Polygonal Meshes

Adam Finkelstein & Tim Weyrich
Princeton University
COS 426, Spring 2008

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

Geometry background

Scene is usually approximated by 3D primitives
- Point
- Vector
- Line segment
- Ray
- Line
- Plane
- Polygon

Geometry & topology

Face
Edge
Vertex
(x,y,z)
3D Point

Spefies a location
- Represented by three coordinates
- Infinitely small

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

3D Vector

Spefies a direction and a magnitude
- Represented by three coordinates
- Magnitude ||V|| = sqrt(dx dx + dy dy + dz dz)
- Has no location

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

3D Vector

Dot product of two 3D vectors
- V1·V2 = ||V1|| || V2|| cos(θ)

3D Vector

Cross product of two 3D vectors
- V1xV2 = (dy1dx2 - dz1dy2, dz1dx2 - dx1dz2, dx1dy2 - dy1dx2)
- V1xV2 = vector perpendicular to both V1 and V2
- ||V1xV2|| = ||V1|| || V2|| sin(θ)

3D Line Segment

Linear path between two points
- Parametric representation:
  P = P1 + t (P2 - P1), (0 ≤ t ≤ 1)

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

3D Ray

Line segment with one endpoint at infinity
- Parametric representation:
  P = P1 + t V, (0 ≤ t < ∞)

typedef struct {
  Point P1;
  Vector V;
} Ray;
3D Line

- Line segment with both endpoints at infinity
  - Parametric representation:
    » \( P = P_1 + tV \), \((-\infty < t < \infty)\)

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

3D Plane

- A linear combination of three points
  - Implicit representation:
    » \( P \cdot N + d = 0 \), or
    » \( ax + by + cz + d = 0 \)
    - \( N \) is the plane “normal”
    - Unit-length vector
    - Perpendicular to plane

```
typedef struct {
  Vector N;
  Distance d;
} Plane;
```

3D Polygon

- Set of points “inside” a sequence of coplanar points
  - Points are in counter-clockwise order

```
typedef struct {
  Point *points;
  int npoints;
} Polygon;
```

3D Polygonal Mesh

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

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

Digital Michelangelo Project
Stanford
**Polygonal Mesh Acquisition**

Interactive modeling
- Polygon editors
- Interchange formats

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

Simulations
- Physical processes

**Outline**

Acquisition
- Processing
- Representation

**Polygonal Mesh Processing**

Warps
- Rotate
- Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel

Analysis
- Normals
- Curvature
Polygonal Mesh Processing

Warps
- Rotate
- Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel

Analysis
- Normals
- Curvature

Sheffer

Weighted Average of Neighbor Vertices

Thouis “Ray” Jones

Olga Sorkine
Polygonal Mesh Processing

Remeshing
- Subdivide
- Resample
- Simplify

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

Boolean operations
- Crop
- Subtract

---

Remeshing
- Subdivide
- Resample
- Simplify

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

Boolean operations
- Crop
- Subtract

---

Remeshing
- Subdivide
- Resample
- Simplify

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

Boolean operations
- Crop
- Subtract

---

Procedural generation
- Surface of revolution
- Sweep
- Fractalize
Polygonal Mesh Processing

Procedural generation
- Surface of revolution
- Sweep
- Fractalize

Most operations use a few low-level operations:
- Subdivide face
- Subdivide edge
- Collapse edge
- Merge vertices
- Remove vertex

Subdivide face

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

Outline

Acquisition
Processing
Representation

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

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

Vertex and Face Tables

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

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

**Example:**

```
\begin{tabular}{|c|c|c|c|}
\hline
$V_1$ & $V_2$ & $V_3$ & $V_4$ \\
\hline
$E_1$ & $E_2$ & $E_3$ & \ \\
\hline
\end{tabular}
```

**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

**Simple Triangle Mesh**

Do not store edges at all
- All faces have 3 vertices and 3 neighbors

Store adjacency in vertices and faces
- For each face: 3 vertices and 3 faces
- For each vertex: $N$ faces

**Summary**

Polygonal meshes
- Easy acquisition
- 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