Overview

• 3D solid model representations
  – Implicit models
  – Super/quadrics
  – Blobbies
  – Swept objects
  – Boundary representations
  – Spatial enumerations
  – Distance fields
  – Quadtrees/octrees
  – Stochastic models
Implicit Solid Modeling

• Idea:
  – Represents solid as the set of points where an implicit global function takes on certain values
    • Usually
      • $F(x,y,z) < 0$, points inside of object
      • $F(x,y,z) = 0$, points on object’s surface
      • $F(x,y,z) > 0$, points outside of object
  – Primitive solids are combined using CSG
  – Composition operations are implemented by functionals which provide an implicit function for the resulting solid

From M.Ganter, D. Storti, G. Turkiyyah @ UW
Quadratic Surfaces

- **Sphere**
  \[ x^2 + y^2 + z^2 = r^2 \]

- **Ellipsoid**
  \[ \left( \frac{x}{r_x} \right)^2 + \left( \frac{y}{r_y} \right)^2 + \left( \frac{z}{r_z} \right)^2 = 1 \]

- **Torus**
  \[ \left[ r - \sqrt{\left( \frac{x}{r_x} \right)^2 + \left( \frac{y}{r_y} \right)^2} \right]^2 + \left( \frac{z}{r_z} \right)^2 = 1 \]

- **General form**
  \[ a \cdot x^2 + b \cdot y^2 + c \cdot z^2 + 2f \cdot yz + 2g \cdot xz + 2h \cdot xy + 2p \cdot x + 2q \cdot y + 2r \cdot z + d = 0 \]
Superellipsoid Surfaces

- Generalization of ellipsoid
- Control parameters $s_1$ and $s_2$

\[
\left( \frac{x}{r_x} \right)^{2/s_2} + \left( \frac{y}{r_y} \right)^{2/s_2} + \left( \frac{z}{r_z} \right)^{2/s_1} = 1
\]

- If $s_1 = s_2 = 1$ then regular ellipsoid
- Has an implicit and parametric form!
CSG with Superquadrics
CSG with Superellipsoids
Blobby Objects

- Do not maintain shape, topology
  - Water drops
  - Molecules
  - Force fields

- But can maintain other properties, like volume
Gaussian Bumps

- Model object as a sum of Gaussian bumps/blobs

\[ f(x,y,z) = \sum_{k} b_k e^{-a_k r_k^2} - T = 0 \]

- Where \( r_k^2 = x_k^2 + y_k^2 + z_k^2 \) and \( T \) is a threshold.
Metaballs (Blinn Blobbies)
Implicit Modeling System
U. of Calgary

- Combine “primitives”
  - Points, lines, planes, polygons, cylinders, ellipsoids
- Calculate field around primitives
- View Iso-surface of implicit function
Implicit Modeling System
U. of Calgary

The Blob Tree

Can apply blends and warps
Sweep Representations

• An alternative way to represent a 3D object

• Idea
  – Given a primitive (e.g. polygon, sphere)
  – And a sweep (e.g. vector, curve…)
  – Define solid as space swept out by primitive

Foley/VanDam, 1990/1994
Sweep Representations

• Issues:
  – How to generate resulting surface?
  – What about self-intersections?
  – How to define intersection?

Foley/VanDam, 1990/1994
Approximate Representations

• Idea: discretize the world!
• Surface Models
  – Mesh, facet and polygon representations
• Volume Models
  – spatial enumeration
  – voxelization
Examples

• From exact to facets…. 

Pics/Math courtesy of Dave Mount @ UMD-CP
Boundary Representation
Solid Modeling

• The de facto standard for CAD since ~1987
  – BReps integrated into CAGD surfaces + analytic surfaces + boolean modeling
• Models are defined by their boundaries
• Topological and geometric integrity constraints are enforced for the boundaries
  – Faces meet at shared edges, vertices are shared, etc.
Let’s Start Simple: Polyhedral Solid Modeling

• Definition
  – Solid bounded by polygons whose edges are each a member of an even number of polygons
  – A 2-manifold: edges members of 2 polygons
Properties of 2-Manifolds

• For any point on the boundary, its neighborhood is a topological 2D disk
• If not a 2-manifold, neighborhood not a disk
Euler’s Formula

- For simple polyhedron (no holes):
  \#Vertices - \#Edges + \#Faces = 2
- If formula is true the surface is closed

Foley/VanDam, 1990/1994
Euler’s Formula (Generalized)

\[#\text{Vertices} - \#\text{Edges} + \#\text{Faces} - \#\text{Holes\_in\_faces} = 2 (#\text{Components} – \#\text{Genus})\]

- Genus is the \# holes through the object
- Euler Operators have been the basis of several modeling systems (Mantyla et al.)

\[V - E + F - H = 2(C - G)\]

\[\begin{array}{cccccc}
24 & 36 & 15 & 3 & 1 & 1
\end{array}\]
Euler Operators

<table>
<thead>
<tr>
<th>Operator Name</th>
<th>Meaning</th>
<th>V</th>
<th>E</th>
<th>F</th>
<th>L</th>
<th>S</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEV</td>
<td>Make an edge and a vertex</td>
<td>+1</td>
<td>+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFE</td>
<td>Make a face and an edge</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSFV</td>
<td>Make a shell, a face and a vertex</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG</td>
<td>Make a shell and a hole</td>
<td></td>
<td></td>
<td></td>
<td>+1</td>
<td>+1</td>
<td></td>
</tr>
<tr>
<td>MEKL</td>
<td>Make an edge and kill a loop</td>
<td>+1</td>
<td></td>
<td></td>
<td>-1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator Name</th>
<th>Meaning</th>
<th>V</th>
<th>E</th>
<th>F</th>
<th>L</th>
<th>S</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEV</td>
<td>Kill an edge and a vertex</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KFE</td>
<td>Kill a face and an edge</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KSFV</td>
<td>Kill a shell, a face and a vertex</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KSG</td>
<td>Kill a shell and a hole</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>KEML</td>
<td>Kill an edge and make a loop</td>
<td>-1</td>
<td>+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Loop L $\rightarrow$ H,  Shell S $\rightarrow$ C
Steps to Creating a Polyhedral Solid Modeler

• Representation
  – Points, Lines/Edges, Polygons

• Modeling
  – Generalization of 3D clipping to non-convex polyhedra, enables implementation of booleans
State of the Art: BRep Solid Modeling

• … but much more than polyhedra
• Two main (commercial) alternatives
  – All NURBS, all the time
    • Pro/E, SDRC, …
  – Analytic surfaces + parametric surfaces + NURBS + …. all stitched together at edges
    • Parasolid, ACIS, …
Issues in Boundary Representation Solid Modeling

- Very complex data structures
  - NURBS-based winged-edges, etc
- Complex algorithms
  - manipulation, booleans, collision detection
- Robustness
- Integrity
- Translation
- Features
- Constraints and Parametrics
Other Issues: Non-Manifold Solids

• There are cases where you may need to model entities that are not entirely 3D
Cell Decomposition

- Set of primitive cells
- Parameterized
- Often curved
- Compose complex objects by gluing cells together
- Used in finite-element analysis

Foley/VanDam, 1990/1994
Spatial Occupancy Enumeration

- Brute force
  - A grid
- Pixels
  - Picture elements
- Voxels
  - Volume elements
- Quadtrees
  - 2D adaptive representation
- Octrees
  - 3D adaptive representation
  - Extension of quadtrees
Brute Force Spatial Occupancy Enumeration

- Impose a 2D/3D grid
  - Like graph paper or sugar cubes
- Identify occupied cells
- Problems
  - High fidelity requires many cells
- "Modified"
  - Partial occupancy
Distance Volume

• Store signed distance to surface at each voxel

Iso-surface at value 0 approximates the original surface.
Offset Surfaces from Distance Volumes
Quadtree

- Hierarchically represent spatial occupancy
- Tree with four regions
  - NE, NW, SE, SW
  - “dark” if occupied

Foley/VanDam, 1990/1994
Quadtree Data Structure

F = full     P = partially full     E = empty

Quadrant numbering

Foley/VanDam, 1990/1994
Octree

• 8 octants 3D space
  – Left, Right, Up, Down, Front, Back
Boolean Operations on Octrees

\[ S \cup T \quad S \cap T \]

Foley/VanDam, 1990/1994
Adaptive Distance Fields

- Quadtrees/Octrees that store distances
Applications for Spatial Occupancy Enumeration

- Many different applications
  - GIS
  - Medical
  - Engineering Simulation
  - Volume Rendering
  - Video Gaming
  - Approximating real-world data
  - ....
Issues with Spatial Occupancy Enumeration

• Approximate
  – Kind of like faceting a surface, discretizing 3D space
  – Operationally, the combinatorics (as opposed to the numerics) can be challenging
  – Not as good for applications wanting exact computation (e.g. tool path programming)
Binary Space Partition Trees (BSP Trees)

- Recursively divide space into subspaces
- Arbitrary orientation and position of planes
- Homogeneous regions are leafs called in/out cells

Foley/VanDam, 1990/1994
Statistical Representations

- Store density (material vs. void)
- Statistical description of geometry
- Goal – describe the porosity without storing the geometry information

Generated by Termite Agents Simulation.
Stochastic Geometry

• Need some way of converting a solid into some representative statistical form
• From each material voxel, calculate the distance to the nearest voxel that is not material
• Repeat for void voxels
• Store distributions:
  – one for empty space
  – one for material
  – density value
Application: Biological Models

- Bone tissue
- MRI data
- Other biological data
- Solid modeling

MRI scan of left shoulder

Bone matrix from scanned data
## Application: Surface Texture

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>![Texture 26]</td>
<td>![Texture 27]</td>
<td>![Texture 28]</td>
<td>![Texture 29]</td>
<td>![Texture 30]</td>
</tr>
<tr>
<td>![Texture 31]</td>
<td>![Texture 32]</td>
<td>![Texture 33]</td>
<td>![Texture 34]</td>
<td>![Texture 35]</td>
</tr>
<tr>
<td>![Texture 36]</td>
<td>![Texture 37]</td>
<td>![Texture 38]</td>
<td>![Texture 39]</td>
<td>![Texture 40]</td>
</tr>
<tr>
<td>![Texture 41]</td>
<td>![Texture 42]</td>
<td>![Texture 43]</td>
<td>![Texture 44]</td>
<td>![Texture 45]</td>
</tr>
</tbody>
</table>
Application: Surface Texture
Programming Assignment 5

• Extend XPM to 60 different RGB colors
• Read 3 models and assign each a color
• Implement Z-buffer rendering
• Implement front & back cutting planes
  – Only render parts of models between planes
• Implement linear depth-cueing
  – Color = base_color*(z-far)/(near-far)
• Re-use and extend 2D polygon filling
End