Geometric Modeling Systems

- Wireframe Modeling Systems
- Surface Modeling Systems
- Solid Modeling Systems
Wireframe Modeling Systems

Shapes are represented by defining points in space and connecting them by lines

- Simple user input
- Easy software development
- Low computational power demanded
- Ambiguous representation
- No information about solid object and surfaces
- No engineering tools applicable
- Limited usability
Surface Modeling Systems

1. Input points interpolation
2. Input curves nets interpolation
3. Translation and revolution of specified curves

- Used to create models with complex surfaces
- Can be used for NC tool path machining
- Surface connectivity information available

http://www.danplatt.com
Solid Modeling Systems

• Closed volumes are modeled
• Comprehensive solid information can be evaluated or embedded
• Surface information also included
• Best solution for engineering analysis tools
Solid Modeling: Modeling Functions

- Primitive creation functions. Predefined simple shapes

- Boolean operators
Solid Modeling: Modeling Functions

- Sweeping: translational or rotational

- Skinning: multiple cross section perimeter interpolation

- Rounding
Solid Modeling: Modeling Functions

- Lifting: use a portion of a surface for lengthening part of the solid.

- Boundary modeling

  VERTEX REPLACEMENT
  EDGE REPLACEMENT
  SURFACE REPLACEMENT
Solid Modeling: Modeling Functions

- Feature-based modeling: similar to primitive, but more complex and specialized for manufacturing, like “Hole”, or “Slot”, or “Pocket”...

- Parametric modeling: use relationships and constraints
Data Structure

- **Constructive Solid Geometry (CSG) representation**
  - Application of boolean operations on primitives and stored in a tree type of data structure

- Data structure is simple and compact
- The solid stored is always valid
- Easy parametric implementation
- Only boolean operators allowed: shape definition severely limited
- It requires lots of computation to derive information of boundary surface, edges, connectivity
Data Structure

• Boundary Representation (B-Rep)
  – Boundary information collected in a tree table:

<table>
<thead>
<tr>
<th>Face</th>
<th>Edges</th>
<th>Edge</th>
<th>Vertices</th>
<th>Vertex</th>
<th>Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>F₁</td>
<td>E₁, E₂, E₅</td>
<td>E₁</td>
<td>V₁, V₂</td>
<td>V₁</td>
<td>x₁, y₁, z₁</td>
</tr>
<tr>
<td>F₂</td>
<td>E₂, E₆, E₇</td>
<td>E₂</td>
<td>V₂, V₃</td>
<td>V₂</td>
<td>x₂, y₂, z₂</td>
</tr>
<tr>
<td>F₃</td>
<td>E₃, E₇, E₈</td>
<td>E₃</td>
<td>V₃, V₄</td>
<td>V₃</td>
<td>x₃, y₃, z₃</td>
</tr>
<tr>
<td>F₄</td>
<td>E₄, E₈, E₅</td>
<td>E₄</td>
<td>V₄, V₁</td>
<td>V₄</td>
<td>x₄, y₄, z₄</td>
</tr>
<tr>
<td>F₅</td>
<td>E₅, E₉, E₆</td>
<td>E₅</td>
<td>V₅, V₃</td>
<td>V₅</td>
<td>x₅, y₅, z₅</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E₆</td>
<td>V₂, V₃</td>
<td>V₆</td>
<td>x₆, y₆, z₆</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E₇</td>
<td>V₃, V₅</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>E₈</td>
<td>V₄, V₅</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  – Works only with planar polyhedra
  – It needs a workaround for faces with inner and outer boundaries
  – Deriving connectivity information from the tables can be cumbersome
  – Variation from B-Rep uses double linked lists: **half-edge data structure**
Data Structure

- **Winged-edge data structure**
  - Similar concept to half edge data structure (B-rep) but instead of being centered on faces definition it is based on edges definition:
Data Structure

- **Decomposition Model Structure**
  - Voxel Representation: it is a 3D extension of a raster representation: Voxel stands for Volume-Pixel
  - Easy to represent arbitrary shapes
  - Modification of this type of data structure is very difficult
  - Easy to evaluate solid properties
  - Memory usage dramatically incremented with size (or resolution)
  - Voxel representation is inherently an approximation of the original solid
Data Structure

• Decomposition Model Structure
  – Octree: similar to Voxel decomposition but volume in which the solid is enclosed is iteratively subdivided based on if the current octant contains some of the defined geometry (grid-based mesh generation approach)
Data Structure

- Cell Representation: similar to voxel but each cell can have an arbitrary shape.
B-Rep Data Structure Modification: Euler Operators

- B-Rep data structure modification is not as straightforward as for CSG.
- Euler-Poincare formula for topology entries:

\[ v-e+f-h=2(s-p) \]

\( v \) = number of vertices  
\( e \) = number of edges  
\( f \) = number of faces  
\( h \) = number of hole loops  
\( s \) = number of shells  
\( p \) = number of passages

Example:

\[
16 - 24 + 10 - 2 = 2(1 - 1)
\]
B-Rep Data Structure Modification: Euler Operators

Since the topology entries are not linear independent, Euler operators are introduced to handle linked modifications among topologies.

<table>
<thead>
<tr>
<th>MEV VLS (KEV VLS)</th>
<th>make (kill) edge, two vertices, loop, shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEL (KEL)</td>
<td>make (kill) edge, loop</td>
</tr>
<tr>
<td>MEV (KEV)</td>
<td>make (kill) edge, vertex</td>
</tr>
<tr>
<td>MVE (KVE)</td>
<td>make (kill) vertex, edge</td>
</tr>
<tr>
<td>MEK H (KEMH)</td>
<td>make (kill) edge, kill (make) hole</td>
</tr>
<tr>
<td>MZEV (KZEV)</td>
<td>make (kill) zero length edge, vertex</td>
</tr>
<tr>
<td>MPK H (KPMH)</td>
<td>make(kill) peripheral loop, kill (make) hole loop</td>
</tr>
</tbody>
</table>

![Diagram showing examples of Euler operators and their effects on topologies.](image)
B-Rep Data Structure Modification: Boolean Operators

• Also in B-Rep data structure Boolean operators are not as straightforward as for CSG.

• Method for 2D (3D similar but more complex):
  – define face A and face B on which to apply the Boolean operator
  – Find face C
  – Find all edges (including edges of C)
  – Classify all edges according to each face:
    • inside, on outside A; inside, on outside B
  – Collect edges based on Boolean:
    • Union: reject “in A” & “in B”
    • Intersection: reject “out of A” & out of B”
    • Subtraction: reject “on A” & “in B” and “out of A” & “in B”