3D Surfaces

source:
Mesh Representations & Subdivision Surfaces

- Tom Funkhouser
- Princeton University
- COS 426, Spring 2007
3D Object Representations

• Raw data
  o Voxels
  o Point cloud
  o Range image
  o Polygons

• Surfaces
  o Mesh
  o Subdivision
  o Parametric
  o Implicit

• Solids
  o Octree
  o BSP tree
  o CSG
  o Sweep

• High-level structures
  o Scene graph
  o Application specific
Surfaces

- What makes a good surface representation?
  - Accurate
  - Concise
  - Intuitive specification
  - Local support
  - Affine invariant
  - Arbitrary topology
  - Guaranteed continuity
  - Natural parameterization
  - Efficient display
  - Efficient intersections

H&B Figure 10.46
2D Scalar Field

- $z = f(x, y)$

$$f(x, y) = \begin{cases} 
1 - x^2 - y^2, & \text{if } x^2 + y^2 < 1 \\
0 & \text{otherwise}
\end{cases}$$

How do you visualize this function?
Height Field

- Visualizing an explicit function
  \[ z = f(x,y) \]

- Adding contour curves
  \[ f(x,y) = c \]
Implicit $\rightarrow$ Explicit 2D
(Marching Squares Algorithm)
Marching Squares

- Sample function $f$ at every grid point $x_i, y_j$
- For every point $f_{ij} = f(x_i, y_j)$ either $f_{ij} \leq c$ or $f_{ij} > c$
Cases for Vertex Labels

16 cases for vertex labels

4 unique mod. symmetries
Ambiguities of Labelings

Ambiguous labels

Different resulting contours

Resolution by subdivision (where possible)
Marching Squares Examples

Can you do better?
Interpolating Intersections

- Approximate intersection
  - Midpoint between \( x_i \), \( x_{i+1} \) and \( y_j \), \( y_{j+1} \)
  - Better: interpolate

- If \( f_{i,j} = a \) is closer to \( c \) than \( b = f_{i+1,j} \) then intersection is closer to \((x_i, y_j)\):
  \[
  \frac{x - x_i}{x_{i+1} - x} = \frac{c - a}{b - c}
  \]

- Analogous calculation for \( y \) direction

\[ f_{i,j} = a < c \quad c < b = f_{i+1,j} \]
Implicit → Explicit 3D
(Marching Cubes Algorithm)
3D Scalar Fields

- Volumetric data sets
- Example: tissue density
- Assume again regularly sampled

\[
\begin{align*}
x_i &= x_0 + i \Delta x \\
y_j &= y_0 + j \Delta y \\
z_k &= z_0 + k \Delta z
\end{align*}
\]

- Represent as voxels

- Two rendering methods
  - Isosurface rendering
  - Direct volume rendering
Isosurfaces

• Generalize contour curves to 3D

• Isosurface given by $f(x,y,z) = c$
  - $f(x, y, z) < c$ inside
  - $f(x, y, z) = c$ surface
  - $f(x, y, z) > c$ outside
Marching Cubes

- Display technique for isosurfaces
- 3D version of marching squares
- How many possible cases?

\[ 2^8 = 256 \]
Marching Cubes

- 14 cube labelings (after elimination symmetries)
Marching Cube Tessellations

- Generalize marching squares, just more cases
- Interpolate as in 2D
- Ambiguities similar to 2D
Marching Squares Examples
Marching Squares Examples

12 Subdivisions

20 Subdivisions

50 Subdivisions

12 Adaptive Subdivisions

20 Adaptive Subdivisions
3D Object Representations

- **Raw data**
  - Voxels
  - Point cloud
  - Range image
  - Polygons

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

- **Solids**
  - Octree
  - BSP tree
  - CSG
  - Sweep

- **High-level structures**
  - Scene graph
  - Application specific
Polygon Meshes

• How should we represent a mesh in a computer?
  o Efficient traversal of topology
  o Efficient use of memory
  o Efficient updates

• Mesh Representations
  o Independent faces
  o Vertex and face tables
  o Adjacency lists
  o Winged-Edge
  o Half-Edge
  o etc.

Zorin & Schroeder, SIGGRAPH 99, Course Notes
Independent Faces

• Each face lists vertex coordinates
  o Redundant vertices
  o No adjacency information

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

• Each face lists vertex references
  o Shared vertices
  o Still no adjacency information

\[
\begin{align*}
F_1 &: (x_1, y_1, z_1) & (x_3, y_3, z_3) \\
F_2 &: (x_2, y_2, z_2) \\
F_3 &: (x_4, y_4, z_4) & (x_5, y_5, z_5)
\end{align*}
\]

<table>
<thead>
<tr>
<th>VERTEX TABLE</th>
<th>FACE TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_1 \ V_2 \ V_3 \ V_4 \ V_5</td>
<td>F_1 \ F_2 \ F_3</td>
</tr>
<tr>
<td>X_1 \ Y_1 \ Z_1</td>
<td>V_1 \ V_2 \ V_3</td>
</tr>
<tr>
<td>X_2 \ Y_2 \ Z_2</td>
<td>V_2 \ V_4 \ V_3</td>
</tr>
<tr>
<td>X_3 \ Y_3 \ Z_3</td>
<td>V_2 \ V_5 \ V_4</td>
</tr>
<tr>
<td>X_4 \ Y_4 \ Z_4</td>
<td>\</td>
</tr>
</tbody>
</table>
Possible Queries

• Which faces use this vertex?
• Which edges use this vertex?
• Which faces border this edge?
• Which edges border this face?
• Which faces are adjacent to this face?
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
  o All adjacencies in O(1) time
  o Little extra storage (fixed records)
  o Arbitrary polygons
Winged Edge

- Example:

![Diagram of Winged Edge](image)

**VERTEX TABLE**

<table>
<thead>
<tr>
<th>V_i</th>
<th>X_i</th>
<th>Y_i</th>
<th>Z_i</th>
<th>e_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_1</td>
<td>X_1</td>
<td>Y_1</td>
<td>Z_1</td>
<td>e_1</td>
</tr>
<tr>
<td>V_2</td>
<td>X_2</td>
<td>Y_2</td>
<td>Z_2</td>
<td>e_6</td>
</tr>
<tr>
<td>V_3</td>
<td>X_3</td>
<td>Y_3</td>
<td>Z_3</td>
<td>e_3</td>
</tr>
<tr>
<td>V_4</td>
<td>X_4</td>
<td>Y_4</td>
<td>Z_4</td>
<td>e_5</td>
</tr>
<tr>
<td>V_5</td>
<td>X_5</td>
<td>Y_5</td>
<td>Z_5</td>
<td>e_6</td>
</tr>
</tbody>
</table>

**EDGE TABLE**

<table>
<thead>
<tr>
<th>e_i</th>
<th>V_1</th>
<th>V_2</th>
<th>F_1</th>
<th>F_2</th>
<th>F_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_1</td>
<td>V_1</td>
<td>V_3</td>
<td>F_1</td>
<td>F_1</td>
<td>F_3</td>
</tr>
<tr>
<td>e_2</td>
<td>V_1</td>
<td>V_2</td>
<td>F_1</td>
<td>F_2</td>
<td>F_3</td>
</tr>
<tr>
<td>e_3</td>
<td>V_2</td>
<td>V_3</td>
<td>F_1</td>
<td>F_2</td>
<td>F_3</td>
</tr>
<tr>
<td>e_4</td>
<td>V_3</td>
<td>V_4</td>
<td>F_2</td>
<td>F_2</td>
<td>F_3</td>
</tr>
<tr>
<td>e_5</td>
<td>V_2</td>
<td>V_4</td>
<td>F_2</td>
<td>F_3</td>
<td>F_3</td>
</tr>
<tr>
<td>e_6</td>
<td>V_2</td>
<td>V_5</td>
<td>F_3</td>
<td>F_3</td>
<td>F_3</td>
</tr>
<tr>
<td>e_7</td>
<td>V_4</td>
<td>V_5</td>
<td>F_3</td>
<td>F_3</td>
<td>F_3</td>
</tr>
</tbody>
</table>

**FACE TABLE**

<table>
<thead>
<tr>
<th>F_i</th>
<th>e_j</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_1</td>
<td>e_1</td>
</tr>
<tr>
<td>F_2</td>
<td>e_3</td>
</tr>
<tr>
<td>F_3</td>
<td>e_5</td>
</tr>
</tbody>
</table>
Simple Triangle Mesh

- Do not store edges at all
  o All faces have 3 vertices and 3 neighbors
- Store adjacency in vertices and faces
  o For each face: 3 vertices and 3 faces
  o For each vertex: N faces
3D Object Representations

- **Raw data**
  - Voxels
  - Point cloud
  - Range image
  - Polygons

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

- **Solids**
  - Octree
  - BSP tree
  - CSG
  - Sweep

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

• What makes a good surface representation?
  o Accurate
  o Concise
  o Intuitive specification
  o Local support
  o Affine invariant
  o Arbitrary topology
  ➢ Guaranteed continuity
  o Natural parameterization
  o Efficient display
  o Efficient intersections

H&B Figure 10.46
Subdivision

- How do you make a smooth curve?
Subdivision Surfaces

- Coarse mesh & subdivision rule
  - Define smooth surface as limit of sequence of refinements
Key Questions

- How refine mesh?
  - Aim for properties like smoothness

- How store mesh?
  - Aim for efficiency for implementing subdivision rules
Subdivision Surfaces – A 3D example
Applications: Computer Graphics Animation
Subdivision Surfaces

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

• Difficulties:
  o Intuitive specification
  o Parameterization
  o Intersections
<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>