Polygonal Meshes

COS 426
# 3D Object Representations

## Points
- Range image
- Point cloud

## Surfaces
- Polyhedral 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

Face
Edge
Vertex (x,y,z)
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

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

\((x,y,z)\)
3D Vector

Specifies a direction and a magnitude

- Represented by three coordinates
- Magnitude $\|V\| = \sqrt{dx^2 + dy^2 + dz^2}$
- Has no location

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

$(dx, dy, dz)$
Scalar / dot product of two 3D vectors

\[ V_1 \cdot V_2 = dx_1 \cdot dx_2 + dy_1 \cdot dy_2 + dz_1 \cdot dz_2 = \|V_1\| \|V_2\| \cos(\Theta) \]
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 \( ||V_1|| \cdot ||V_2|| \sin(\Theta) \)
3D Line Segment

Linear path between two points

- Parametric representation:
  \[ p = P_1 + t (P_2 - P_1), \quad (0 \leq t \leq 1) \]

```cpp
struct Segment {
    Point P1;
    Point P2;
};
```
3D Ray

Line segment with one endpoint at infinity

- Parametric representation:
  \[ p = P_1 + t \mathbf{V}, \quad (0 \leq t < \infty) \]

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

![Diagram of 3D Ray with parametric representation and structure definition]
3D Line

Line segment with both endpoints at infinity

- Parametric representation:
  \[ p = P_1 + t \mathbf{V}, \quad (-\infty < t < \infty) \]

```c
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
  - \( P \cdot N + d = 0 \)

- \( N \) is the plane normal
  - Unit-length
  - Perpendicular to plane

```c
struct Plane {
    Vector N;
    float d;
};
```
Region “inside” a sequence of coplanar points

```cpp
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

- Triceratops
  - $v = 2832$
  - $p = 2834$

- Galleon
  - $v = 2372$
  - $p = 2384$

- Cessna
  - $v = 3745$
  - $p = 3927$

- Beethoven
  - $v = 2655$
  - $p = 2812$

- Sandal
  - $v = 2636$
  - $p = 2953$

- Cow
  - $v = 2904$
  - $p = 5804$

- Cow Poly
  - $v = 2904$
  - $p = 3263$

(shown are some data points for the models, the cow is not shown, it is the same cow model, but not fully triangulated)
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
3D Polygonal Meshes

Properties

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

MIT

SGI
Polygonal Mesh Processing

Analysis
- Normals
- Curvature

Warp
- Rotate
- Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel
Polygonal Mesh Processing

Analysis
- Normals
  - Curvature

Warp:
- 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

Warp
- Rotate
- Deform

Filter
- 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

Thouis "Ray" Jones

Weighted Average of Neighbor Vertices

Olga Sorkine
Polygonal Mesh Processing

## Analysis
- Normals
- Curvature

## Warps
- Rotate
- Deform

## Filters
- Smooth
- Sharpen
- Truncate
- Bevel

---

Desbrun

Weighted Average of Neighbor Vertices

Olga Sorkine
Polygonal Mesh Processing

Analysis
- Normals
- Curvature

Warp
- Rotate
- Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel

Conway 0.35
Polygonal Mesh Processing

Analysis
- Normals
- Curvature

Warp
- Rotate
- Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel

Jarek Rossignac

Wikipedia

Conway

0.40
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

Dirk Balfanz, Igor Guskov, Sanjeev Kumar, & Rudro Samanta,
Polyhedral Mesh Processing

Remeshing
- Subdivide
- Resample
- Simplify

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

Boolean operations
- Crop
- Subtract

- more uniform distribution
- triangles with nicer aspect

Original Resampled

Sorkine
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

Vertex Clustering

Rossignac
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
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
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](image)
Most operations use a few low-level operations:

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

Collapse edge
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
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
Important properties of mesh representation?

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

![Diagram of a polyhedron with vertex coordinates](image)

<table>
<thead>
<tr>
<th>FACE TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>F₁</td>
</tr>
<tr>
<td>F₂</td>
</tr>
<tr>
<td>F₃</td>
</tr>
<tr>
<td>(x₁, y₁, z₁) (x₂, y₂, z₂) (x₃, y₃, z₃)</td>
</tr>
<tr>
<td>(x₂, y₂, z₂) (x₄, y₄, z₄) (x₃, y₃, z₃)</td>
</tr>
<tr>
<td>(x₂, y₂, z₂) (x₅, y₅, z₅) (x₄, y₄, z₄)</td>
</tr>
</tbody>
</table>
Vertex and Face Tables

Each face lists vertex references

- Shared vertices
- Still no adjacency information

<table>
<thead>
<tr>
<th>VERTEX TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_1$</td>
</tr>
<tr>
<td>$V_2$</td>
</tr>
<tr>
<td>$V_3$</td>
</tr>
<tr>
<td>$V_4$</td>
</tr>
<tr>
<td>$V_5$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FACE TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$</td>
</tr>
<tr>
<td>$F_2$</td>
</tr>
<tr>
<td>$F_3$</td>
</tr>
</tbody>
</table>
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:

![Diagram of Winged Edge with vertices and faces labeled with coordinates and identifiers]

<table>
<thead>
<tr>
<th>VERTEX TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_1 )</td>
</tr>
<tr>
<td>( V_2 )</td>
</tr>
<tr>
<td>( V_3 )</td>
</tr>
<tr>
<td>( V_4 )</td>
</tr>
<tr>
<td>( V_5 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EDGE TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_1 )</td>
</tr>
<tr>
<td>( e_2 )</td>
</tr>
<tr>
<td>( e_3 )</td>
</tr>
<tr>
<td>( e_4 )</td>
</tr>
<tr>
<td>( e_5 )</td>
</tr>
<tr>
<td>( e_6 )</td>
</tr>
<tr>
<td>( e_7 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FACE TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_1 )</td>
</tr>
<tr>
<td>( F_2 )</td>
</tr>
<tr>
<td>( F_3 )</td>
</tr>
</tbody>
</table>
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