3D Modeling - Overview

CG10a
Lior Shapira
Lecture 08

Some slides from Thomas Funkhouser, Princeton University
Course Syllabus

I. Image processing
II. Rendering
III. **Modeling**
IV. Animation

**Image Processing**
*(Bae et al, Two-scale Tone Management)*

**Modeling**
*(Dennis Zorin, CalTech)*

**Rendering**
*(POVray hall of fame)*

**Animation**
Modeling

- How do we ...
  - **Acquire** computer representations of 3D objects?
  - **Represent** 3D objects in a computer?
  - **Manipulate** computer representations of 3D objects?
How can these objects be represented in a computer?
3D Object Representations

- Points
  - Point cloud
  - Range image

- Surfaces
  - Polygonal Mesh
  - Subdivision
  - Parametric
  - Implicit

- Solids
  - Voxels
  - BSP tree
  - CSG
  - Sweep

- High-level structures
  - Scene graph
  - Application specific
Equivalence of Representations

• Thesis:
  ◦ Each representation has enough expressive power to model the shape of any geometric object
  ◦ It is possible to perform all geometric operations with any fundamental representation

• Analogous to Turing-equivalence
  ◦ Computers / programming languages Turing-equivalent. But each does different things better!
Why different Representations?

- Efficiency for different tasks
  - Acquisition
  - Rendering
  - Manipulation
  - Animation
  - Analysis

Data Structures determine algorithms!
Modeling Operations

• What can we do with a 3D object representation?
  ◦ Edit
  ◦ Transform
  ◦ Smooth
  ◦ Render
  ◦ Animate
  ◦ Morph
  ◦ Compress
  ◦ Transmit
  ◦ Analyze
  ◦ …

Digital Michealangelo

Pirates of the carribean

Smoothing
3D Object Representations

Desirable properties depend on intended use
- Easy to acquire
- Accurate
- Concise
- Intuitive editing
- Efficient editing
- Efficient display
- Efficient intersections
- Guaranteed validity
- Guaranteed smoothness
- …
• **Points**
  ◦ Point cloud
  ◦ Range image

• **Surfaces**
  ◦ Polygonal Mesh
  ◦ Subdivision
  ◦ Parametric
  ◦ Implicit

• **Solids**
  ◦ Voxels
  ◦ BSP tree
  ◦ CSG
  ◦ Sweep

• **High-level structures**
  ◦ Scene graph
  ◦ Application specific
- Set of 3D points mapping to pixels of depth image
  - Acquired from range scanner
Point Cloud

- Unstructured set of 3D point samples
  - Acquired from range finder, computer vision, etc

Hugues Hoppe
• Points
  ◦ Point cloud
  ◦ Range image

• Surfaces
  ◦ Polygonal Mesh
  ◦ Subdivision
  ◦ Parametric
  ◦ Implicit

• Solids
  ◦ Voxels
  ◦ BSP tree
  ◦ CSG
  ◦ Sweep

• High-level structures
  ◦ Scene graph
  ◦ Application specific
Polygonal Mesh

- Connected set of polygons (usually triangles)
Subdivision Surface

- Coarse mesh & subdivision rule
  - Define smooth surface as limit of sequence of refinements
Parametric Surface

- Tensor product spline patches
  - Each patch is a parametric function
  - Careful constraints to maintain continuity

FvDFH Figure 11.44
Implicit Surface

- Points satisfying: $F(x,y,z) = 0$

Polygonal Model

Implicit Model

Bill Lorensen
SIGGRAPH 99
Course #4 Notes
• Points
  ◦ Point cloud
  ◦ Range image

• Surfaces
  ◦ Polygonal Mesh
  ◦ Subdivision
  ◦ Parametric
  ◦ Implicit

• Solids
  ◦ Voxels
  ◦ BSP tree
  ◦ CSG
  ◦ Sweep

• High-level structures
  ◦ Scene graph
  ◦ Application specific
Voxels

- Uniform grid of volumetric samples
  - Acquired from CAT, MRI, etc.
• Binary space partition with solid cells labeled
  ◦ Constructed from polygonal representations
• Hierarchy of boolean set operations (union, difference, intersect) applied to simple shapes

Boolean union  Boolean difference  Boolean intersection
CSG (constructive solid geometry)

- Hierarchy of boolean set operations (union, difference, intersect) applied to simple shapes
• Solid swept by curve along trajectory

Removal Path

Sweep Model
Outline

- **Points**
  - Point cloud
  - Range image

- **Surfaces**
  - Polygonal Mesh
  - Subdivision
  - Parametric
  - Implicit

- **Solids**
  - Voxels
  - BSP tree
  - CSG
  - Sweep

- **High-level structures**
  - Scene graph
  - Application specific
Scene Graph

- Union of objects at leaf nodes
Application Specific

Apo A-1
(Theoretical Biophysics Group, University of Illinois at Urbana-Champaign)

Architectural Floorplan
(CS Building, Princeton University)
Taxonomy of 3D Representations

- Discrete
  - Voxels, Point Sets
- Continuous
- Combinatorial
- Topological
  - Mesh
  - Subdivision
- Set Membership
  - BSP Tree
  - Cell Complex
- Functional
  - Bezier
  - B-Spline
- Parametric
- Implicit
  - Algebraic
Equivalence of Representations

- Thesis:
  - Each representation has enough expressive power to model the shape of any geometric object
  - It is possible to perform all geometric operations with any fundamental representation

- Analogous to Turing-equivalence
  - Computers / programming languages Turing-equivalent. But each does different things better!
Computational Differences

- Efficiency
  - Combinatorial complexity (e.g. $O(n \log n)$)
  - Space/time trade-offs (e.g. z-buffer)
  - Numerical accuracy/stability (degree of polynomial)

- Simplicity
  - Ease of acquisition
  - Hardware acceleration
  - Software creation and maintenance

- Usability
  - Designer interface vs. computational engine
POLYGONAL MESH
Polygon Mesh

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

- Geometry and Topology

![Diagram showing vertex, edge, and face of a polygon mesh.](image-url)
A scene is usually **approximated** by 3D primitives
- Point
- Vector
- Line Segment
- Ray
- Line
- Plane
- Polygon
Geometry Background

• No need to expand too much on each of these right?
3D Polygon Meshes

- So what’s so interesting about them?
  - Simple, Common representation
  - Rendering with hardware support
  - Output of many acquisition tools
  - Input to many simulation/analysis tools
3D Polygon Meshes – what we want

- 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 surveys
  - CAT, MRI etc. (iso-surfaces)
- Simulations
  - Physical processes
Polygonal Mesh Acquisition

- Interactive Modeling
  - Polygon Editors
  - Interchange Formats
- Scanners
  - Laser range scanners
  - Geological surveys
  - CAT, MRI etc. (iso-surfaces)
- Simulations
  - Physical processes
Polygonal Mesh Acquisition

- Interactive Modeling
  - Polygon Editors
  - Interchange Formats

- Scanners
  - Laser range scanners
  - Geological surveys
  - CAT, MRI etc. (iso-surfaces)

- Simulations
  - Physical processes
Polygonal Mesh Acquisition

- Interactive Modeling
  - Polygon Editors
  - Interchange Formats
- Scanners
  - Laser range scanners
  - Geological surveys
  - CAT, MRI etc. (iso-surface)
- Simulations
  - Physical processes
Polygonal Mesh Acquisition

• Interactive Modeling
  ◦ Polygon Editors
  ◦ Interchange Formats

• Scanners
  ◦ Laser range scanners
  ◦ Geological surveys
  ◦ CAT, MRI etc. (iso-surfaces)

• Simulations
  ◦ Physical processes
Processing meshes

- **Analysis**
  - Normals
  - Curvature
  - Volume

- **Warps**
  - Rotate
  - Deform

- **Filters**
  - Smooth
  - Sharpen
  - Truncate
  - Bevel
Processing meshes

- **Analysis**
  - Normals
  - Curvature
  - Volume

- **Warps**
  - Rotate
  - Deform

- **Filters**
  - Smooth
  - Sharpen
  - Truncate
  - Bevel
• Analysis
  ◦ Normals
  ◦ Curvature
  ◦ Volume
• Warps
  ◦ Rotate
  ◦ Deform
• Filters
  ◦ Smooth
  ◦ Sharpen
  ◦ Truncate
  ◦ Bevel
Processing meshes

- Analysis
  - Normals
  - Curvature
  - Volume

- Warps
  - Rotate
  - Deform

- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel
Processing meshes

- **Analysis**
  - Normals
  - Curvature
  - Volume

- **Warps**
  - Rotate
  - Deform

- **Filters**
  - Smooth
  - Sharpen
  - Truncate
  - Bevel
Processing meshes

- Analysis
  - Normals
  - Curvature
  - Volume

- Warps
  - Rotate
  - Deform

- Filters
  - Smooth
  - Sharpen
  - Truncate
  - Bevel
Processing meshes

- **Analysis**
  - Normals
  - Curvature
  - Volume

- **Warps**
  - Rotate
  - Deform

- **Filters**
  - Smooth
  - Sharpen
  - Truncate
  - Bevel
Processing meshes

- **Analysis**
  - Normals
  - Curvature
  - Volume
- **Warps**
  - Rotate
  - Deform
- **Filters**
  - Smooth
  - Sharpen
  - **Truncate**
  - Bevel
Processing meshes

- **Analysis**
  - Normals
  - Curvature
  - Volume
- **Warps**
  - Rotate
  - Deform
- **Filters**
  - Smooth
  - Sharpen
  - Truncate
  - Bevel
Processing Meshes

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

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

Zorin & Schroeder
Processing Meshes

- Remeshing
  - Subdivide
  - Resample
  - Simplify

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

- Boolean operations
  - Crop
  - Subtract
Processing Meshes

- Remeshing
  - Subdivide
  - Resample
  - Simplify

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

- Boolean operations
  - Crop
  - Subtract
Processing Meshes

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

- **Remeshing**
  - Subdivide
  - Resample
  - Simplify

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

- **Boolean operations**
  - Crop
  - Subtract
Processing Meshes

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

- Remeshing
  - Subdivide
  - Resample
  - Simplify

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

- Boolean operations
  - Crop
  - Subtract
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

- What are the important properties of mesh representation?
What are the 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
Vertex and Face Tables

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

<table>
<thead>
<tr>
<th>VERTEX TABLE</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>X1</td>
<td>Y1</td>
<td>Z1</td>
</tr>
<tr>
<td>V2</td>
<td>X2</td>
<td>Y2</td>
<td>Z2</td>
</tr>
<tr>
<td>V3</td>
<td>X3</td>
<td>Y3</td>
<td>Z3</td>
</tr>
<tr>
<td>V4</td>
<td>X4</td>
<td>Y4</td>
<td>Z4</td>
</tr>
<tr>
<td>V5</td>
<td>X5</td>
<td>Y5</td>
<td>Z5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FACE TABLE</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
</tr>
<tr>
<td>F2</td>
<td>V2</td>
<td>V4</td>
<td>V3</td>
</tr>
<tr>
<td>F3</td>
<td>V2</td>
<td>V5</td>
<td>V4</td>
</tr>
</tbody>
</table>
Adjacency Lists

- Store all vertex, edge and face adjacencies
  - Efficient adjacency traversal
  - Extra storage
Can we store only some adjacency relationships and derive others?
• Adjacency encoded in (half) edges
  ◦ All adjacencies in $O(1)$ time
  ◦ Little extra storage (fixed records)
  ◦ Arbitrary polygons
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 depends on which operations must be efficient
Mesh segmentation
Sample application
Mesh Segmentation

- Problem definition
  - Given a mesh, divide it into parts according to a given criteria
    - “Given a mesh” – collection of triangles (or quads), no additional information
    - “Given criteria” – could be a lot of things!
Mesh Segmentation

- So given $M=\{V,E,F\}$

- A segmentation is the set of sub-meshes induced by a partition of the mesh into $k$ (disjoint) subsets.
Some results

- Good or bad?
Criteria?

- Planar segments?
- Smooth segments?
- Round segments?
- Small or large segments?
- Small number of patches?
- Smooth boundaries?
- Small or large segments?
- Small number of patches?
- Smooth boundaries?
- “Natural” segments?
- More…
Two Types of Segmentations

- Patch type (surface)
- Part type (volume)
So how do we solve this?

Define an optimization:

Given a mesh $M=\{V,E,F\}$ and the set of elements $S \in \{V,E,F\}$, find a disjoint partitioning of $S$ into $S_1, \ldots, S_k$ such that the criterion function $J=J(S_1, \ldots, S_k)$ is minimized under a set of constraints $C$. 
If $|S|=n$ and the number of parts is $k$, then the search space is order $k^n$.

Therefore we need an approximation algorithm

- Region growing (local greedy)
- Hierarchical clustering (global greedy)
- K-means (iterative)
- Graph cut
- Spectral analysis
- …
Constraints vs. Attributes

- **Constraints**
  - Imposed on the segments, must be preserved

- **Element attributes**
  - Used for the criteria measure in the optimization process
Types of Constraints

• Cardinality
  ◦ Number of segments or elements within a certain range
  ◦ Overall balanced partition

• Geometry
  ◦ Size: area, diameter, radius
  ◦ Convexity, Roundness
  ◦ Boundary smoothness

• Topology
  ◦ Connectivity (single component)
  ◦ Disk topology
Types of Attributes

- Distance and geodesic distance
- Planarity, normal direction
- Smoothness, curvature
- Distance to complex proxies
- Slippage
- Symmetry
- Medial axis, shape diameter
- Etc.
Example: Simplification

- We want to approximate a complex model with a simpler one
- Segment the mesh into regions, which will be replaced with simpler ones
Example: Simplification

- We want to approximate a complex model with a simpler one

- Strips & Quasi-developable surfaces
  - Mitani et al 2004
  - Julius et al 2005
  - Shatz et al 2006
Example: Compatible Parameterization

- Kraevoy, Sheffer 2004
- Schreiner et al 2004
Example: Shape Analysis

- Mortara et al 2004
- Katz, Tal 2003
- Shapira et al 2007
Example: Skeleton Extraction & Deformation
Example: Shape Matching & Retrieval

Shape retrieval

Query  Results (first page)  Results (second page)

Part retrieval
SUBDIVISION SURFACES
Subdivision Surfaces

- One attribute we’d like is **Guaranteed Continuity**
- Question: How do you make a smoothed curve?
Subdivision Rules

- Repeated application of
  - Mesh refinement
  - Weighted averaging of vertex positions
- Special treatment of surface boundaries
Subdivision Surfaces – Examples
Subdivision Surfaces – Examples

Scott Schaefer
Subdivision Surfaces – Examples

Scott Schaefer
Design of Subdivision Rules

- What types of input?
  - Quad meshes, triangle meshes, etc.

- How to refine topology?
  - Simple implementations

- How to refine geometry?
  - Smoothness guarantees in limit surface
    - Continuity ($C^0, C^1, C^2, \ldots$)
  - Provable relationships between limit surface and original control mesh
    - Interpolation of vertices
Linear Subdivision

- Operates on quad-meshes
  - Input mesh consists of four-sided polygons (quads)
  - Any number of quads may touch each vertex
- Subdivision rule:
  - Split every quad into four at midpoints, and then
  - Average vertex positions

This is a simple example to demonstrate how subdivision schemes work
Linear Subdivision

- Topology refinement
- Geometry refinement
Linear Subdivision \((F_0, V_0, k)\)

for \(i = 1 \ldots k\) levels

\[(F_i, V_i) = \text{RefineTopology}(F_{i-1}, V_{i-1})\]

\[\text{RefineGeometry}(F_i, V_i)\]

return \((F_k, V_k)\)
RefineTopology ( F, V )

newV = V
newF = {}

for each face $F_i$
  Insert new vertex $c$ at centroid of $F_i$ into newV
  for $j = 1$ to $4$
    Insert new vertex $e_j$ at centroid of each edge $(F_{i,j}, F_{i,j+1})$ into newV
  for $j = 1$ to $4$
    Insert new face $(F_{i,j}, e_j, c, e_{j-1})$ into newF

return (newF, newV)
Linear Subdivision

RefineGeometry( F, V )

newV = 0 * V
val = array of 0 whose size is number of vertices
newF = F
for each face $F_i$
    cent = centroid for $F_i$
    newV[$F_i$] += cent  // syntax: repeat for all vtx indices in $F_i$
    val[$F_i$] += 1  // syntax: repeat for all vtx indices in $F_i$
for each vertex newV[i]
    newV[i] /= val[i]
return (newF, newV)
Linear Subdivision

- Example
Catmull-Clark Subdivision

- Founded in 1978 by Adnan Catmull and Jim Clark.
- Won an Academy Award for technical contribution in 2005.

Repeated Averaging  Catmull-Clark
Catmull-Clark Subdivision

- One round of subdivision produces all quads
- Smoothness of limit surface
  - $C^2$ almost everywhere
  - $C^1$ at vertices with valence $\neq 4$
- Relationship to control mesh
  - Does not interpolate input vertices
  - Within convex hull
- Most commonly used subdivision scheme today (still!)
Calculate new position for vertex P:

- \( F \) – average of face vertices
- \( R \) – average of edge vertices
- \( n \) – number of points touching P

Move P to

\[
\hat{P} = \frac{F + 2R + (n - 3)P}{n}
\]
Subdivision Schemes

Loop
Butterfly
Catmull-Clark
Subdivision Surfaces

- Properties:
  - Accurate
  - Concise
  - Intuitive specification
  - Local support
  - Affine invariant
  - Arbitrary topology
  - Guaranteed continuity
  - Natural parameterization
  - Efficient display
  - Efficient intersections
Subdivision Surfaces

- **Advantages:**
  - Simple method for describing complex surfaces
  - Relatively easy to implement
  - Arbitrary topology
  - Local support
  - Guaranteed continuity
  - Multiresolution

- **Difficulties:**
  - Intuitive specification
  - Parameterization
  - Intersections
Subdivision Surfaces

- Used in movie and game industries
- Supported by most 3D modeling software

Scott Schaefer

Geri's Game © Pixar Animation Studios
<table>
<thead>
<tr>
<th>Feature</th>
<th>Polygonal Mesh</th>
<th>Subdivision Surface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accurate</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Concise</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Intuitive specification</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Local support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Affine invariant</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Arbitrary topology</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Guaranteed continuity</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Natural parameterization</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Efficient display</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Efficient intersections</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>