Geometric Modeling

Modeling

- Triangular Mesh Basics
  - parameterization
- Scene-graph
- Object representation
  - B-rep, CSG
- Spatial organizations
  - Octree, kd-tree, BSP
How to represent objects/scenes

Polygon meshes
Scene graphs
Parametric representations
Implicit surfaces
Point-based method

Point Clouds

Discrete 3D point set
- Raw data (scanner)
- Densely sampled CAGD model
- Point-based method
  - Reconstruct patches on demand
  - Hidden point removal possible
Polygon Soup

Unstructured set up polygons
- Just a list of vertices for each polygon
- Redundant
- Usually transformed into better model before being used

What is tessellation?

Representing a surface with flat approximating polygons
- A sampled representation

Geometric Aliasing
- Error in the representation due to the sampling with polygons
Polygon mesh

Common method for describing geometry

Set of vertices
- Coordinates
- Normal
- Texture coordinates
- Etc.

Set of faces
- Pointers to vertices
- Material properties

Common: Triangle mesh

Polygon Mesh

Retains shared-vertex relationships

Vertex a,b,c,d;

Polygon P1(a, b, c);
Polygon P2(c, b, d);

b,c shared by two polygons
A Common Format: VRML

**VRML**

<table>
<thead>
<tr>
<th>Vertices (geometry)</th>
<th>Faces (connectivity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1 (x1,y1,z1)</td>
<td>f1 (v1,v3,v2)</td>
</tr>
<tr>
<td>v2 (x2,y2,z2)</td>
<td>f2 (v4,v3,v1)</td>
</tr>
<tr>
<td>v3 (x3,y3,z3)</td>
<td>f3 (v4,v1,v5)</td>
</tr>
<tr>
<td>v4 (x4,y4,z4)</td>
<td>f4 (v1,v6,v5)</td>
</tr>
<tr>
<td>v5 (x5,y5,z5)</td>
<td>f5 (v6,v1,v7)</td>
</tr>
<tr>
<td>v6 (x6,y6,z6)</td>
<td>f6 (v2,v7,v1)</td>
</tr>
<tr>
<td>v7 (x7,y7,z7)</td>
<td>f7 (...)</td>
</tr>
</tbody>
</table>

Valence n: n times in face list

Surface normal

Assume the following polygon:

\[
N = \frac{(p_2 - p_1) \times (p_4 - p_1)}{|(p_2 - p_1) \times (p_4 - p_1)|}
\]
Vertex normals

Add normals for surfaces incident on vertex:

Winged-Edge Data Structure

Commonly used B-rep
- Close to half-edge

Maintains
- List of vertices
- List of edges
- List of faces
- Adjacency
Winged-Edge Data Structure

### Vertex table:

```
<table>
<thead>
<tr>
<th>Vertex</th>
<th>Incident Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>a</td>
</tr>
<tr>
<td>B</td>
<td>b</td>
</tr>
<tr>
<td>C</td>
<td>e</td>
</tr>
<tr>
<td>D</td>
<td>a</td>
</tr>
</tbody>
</table>
```

### Face table:

```
<table>
<thead>
<tr>
<th>Face</th>
<th>Incident Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>e</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
</tr>
</tbody>
</table>
```

---

**Edge Table**

```
<table>
<thead>
<tr>
<th>Edge Name</th>
<th>Vertices</th>
<th>Faces</th>
<th>Left Traverse</th>
<th>Right Traverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>B D</td>
<td>1 2</td>
<td>b a</td>
<td>e d</td>
</tr>
</tbody>
</table>
```

---

CSE 872 Fall 2011
Voxels

Grid of volumetric samples
- CAT, MRI, etc.
- Enumerate small cubes inside

[HTG03]

Scene Graphs

Composite → Separator → Translate → Color → Composite

Composite → Separator → Translate → Color → Composite

Barbell Ends

Barbell Bar
Collapsing a Scene Graph

It’s often possible to apply transformations in a scene graph once to the underlying vertices, collapsing some nodes. Example:

Cloning

If part of a scene graph has two edges to it, it will be necessary to clone before collapsing.
SceneGraphs

- OpenSG
- OpenScenGraph
- Older
  - OpenGL Performer
  - Open Inventor
- Part of game engines
Constructive Solid Geometry

Composite (scene graph)
- Boolean Operations
  - Union, Intersection, Subtraction

- Usually in tree structure (with instancing)

Bounding Boxes
- AABB tree vs OBB tree
- Bounding Sphere

---

[Cybercreations.com] [Gottschalk et al]
Levels of Detail

Alternative scene graphs with different resolutions
- Varying tessellation
- Progressive refinement, visibility
- Static vs. viewing direction, performance and phenomena

Spatial Organization

- Hierarchical Bounding Box Tree
- Spatial Indexing or Partitioning
  - Going from x,y,z to Objects
    - Sometimes, with a viewing direction
  - BSP (k-d tree, octree)
  - Applications
    - Collision detection, particle systems,
      user interaction, raytracing,
      painter’s algorithm...
Sample application

(a) Collision Culling with AABB hierarchy  
(b) Collision Culling with AABB hierarchy + 2.5D overlap tests  

[Govindaraju et al. 05]

BSP Trees

Partition space into 2 half-spaces via a hyper-plane
BSP Trees

Advantages
- view-independent tree
- anti-aliasing
- transparency

Disadvantages
- many, small polygons
- over-rendering
- hard to balance tree

Painter’s Algorithm

Farthest z extent is insufficient
Cannot resolve dependency cycles
k-d Trees

k is dimensions

Node
- Info
- x, y
- lt-link
- ge-link

Levels

Level 0 – Index X
- Level 1 – Index Y
- Level 2 – Index X

level = 0 if root, level(parent) + 1 otherwise
For each level, index: level mod k

3D: Index X, Index Y, Index Z, Index X, etc…
**Example**

Banja Luca (19, 45)
Derventa (40, 50)
Teslic (38, 38)
Tuzla (54, 40)
Sinj (4, 4)

Insert: East Broko (21, 57)
Search for exact?
Search for nearest?
Range searches?

---

**k-d Trees**

Search for exact
- Okay, but may be o(k)

Search for nearest
- In a moment

Range search?
Range Search

Consider each tree level to be region
Recursively search all regions that overlap search range

Example Regions

Range: (10,20) (20,45)
k-d Trees

Fast and easy
Tend to be rather tall (unbalanced)
How could we extend to disk structure?
What if k=1?

Quadtrees

2-d only
Split data 4 ways
Node:
- info
- x,y
- nw,sw,ne,se
Example

- Banja Luca (19, 45)
- Derventa (40, 50)
- Teslic (38, 38)
- Tuzla (54, 40)
- Sinj (4, 4)

Insert: East Broko (21, 57)
Search for exact?
Search for nearest?
Range searches?

Octtrees

3-d
Split data 8 ways
Node:
- info
- x,y,z
- up nw,sw,ne,se, down nw,sw,ne,se