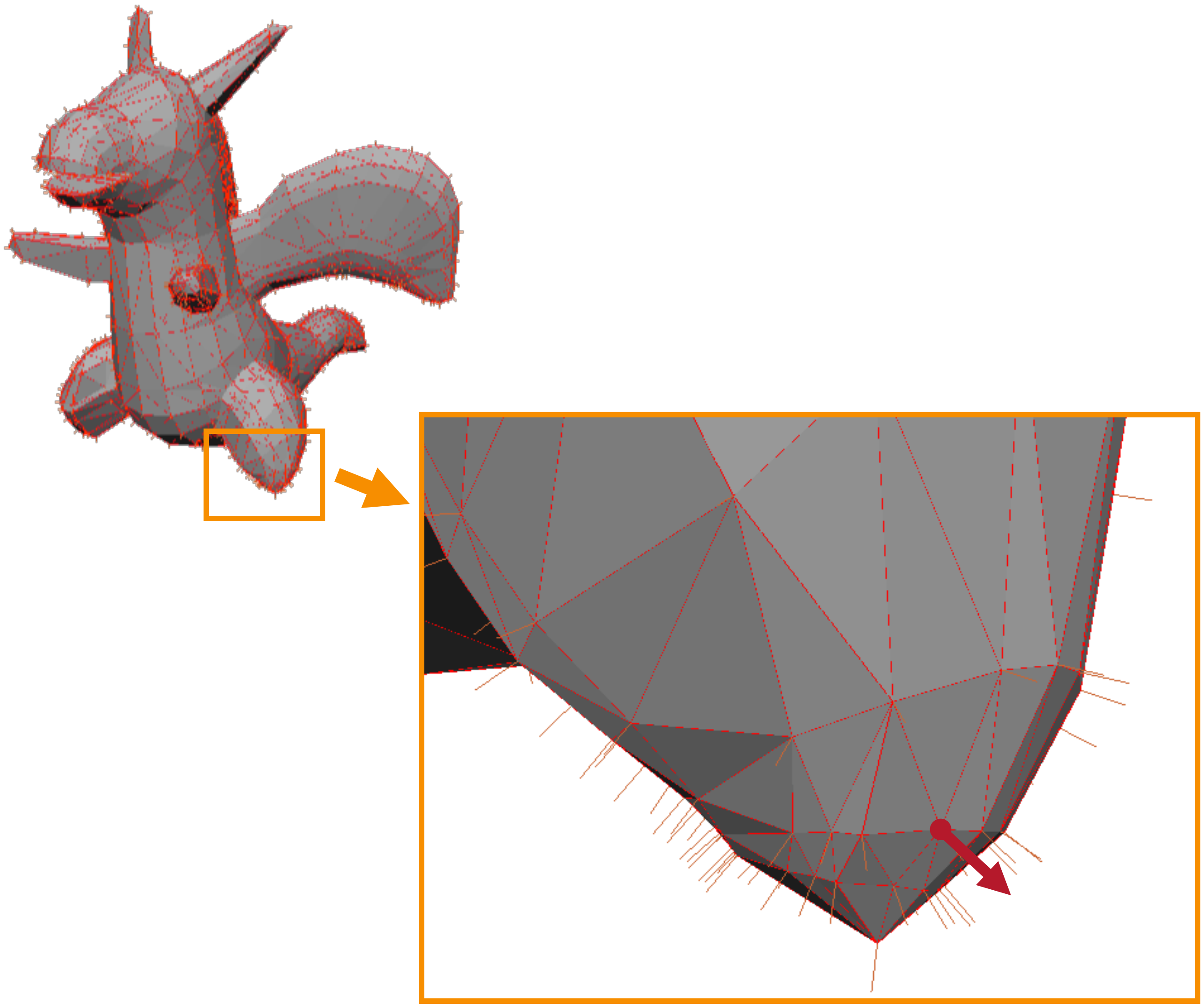
Filters
- Smooth
- Sharpen
- Truncate
- Bevel

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

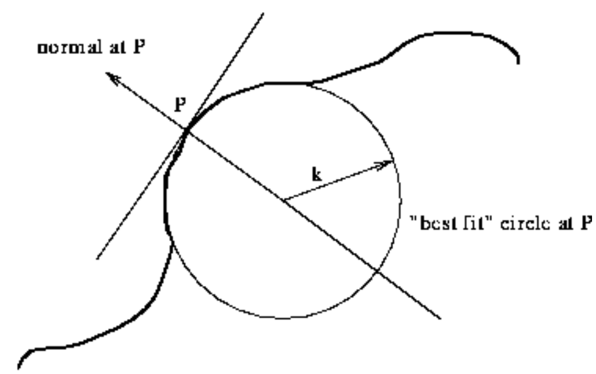


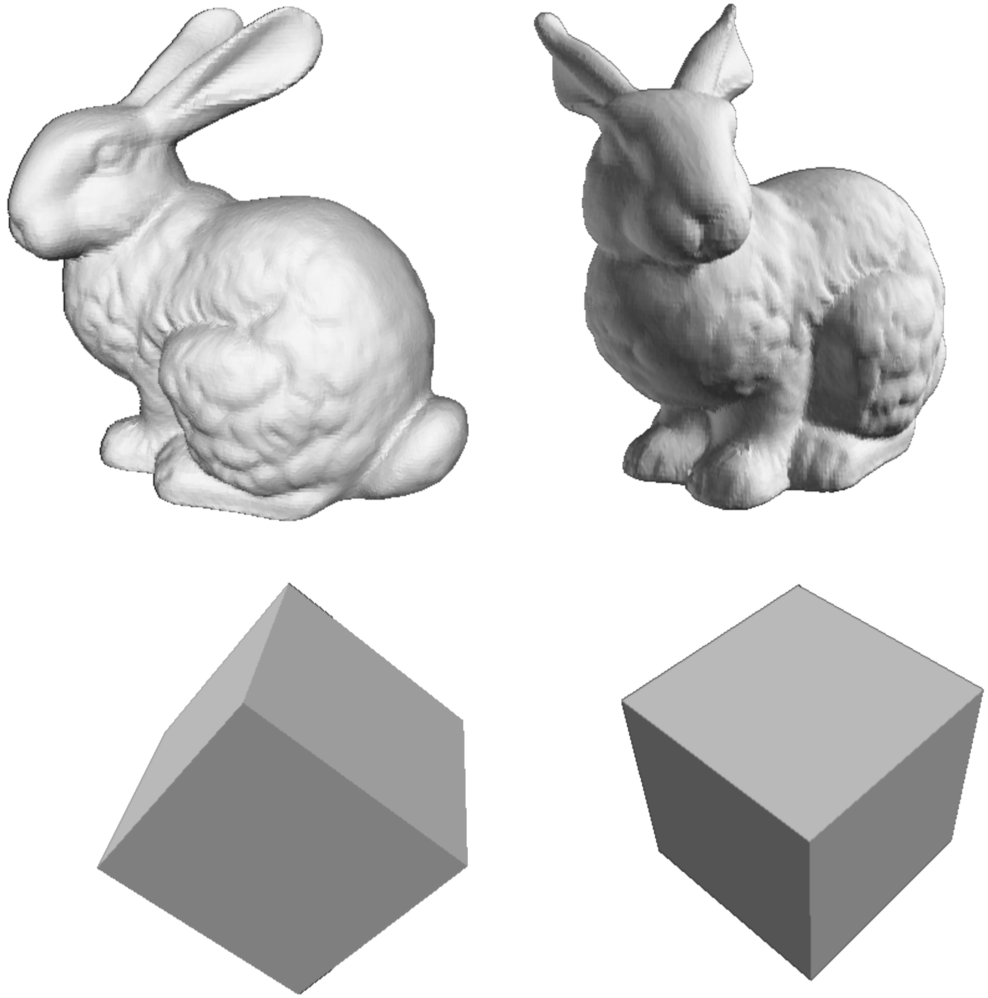
Figure 32: curvature of curve at P is $1/k$

Polygonal Mesh Processing

Analysis
- Normals
- Curvature

Warps
- ➢ Rotate
- Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel

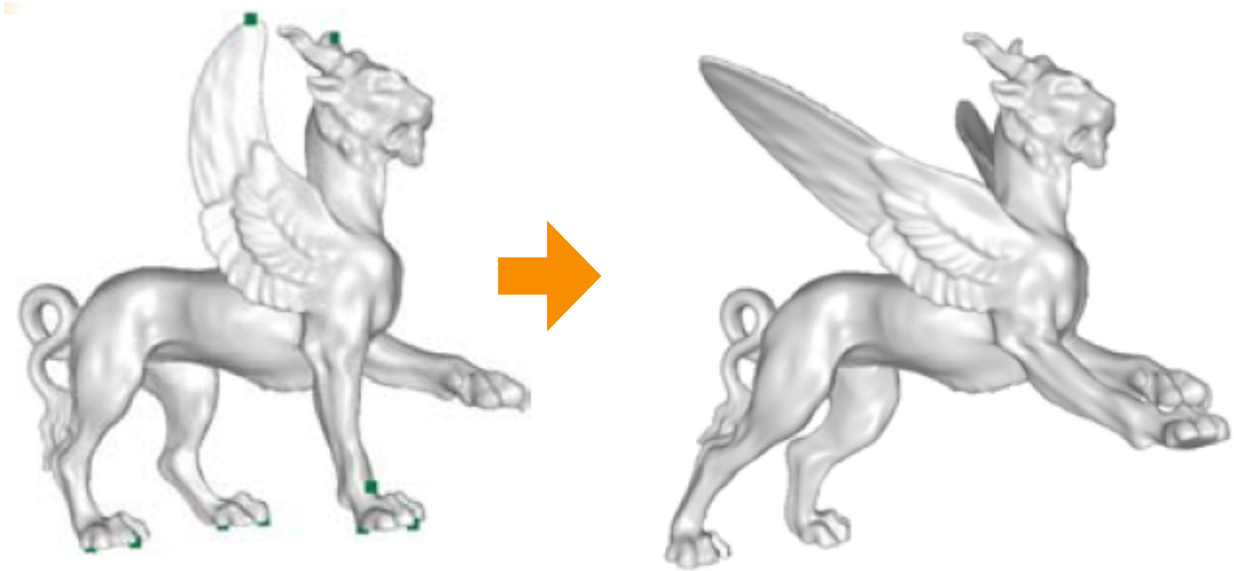


Polygonal Mesh Processing

Analysis
- Normals
- Curvature

Warps
- Rotate
- ➢ Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel



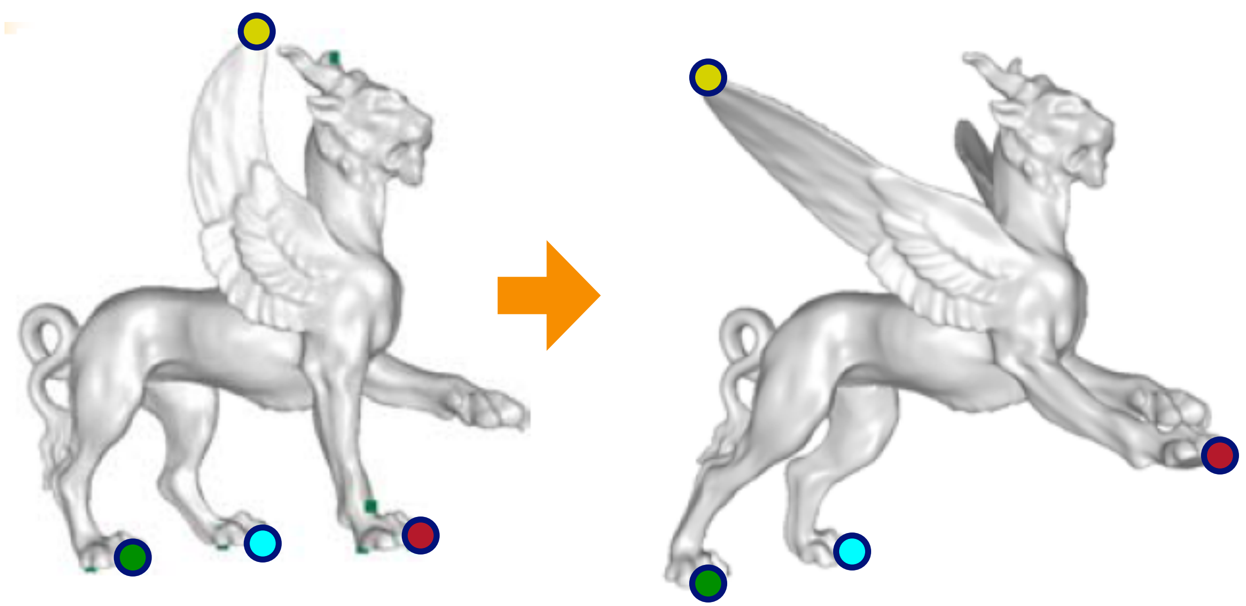
Sheffer

Polygonal Mesh Processing

Analysis
- Normals
- Curvature

Warps
- Rotate
- ➢ Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel



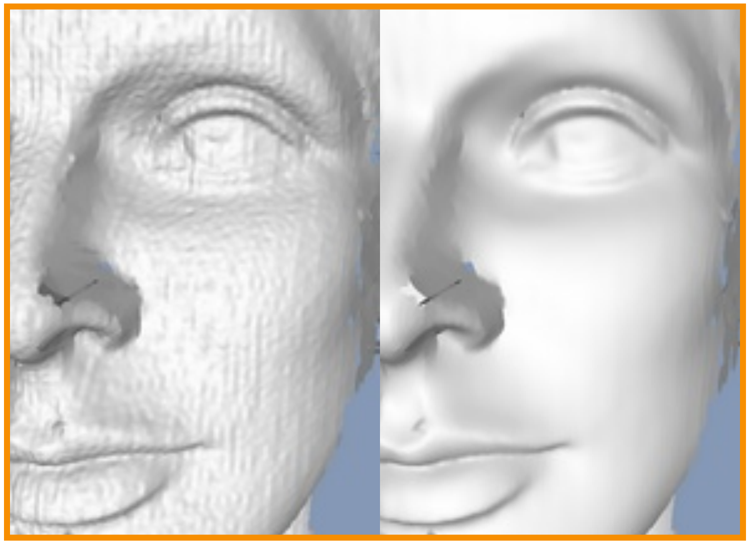
Sheffer

Polygonal Mesh Processing

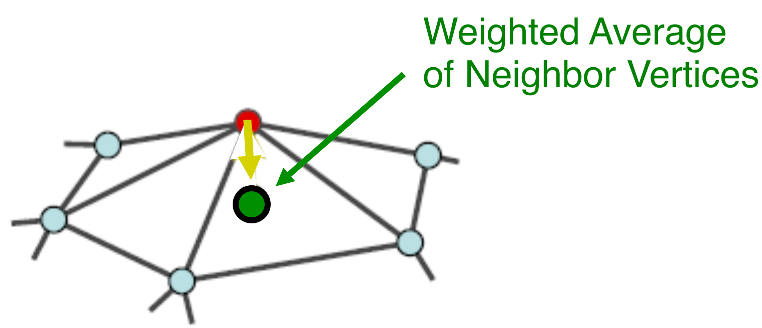
Analysis
- Normals
- Curvature

Warps
- Rotate
- Deform

Filters
- ➢ Smooth
- Sharpen
- Truncate
- Bevel



Thouis "Ray" Jones



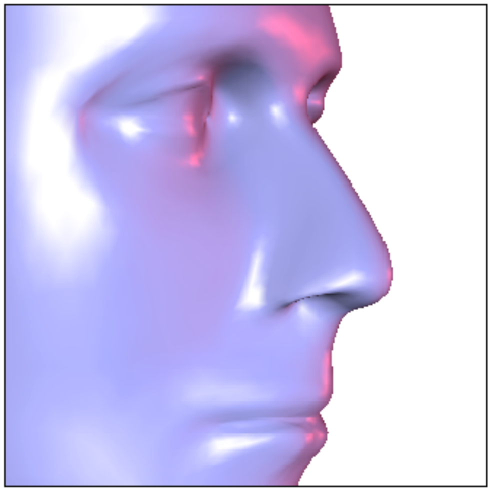
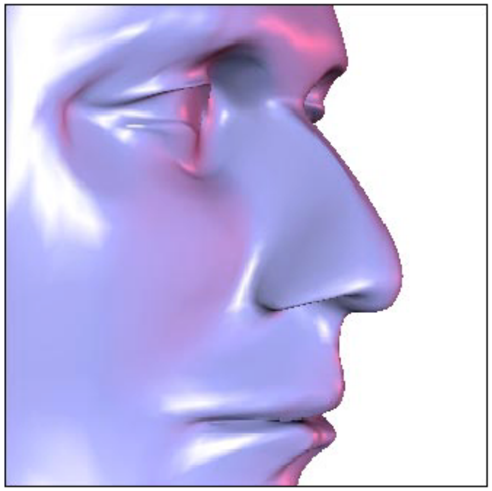
Olga Sorkine

Polygonal Mesh Processing

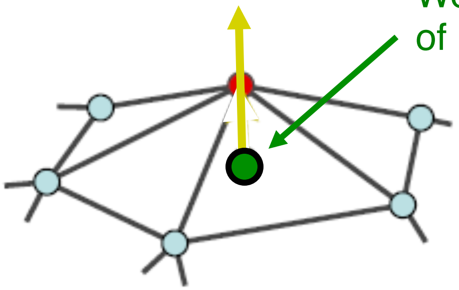
Analysis
- Normals
- Curvature

Warps
- Rotate
- Deform

Filters
- Smooth
- ➢ Sharpen
- Truncate
- Bevel

Desbrun



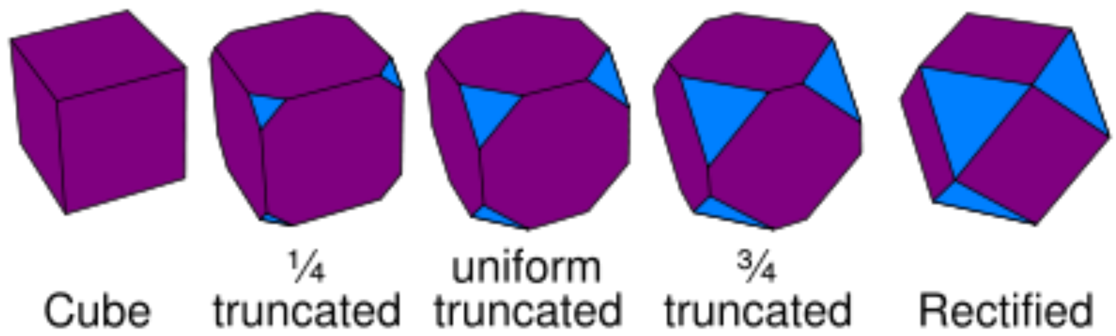
Olga Sorkine

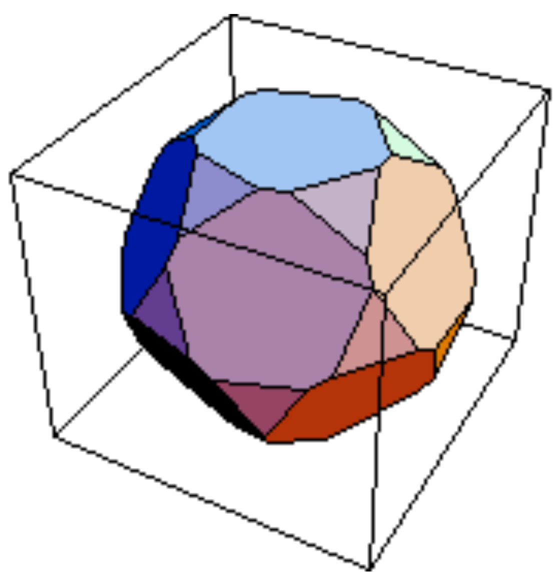
Polygonal Mesh Processing

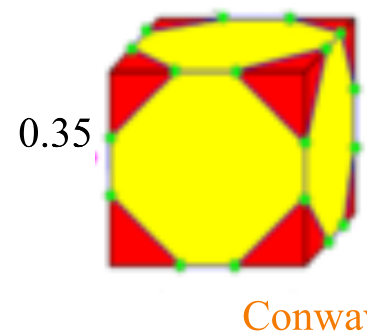
Analysis
- Normals
- Curvature

Warps
- Rotate
- Deform

Filters
- Smooth
- Sharpen
- ➢ Truncate
- Bevel







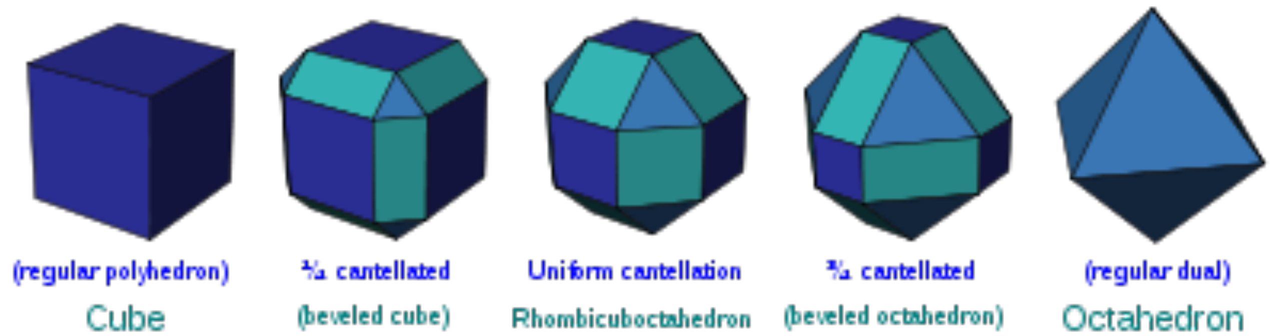
Polygonal Mesh Processing

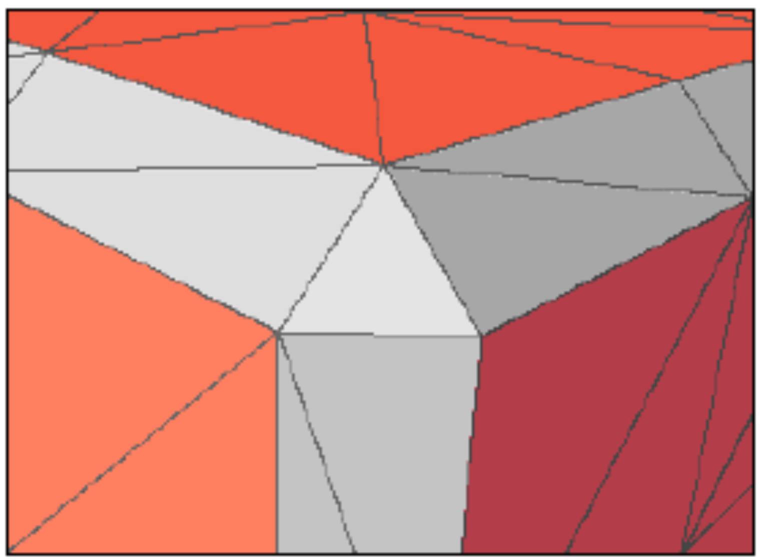
Analysis
- Normals
- Curvature

Warps
- Rotate
- Deform

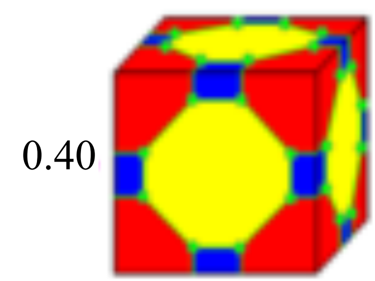
Filters
- Smooth
- Sharpen
- Truncate
- ➢ Bevel

Wikipedia





Jarek Rossignac



Conway

Polygonal Mesh Processing

Remeshing
- Subdivide
- Resample
- Simplify

Topological fixup
- Fill holes
- Fix cracks
- Fix self-intersections

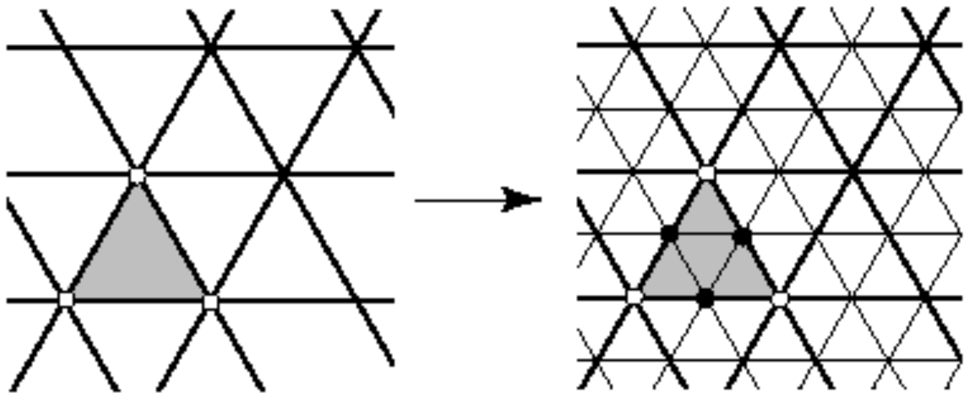
Boolean operations
- Crop
- Subtract

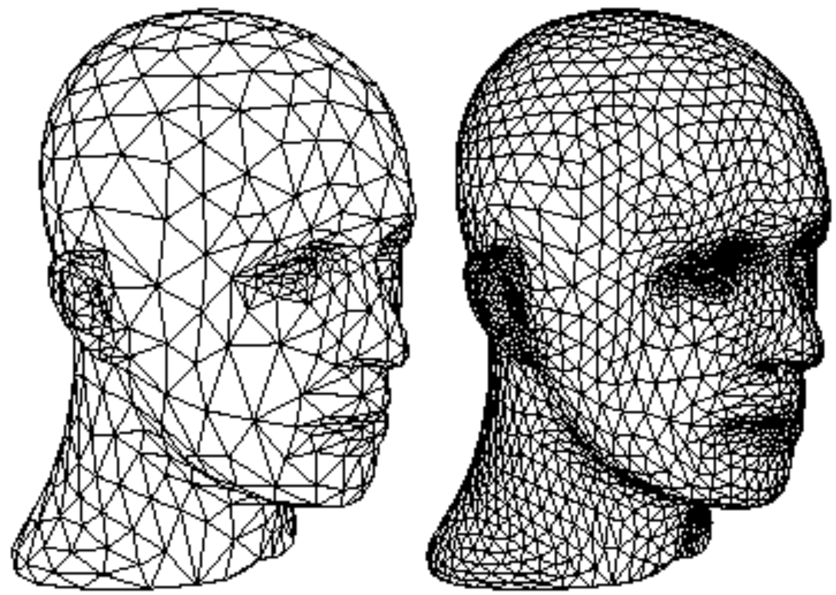
Polygonal Mesh Processing

Remeshing
- ➢ Subdivide
- ◦ Resample
- ◦ Simplify

Topological fixup
- ◦ Fill holes
- ◦ Fix cracks
- ◦ Fix self-intersections

Boolean operations
- ◦ Crop
- ◦ Subtract





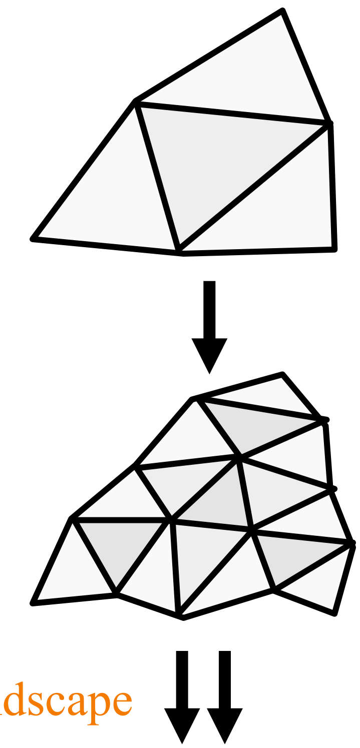
Zorin & Schroeder

Polygonal Mesh Processing

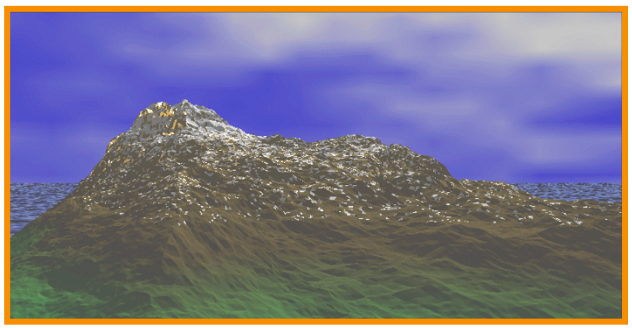
Remeshing
- Subdivide
- Resample
- Simplify

Topological fixup
- Fill holes
- Fix cracks
- Fix self-intersections

Boolean operations
- Crop
- Subtract



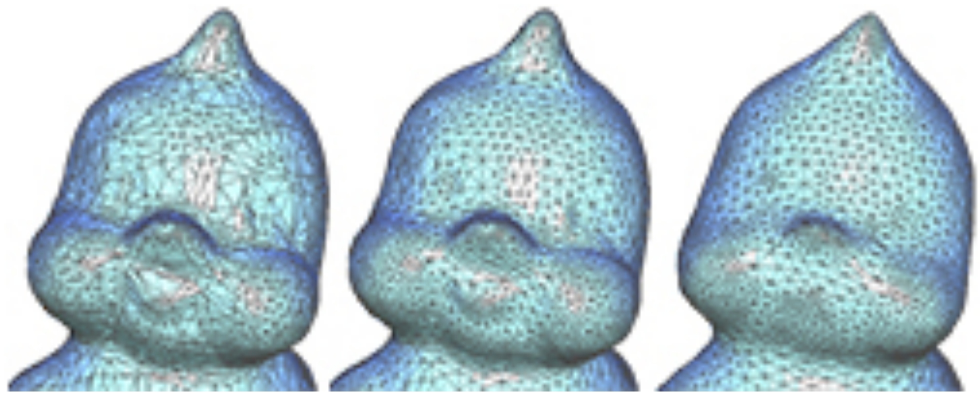
Fractal Landscape



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

Polygonal Mesh Processing

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



Original   Resampled

- more uniform distribution
- triangles with nicer aspect

Sorkine

Polygonal Mesh Processing

Remeshing
- Subdivide
- Resample
- Simplify

Topological fixup
- Fill holes
- Fix cracks
- Fix self-intersections

Boolean operations
- Crop
- Subtract

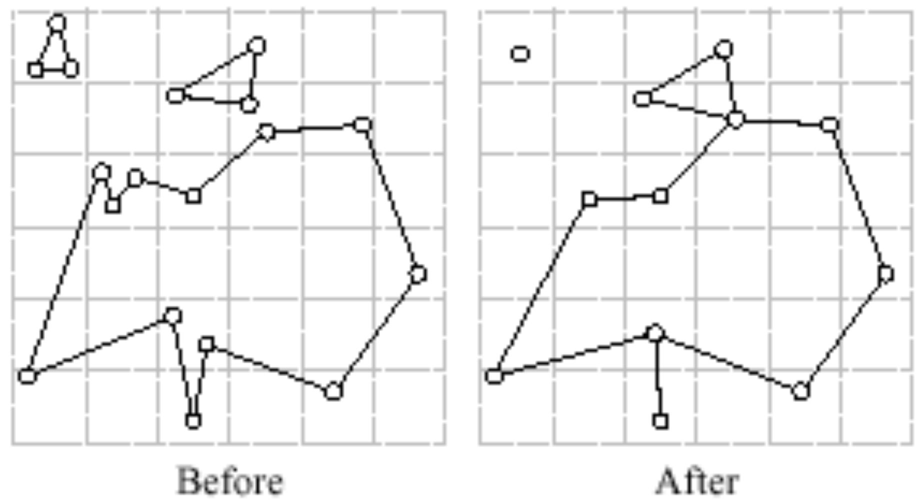
Garland

Polygonal Mesh Processing

Remeshing
- Subdivide
- Resample
- Simplify

Topological fixup
- ➤ Fill holes
- Fix cracks
- Fix self-intersections

Boolean operations
- Crop
- Subtract



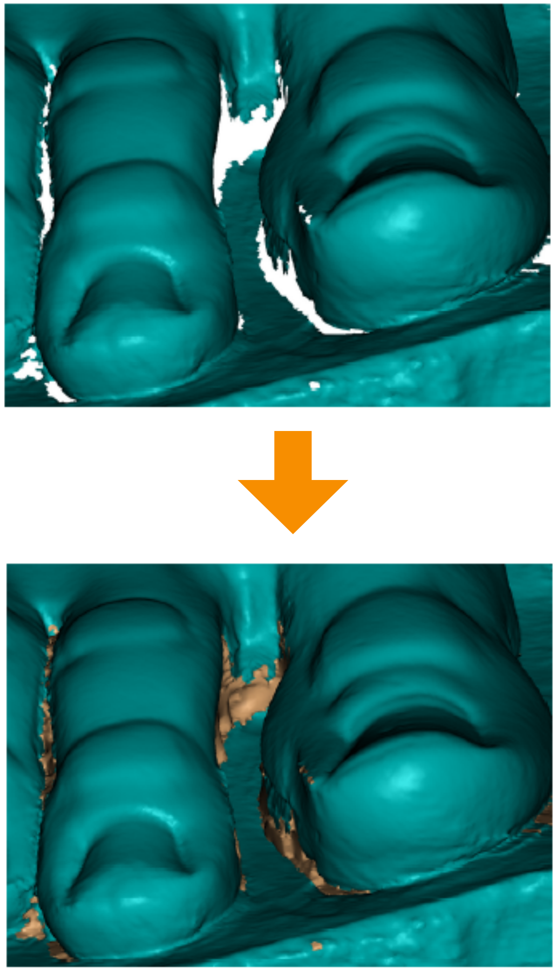
Vertex Clustering

Polygonal Mesh Processing

Remeshing
- Subdivide
- Resample
- Simplify

Topological fixup
- Fill holes
- ➢ Fix cracks
- Fix self-intersections

Boolean operations
- Crop
- Subtract



Podolak

Polygonal Mesh Processing

Remeshing

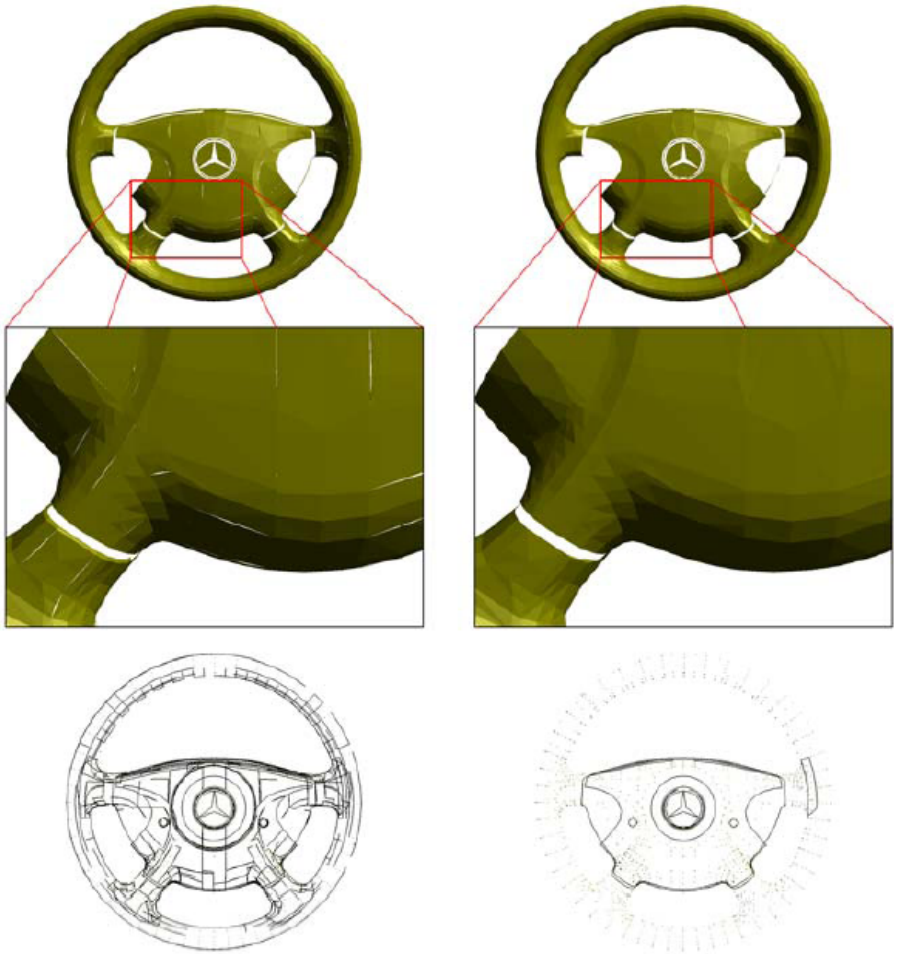
- Subdivide
- Resample
- Simplify

Topological fixup

- Fill holes
- Fix cracks
- ➢ Fix self-intersections

Boolean operations

- Crop
- Subtract



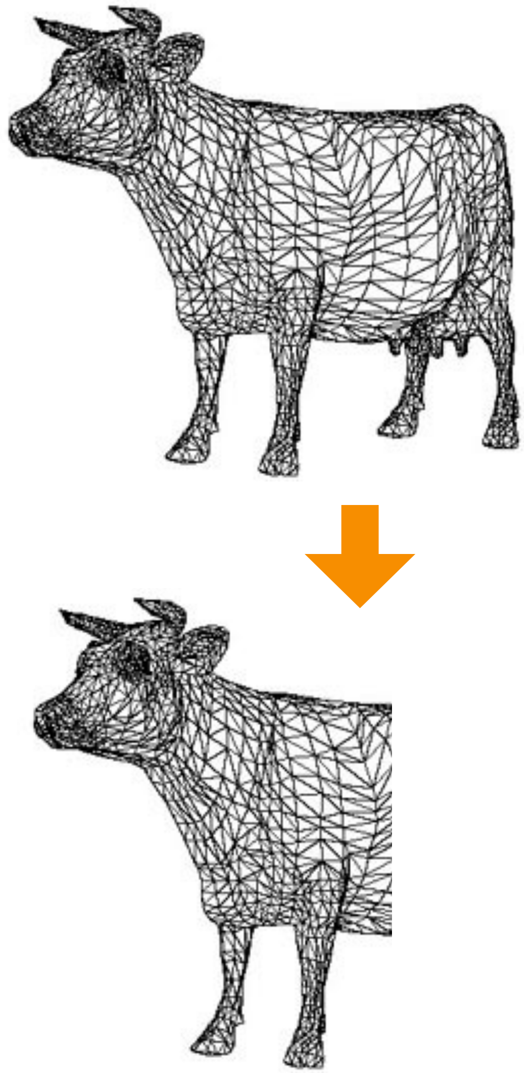
Borodin

Polygonal Mesh Processing

Remeshing
- Subdivide
- Resample
- Simplify

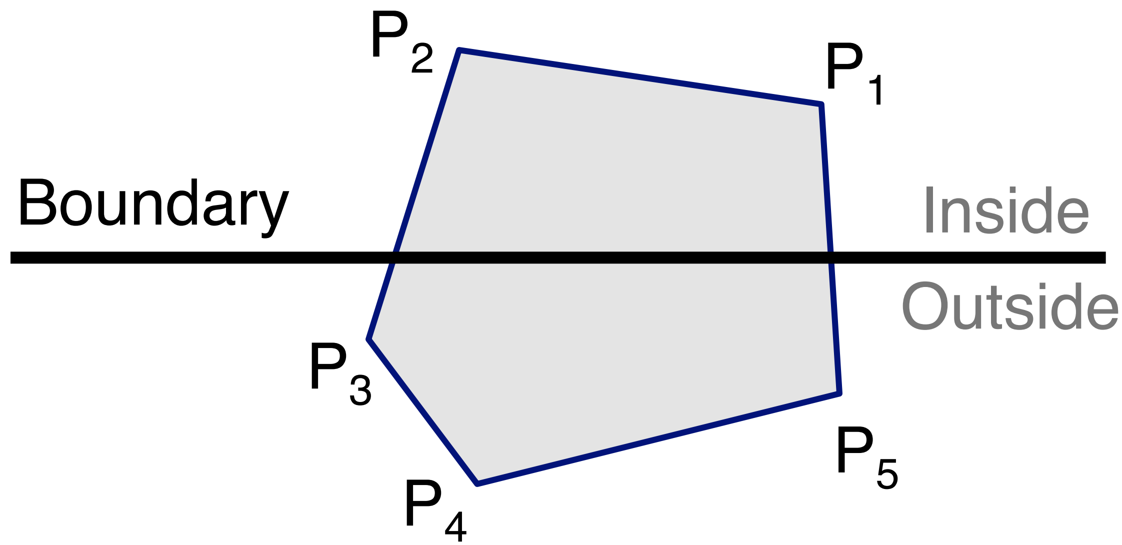
Topological fixup
- Fill holes
- Fix cracks
- Fix self-intersections

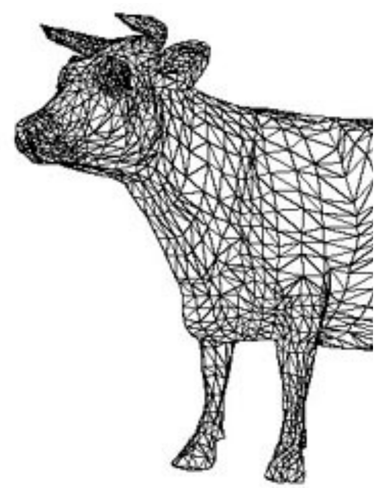
Boolean operations
- ➢ Crop
- Subtract



Polygonal Mesh Processing

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



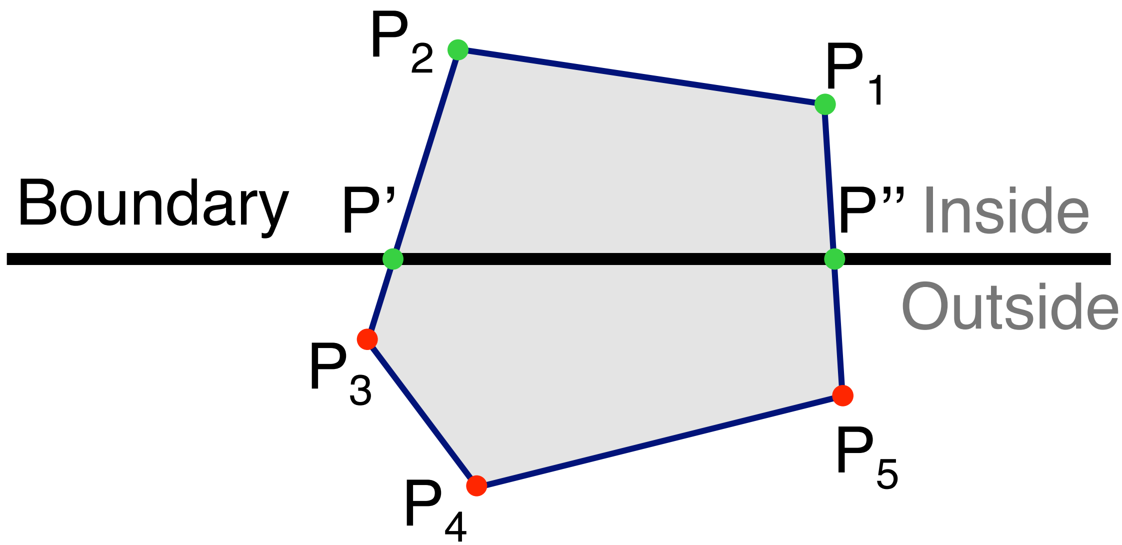


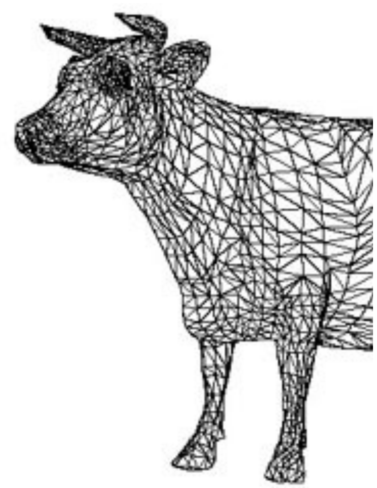
Polygonal Mesh Processing

Remeshing
- Subdivide
- Resample
- Simplify

Topological fixup
- Fill holes
- Fix cracks
- Fix self-intersections

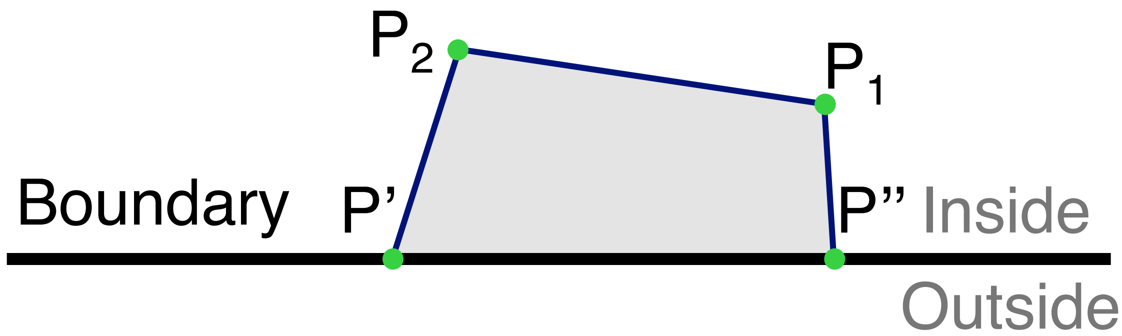
Boolean operations
- ➢ Crop
- Subtract

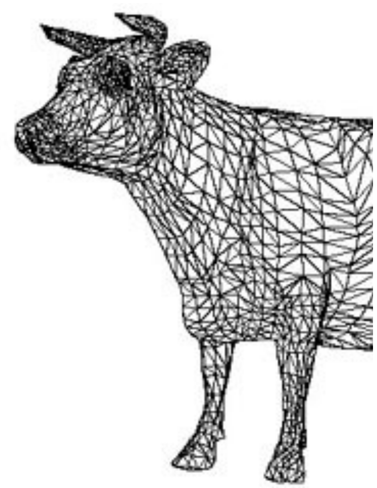




Polygonal Mesh Processing

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



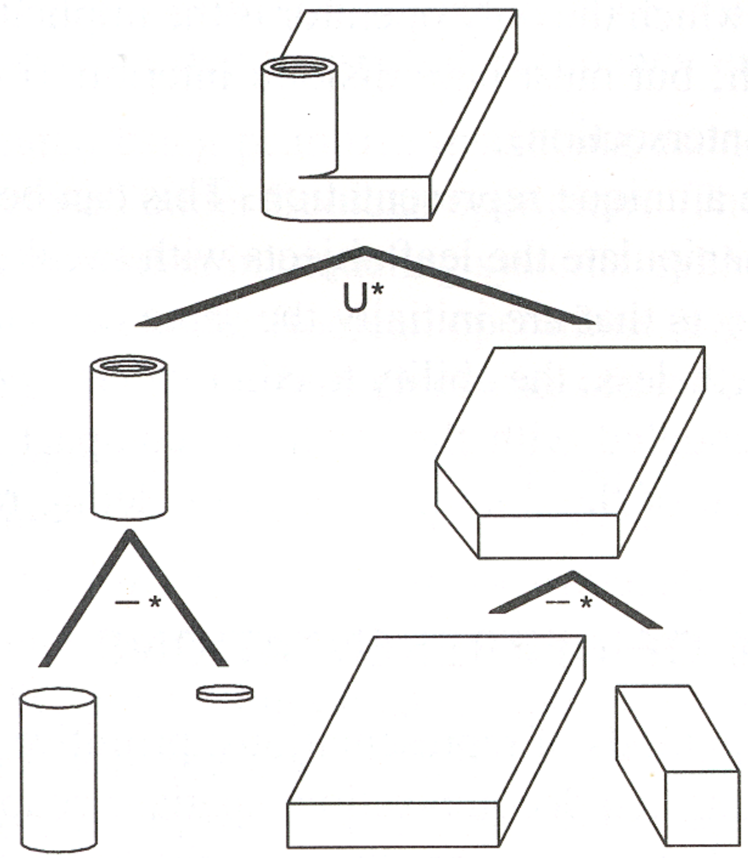


Polygonal Mesh Processing

Remeshing
- Subdivide
- Resample
- Simplify

Topological fixup
- Fill holes
- Fix cracks
- Fix self-intersections

Boolean operations
- Crop
- ➢ Subtract



FvDFH Figure 12.27

Polygonal Mesh Processing

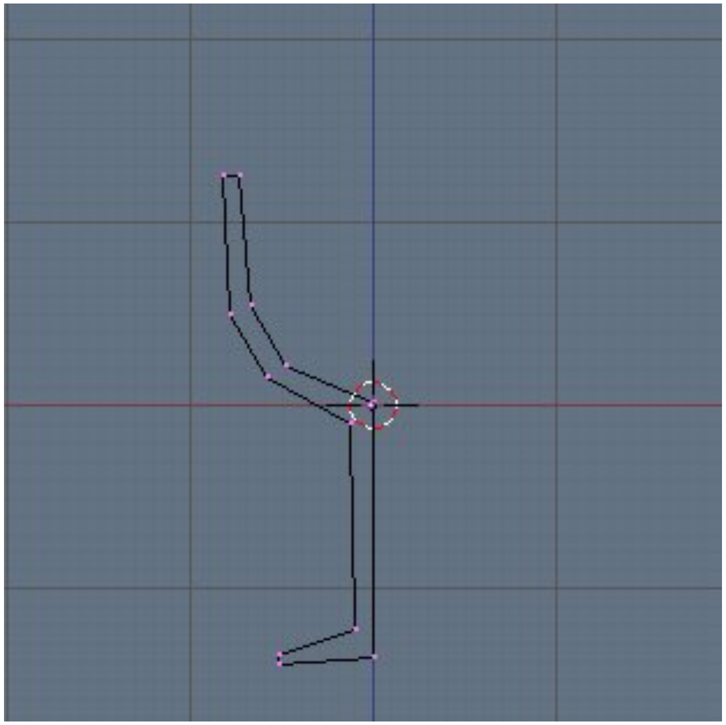
Procedural generation

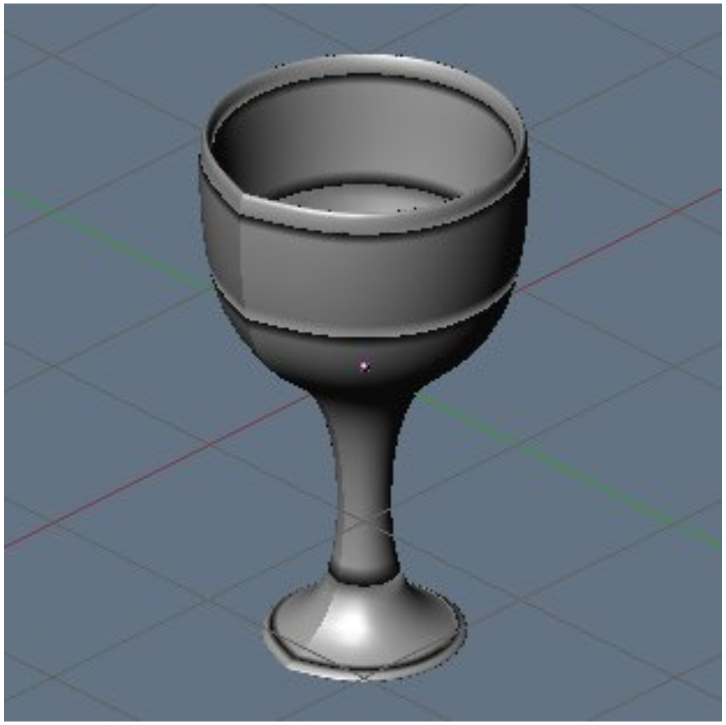
- Surface of revolution
- Sweep

Polygonal Mesh Processing

Procedural generation

- ➢ Surface of revolution
- ◦ Sweep



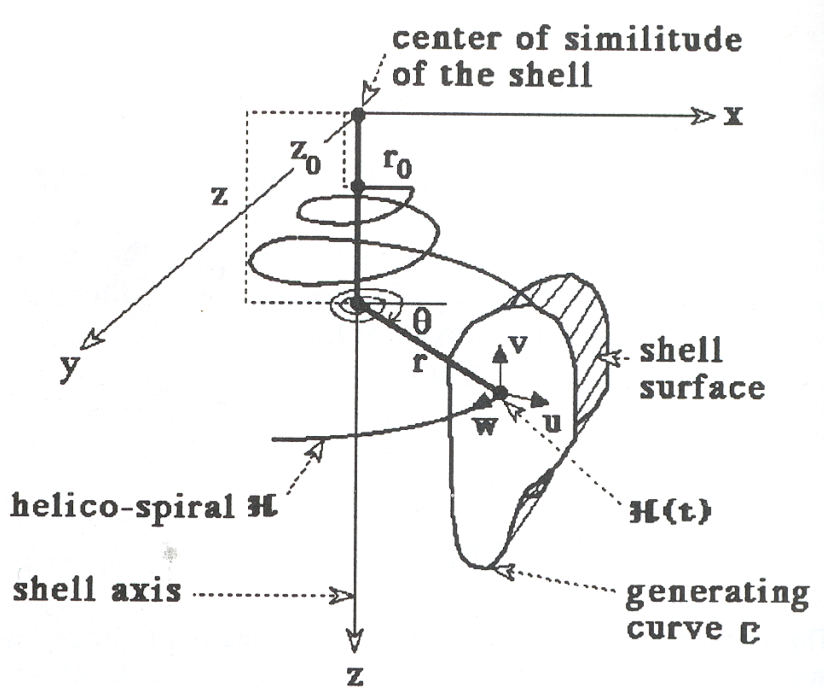


sphynx.co.uk

Polygonal Mesh Processing

Procedural generation

- Surface of revolution
- ➢ Sweep



Fowler et al., 1992

Polygonal Mesh Processing

Procedural generation

- Surface of revolution
- ➢ Sweep

Fowler et al., 1992

Polygonal Mesh Processing

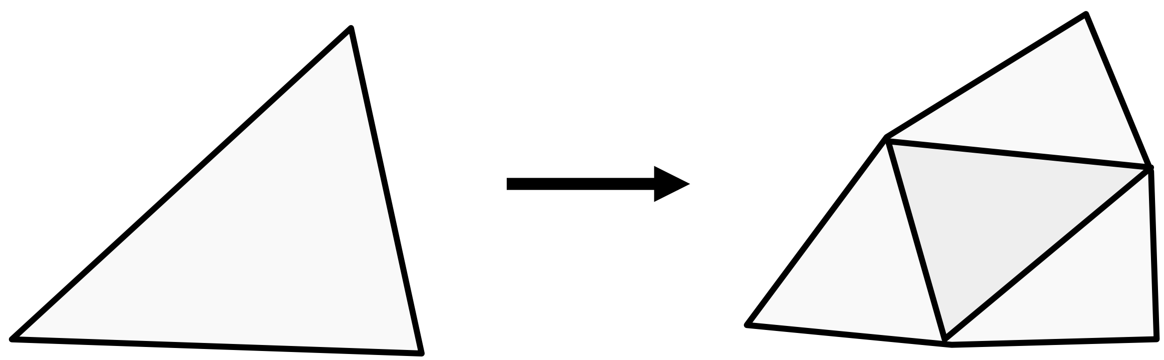
Most operations use a few low-level operations:

- Subdivide face
- Subdivide edge
- Collapse edge
- Merge vertices
- Remove vertex

Polygonal Mesh Processing

Most operations use a few low-level operations:

- Subdivide face
- Subdivide edge
- Collapse edge
- Merge vertices
- Remove vertex

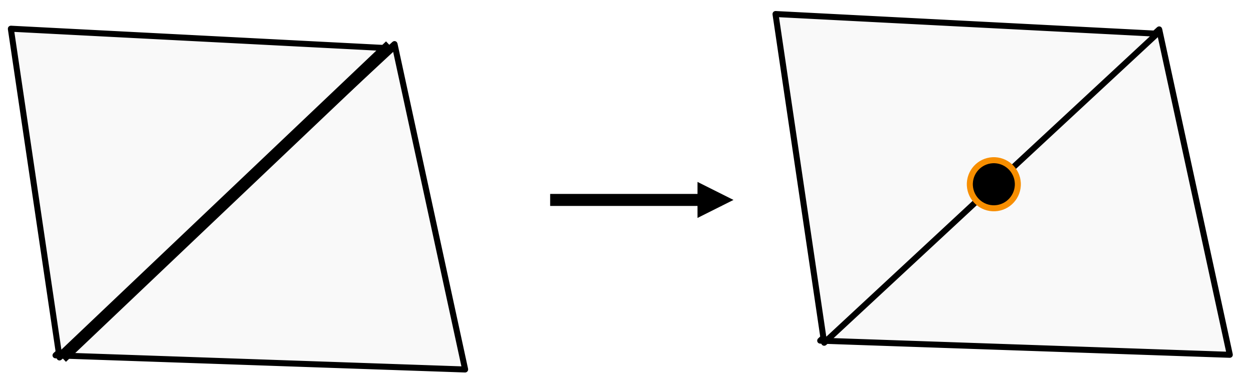


Subdivide face

Polygonal Mesh Processing

Most operations use a few low-level operations:

- Subdivide face
- Subdivide edge
- Collapse edge
- Merge vertices
- Remove vertex

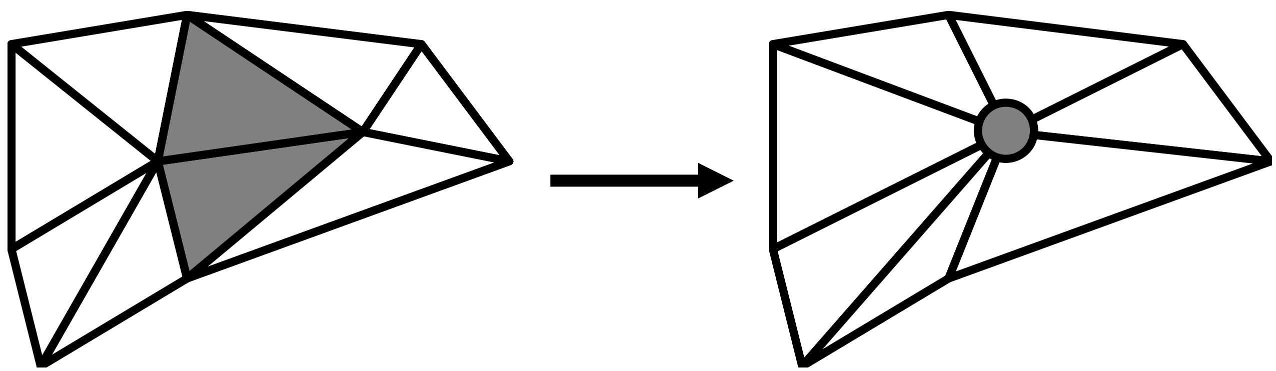


Subdivide edge

Polygonal Mesh Processing

Most operations use a few low-level operations:

- Subdivide face
- Subdivide edge
- Collapse edge
- Merge vertices
- Remove vertex

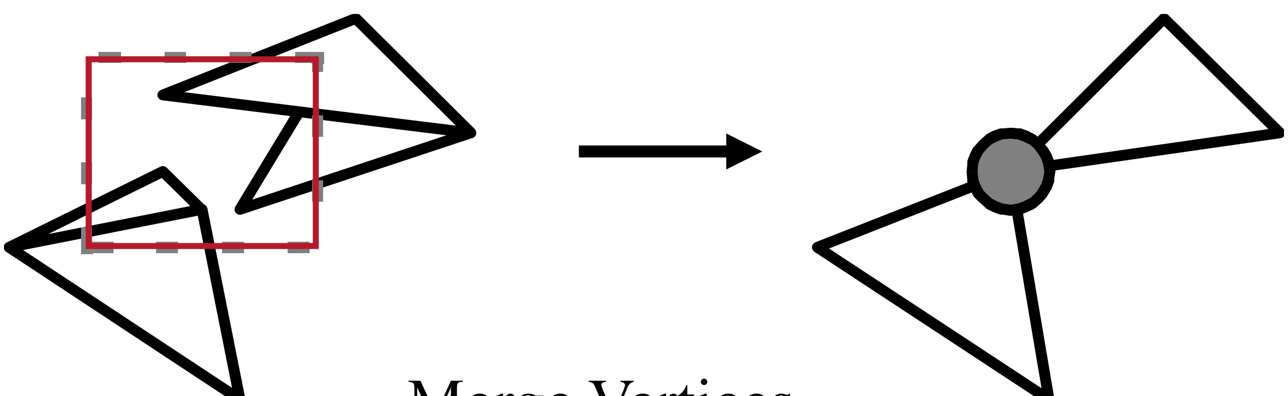


Collapse edge

Polygonal Mesh Processing

Most operations use a few low-level operations:

- Subdivide face
- Subdivide edge
- Collapse edge
- Merge vertices
- Remove vertex

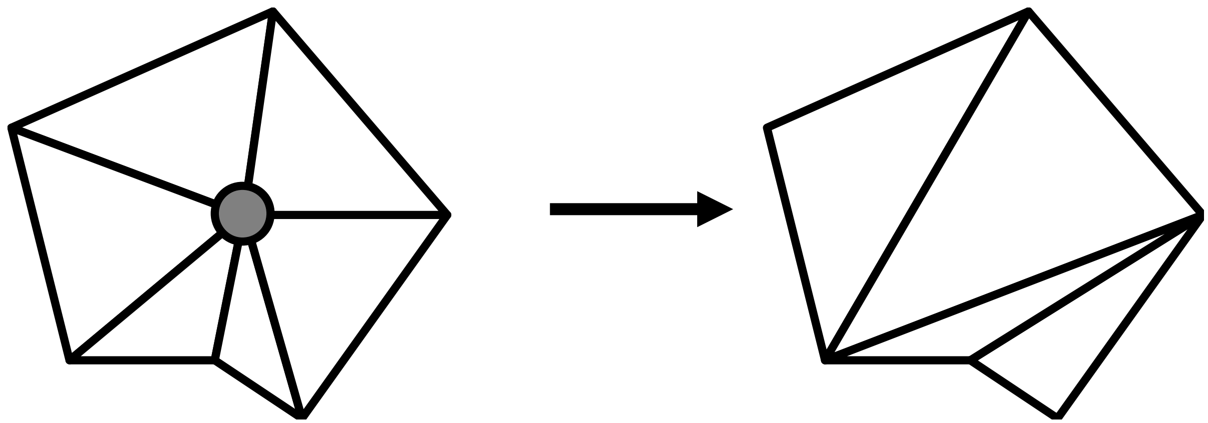


Merge Vertices

Polygonal Mesh Processing

Most operations use a few low-level operations:

- Subdivide face
- Subdivide edge
- Collapse edge
- Merge vertices
- Remove vertex



Remove Vertex

Outline

Acquisition

Processing

Representation ←

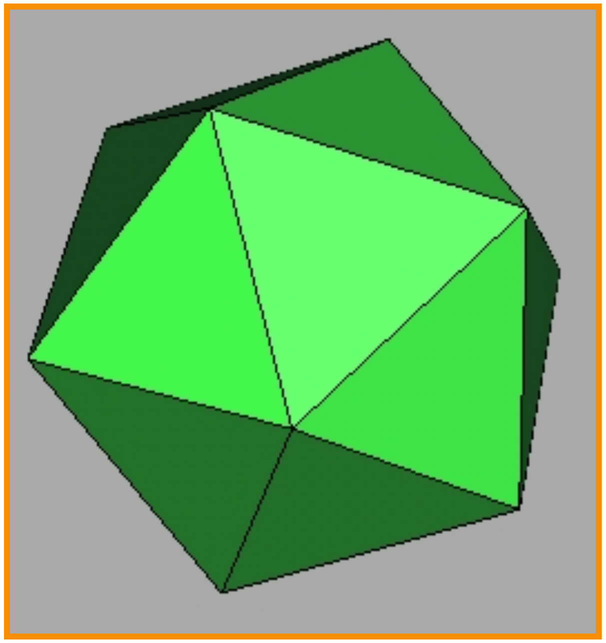
Polygon Mesh Representation

Data structures determine algorithms

- Data structure must support key operations of algorithm efficiently

Examples:

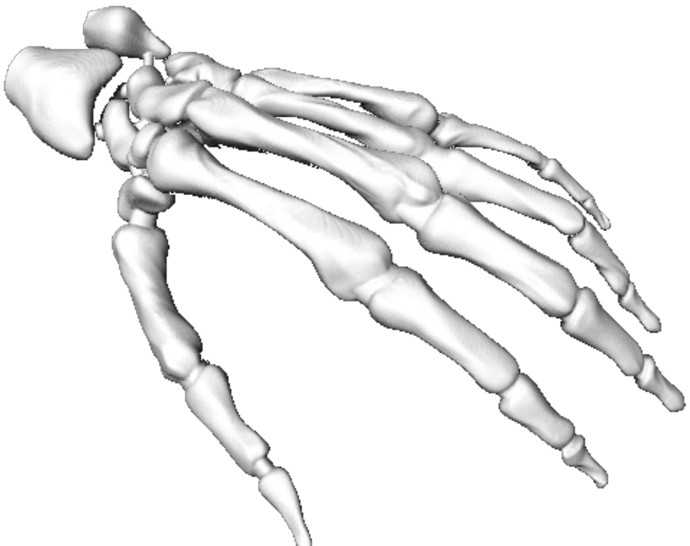
- Drawing a mesh
- Removing a vertex
- Smoothing a region
- Intersecting polyhedra

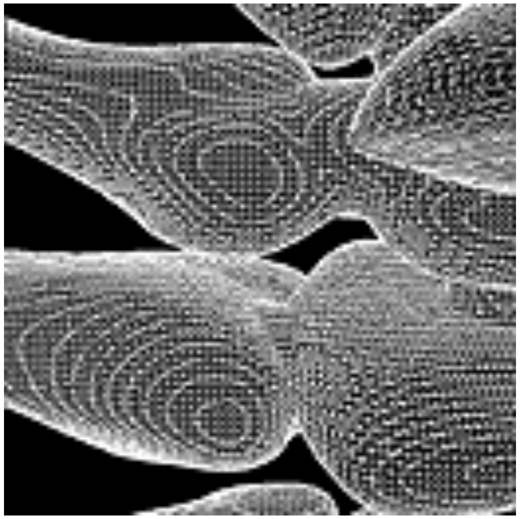


Different data structures for different algorithms

Polygon Mesh Representation

Important properties of mesh representation?



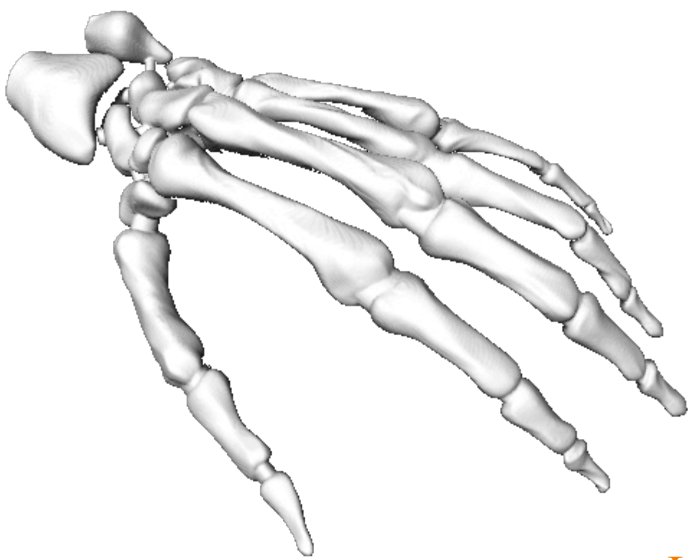


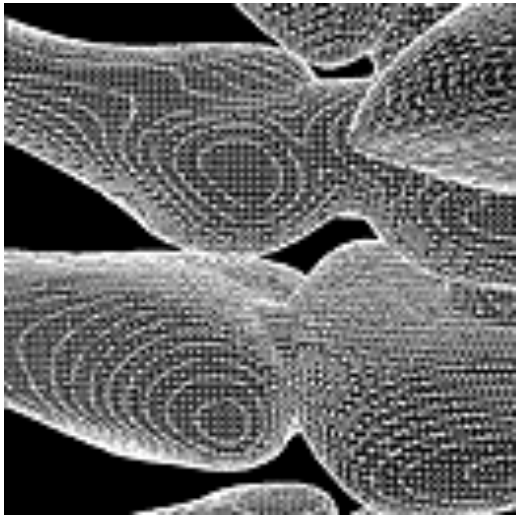
Large Geometric Model Repository
Georgia Tech

Polygon Mesh Representation

Important properties of mesh representation?

- Efficient traversal of topology
- Efficient use of memory
- Efficient updates



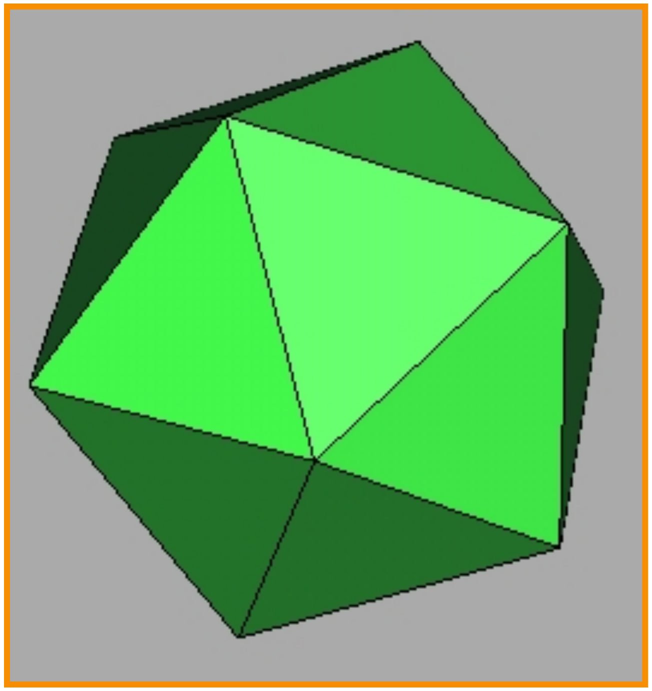


Large Geometric Model Repository
Georgia Tech

Polygon Mesh Representation

Possible data structures

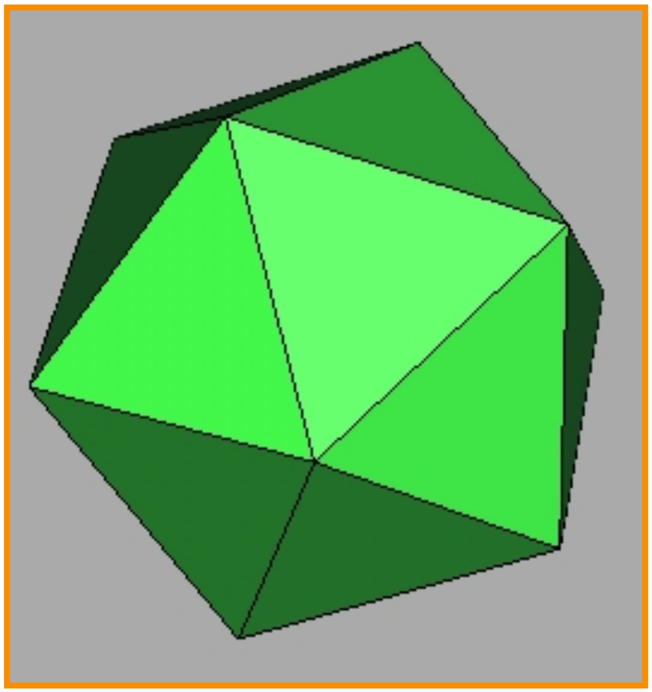
- List of independent faces
- Vertex and face tables
- Adjacency lists
- Winged edge
- Half edge
- etc.

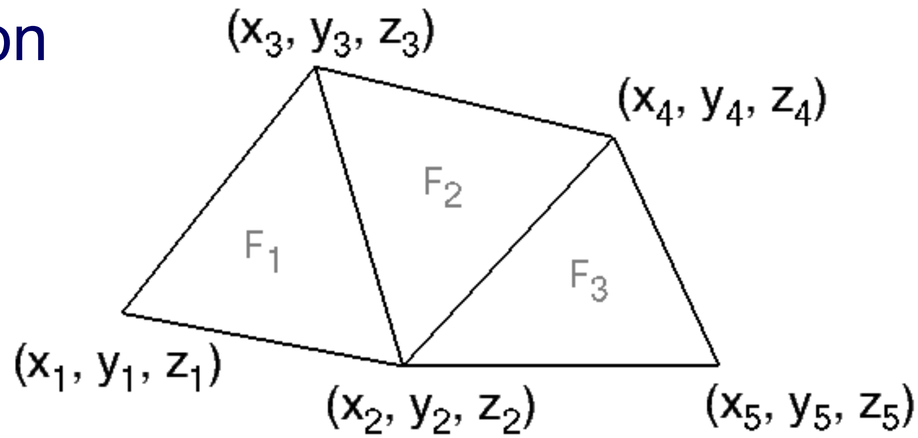


Independent Faces

Each face lists vertex coordinates

- Redundant vertices
- No adjacency information



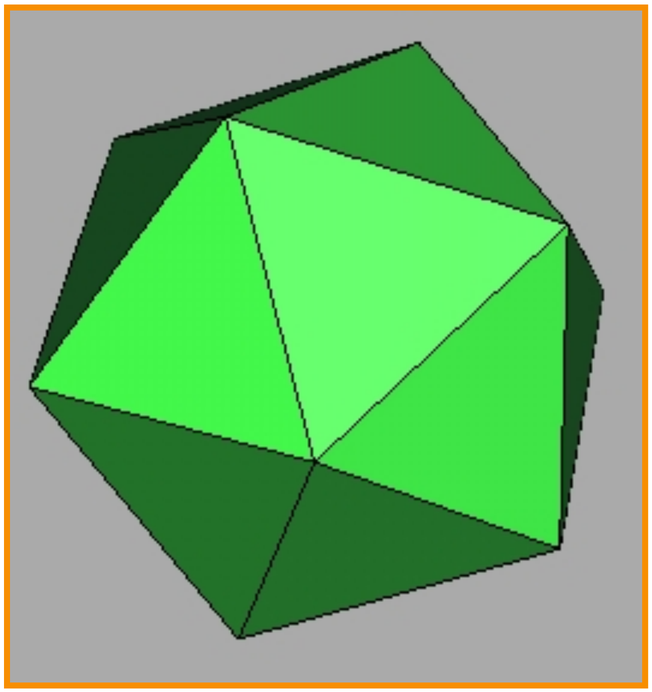


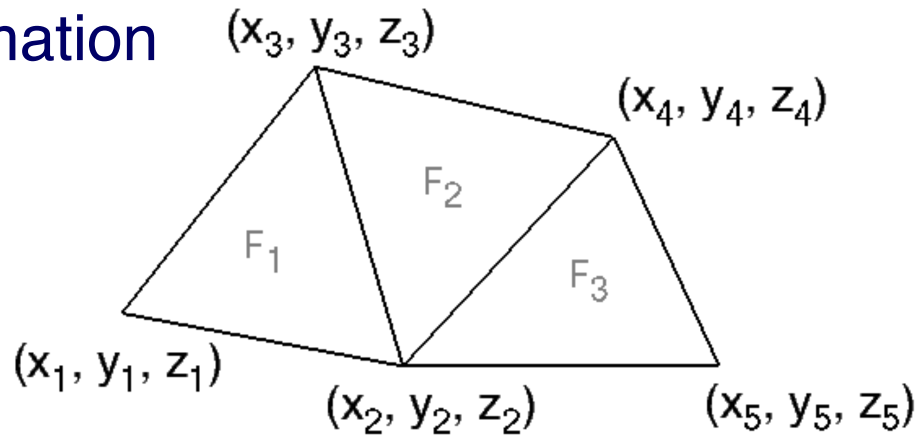
FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables

Each face lists vertex references

- Shared vertices
- Still no adjacency information





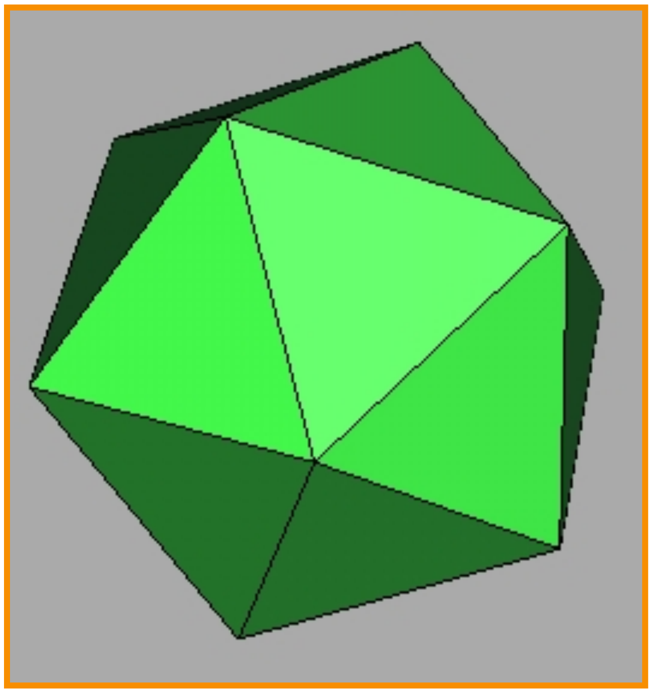
VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

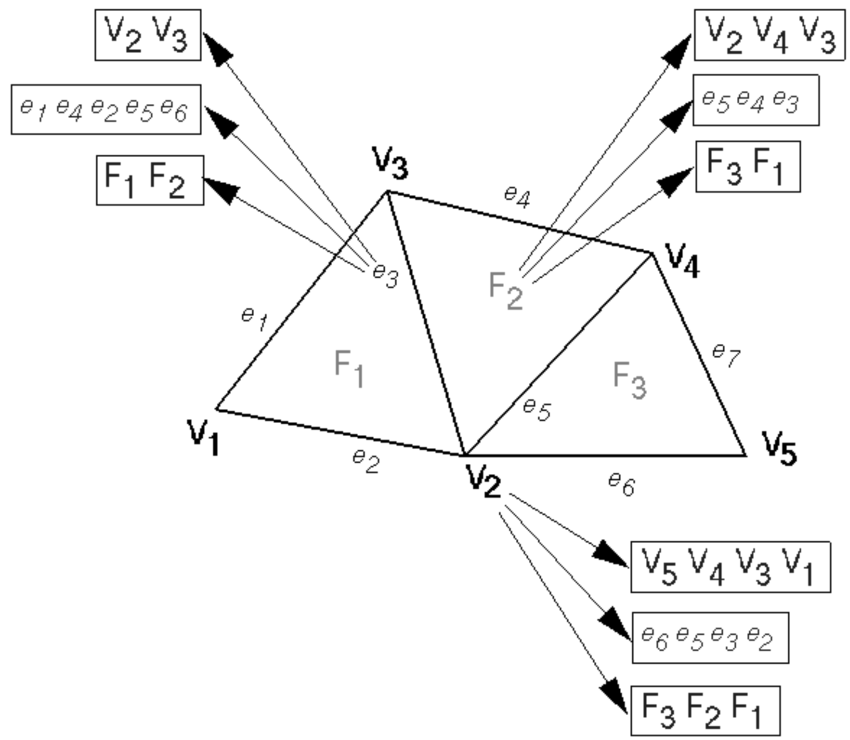
FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

Adjacency Lists

Store all vertex, edge, and face adjacencies

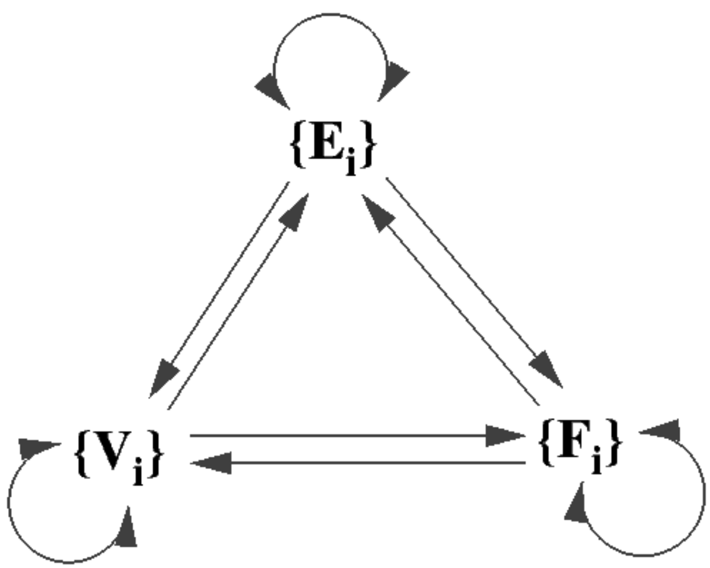
- Efficient adjacency traversal
- Extra storage

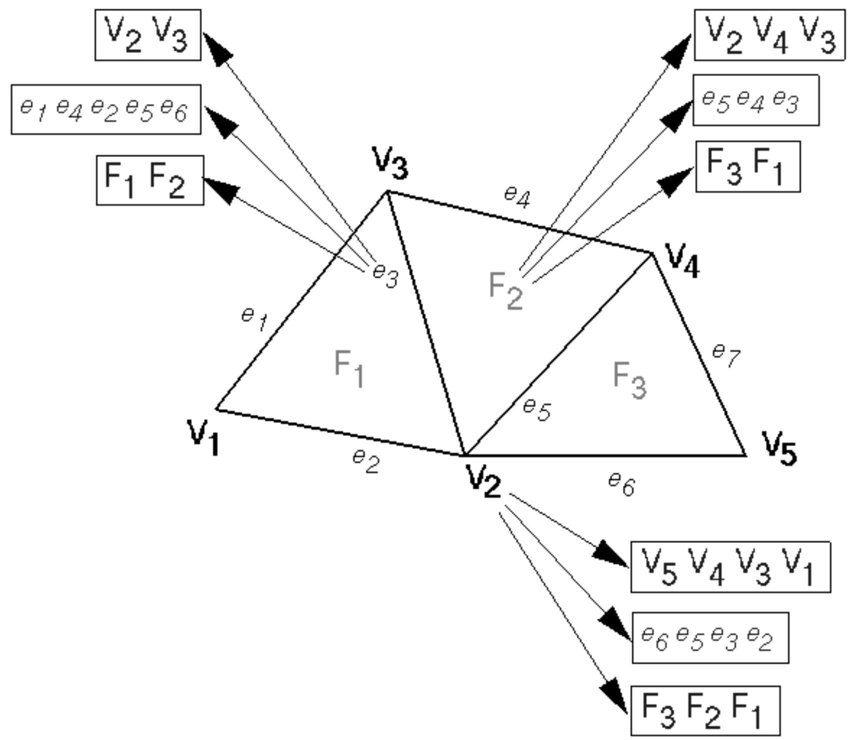




Partial Adjacency Lists

Can we store only some adjacency relationships and derive others?

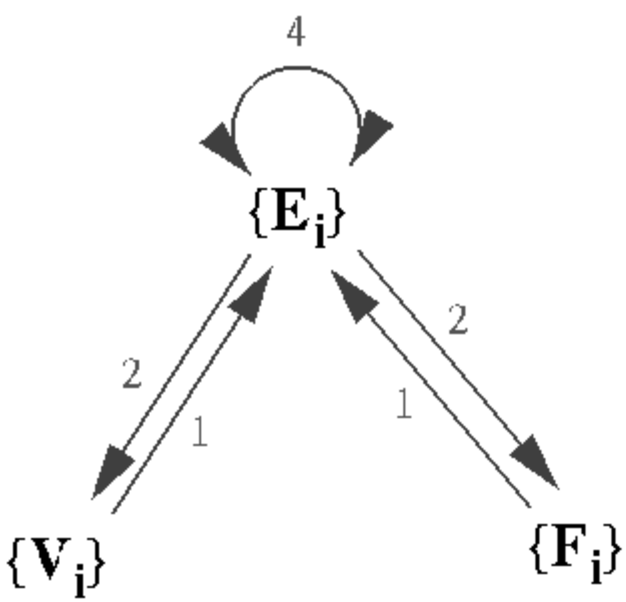


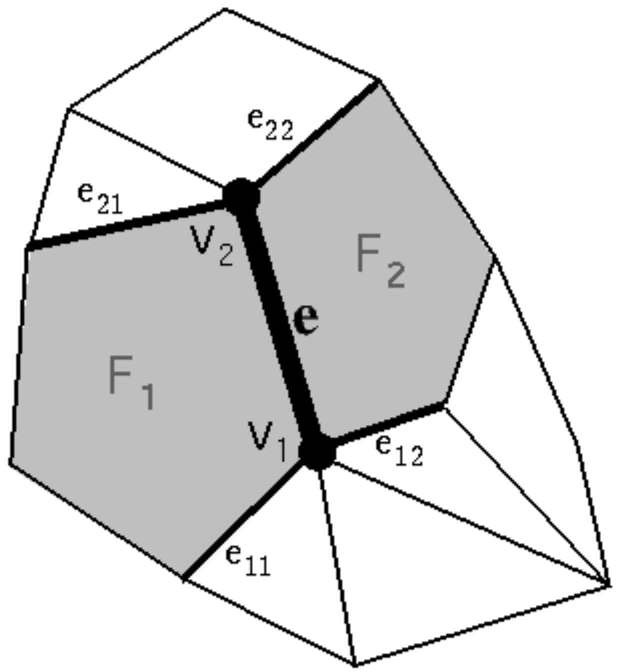


Winged Edge

Adjacency encoded in edges

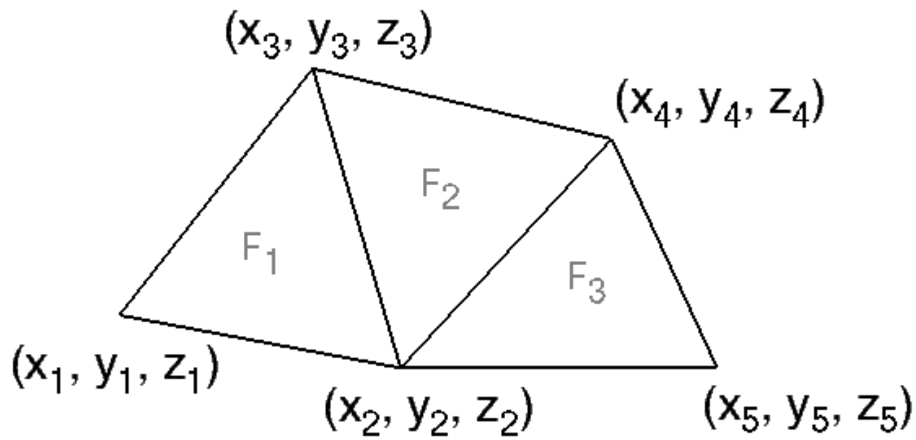
- All adjacencies in O(1) time
- Little extra storage (fixed records)
- Arbitrary polygons





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

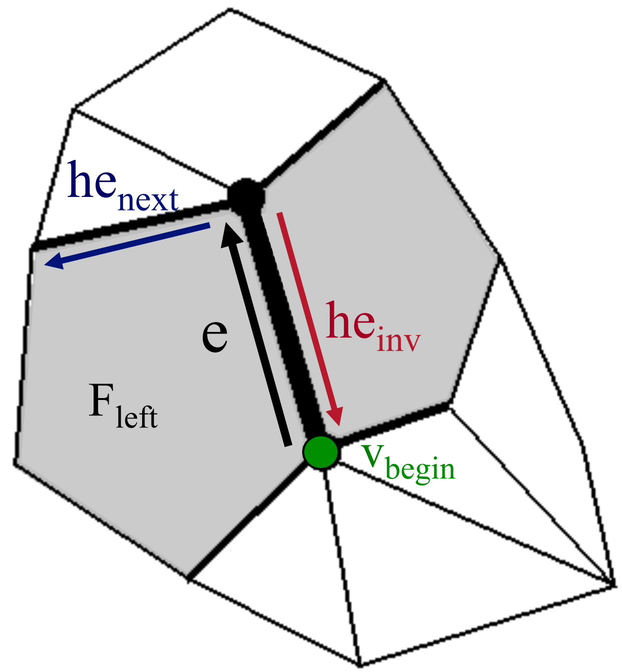
F_1	e_1
F_2	e_3
F_3	e_5

Half Edge

Adjacency encoded in edges

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

Similar to winged-edge, except adjacency encoded in half-edges



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