Solid models

- Solid models developed to address limitations of wireframe modeling.
- Attempt was to create systems which create only “complete” representations.
- Modelers would support direct creation of 3D geometry

Solids as “point set”

- “Real world” solid objects may be thought of as a set of points in Euclidean 3D space which conform to our concept of “solid”.
- Solid modelers encode this infinite point set in a format compatible with computer storage.
- Based upon this concept, we can divide modeling representation schemes into 3 general classes.

Solid representation schemes

1) constructive models
   - point set represented as a combination of more basic primitive point sets
   - each primitive point set is an instance of a defined primitive solid
   - first efforts were called primitive instancing
   - later developed into: Constructive Solid Geometry (CSG)

Solid representation schemes (cont’d)

2) spatial enumeration models
   - In this scheme solids are decomposed into cells each with a simple topological structure and often with a simple geometric structure.
   - For example, define solid point set in terms of collection of non-overlapping “blocks”

Solid representation schemes (cont’d)

3) boundary models (B-Rep)
   - point set defined in terms of its boundaries
   - boundary models have a hierarchical format
     - volume defined by set of faces
     - faces defined by a set of finite curves

Review of Set Theory

- Definitions of:
  - Sets
  - Set properties and operations
  - Properties of set operations
## Sets
- A set is a well-defined collection of elements.
  - Objects belonging to a set are its elements
- Sets may be finite or infinite (# of elements)
- We will be dealing with sets of points (points in 3D space)

## Set Properties
- A set containing all the elements under consideration is the *universal set*.
- Two sets which contain the same elements are *equal sets*.
  \[ A = B \]
- If all the elements of set B are contained in set A, then set B is a *subset* of set A
  \[ A \subset B \]

## Set Operations
- Basic operations
  - Complement
  - Union
  - Intersection
  - Subtraction

### Complement
- Given:
  - \( E \) is the universal set
  - A set A such that \( A \subset E \)
  - The complement of A (written \( A^c \)) is the set of all elements of \( E \) not in \( A \)

### Union
- Given two defined sets A and B
- A *union* B (written \( A \cup B \)) is the set of all elements in set A or Set B (logical “or”)

\[
\text{Union} \\
A = \{a,b,c\} \quad B = \{c,d,e,f\} \\
A \cup B = \{a,b,c,d,e,f\} \\
\text{Notice there is no repetition of common elements.} \\
\text{union operations are commutative and associative} \\
A \cup B = B \cup A \\
(A \cup B) \cup C = A \cup (B \cup C)
\]
Intersection

- A \textit{intersection} \( B \) (written \( A \cap B \)) is the set of all elements in both set \( A \) and Set \( B \). (logical “and”)

\begin{align*}
\text{Intersection} \\
\text{• Intersection operations are commutative and associative:} \\
A \cap B &= B \cap A \\
(A \cap B) \cap C &= A \cap (B \cap C)
\end{align*}

Subtraction (Difference)

- the elements in a source set \textbf{not} also in a second set. (logical “not”)

\begin{align*}
\text{Subtraction} \\
\text{• Sets must share common elements else produces error (null set)} \\
\text{• Subtraction is not associative or commutative} \\
\text{BE Careful:} \\
(A \cup A) - A &= 0 \\
\text{But} \\
A \cup (A - A) &= A
\end{align*}

CSG modeling

- objects are represented as a combination of simpler solid objects referred to as “\textit{primitives}”.
- Copies or “instances” of these primitive shapes are created and positioned.
- a complete solid model is constructed by combining these “instances” using set specific, logic operations (Booleans).

Boolean Operations

- operations which are used to combine primitives (geometric objects)
- may be used to combine:
  - valid primitive instances
  - solids resulting from previous Boolean operations with each other or other instances
Boolean Operations

• Set operations union, intersection and subtraction are the Booleans
• Booleans for our use are “regularized”
• Must produce as output objects having the same dimensionality as the input
• Boolean Operations
  – are intuitive to user
  – are easy to use and understand
  – provide for the rapid manipulation of large amounts of data

Boolean Operations

Set membership

• Critical to the use of Boolean operations is the ability to perform set membership tests.
• Set membership: Does a given location in 3D space lie within the bounds of one of the two sets upon which the operation is performed.

Typical CSG Primitive Set

Geometric Objects

• Must be closed sets (finite)
• Must have a defined boundary and defined interior subsets
• (must be true for CSG primitives and the objects used in other modeling techniques)

CSG Primitives

• Primitives themselves defined by combining lower-level entities
  – primitives are the Boolean combination of unbounded geometric entities known as “half-spaces”
• Half-space definitions separate 3D space into infinite regions (solid and void)

Half space

Example:
Divides 3D space into “all points greater than or equal to $Y_1$, and all points less than $Y_1$”
CSG Primitives

- Primitive definition through half-spaces allows for set membership evaluation
- Any point within modelspace can be defined as existing:
  - within a solid
  - in the void (outside the solid)
  - or on the border in between.
  - e.g. determine if point is a member of primitive point set

Box Primitive defined by bounding half-spaces. User defined input is corner location (origin) and Width, Height and Depth values (W, H, D)

Cylinder Primitive. User inputs are location (origin), Radius and Height. Half-space are all points greater then or equal to zero and less than or equal to “H”, along with all points(X^2+Y^2) less than or equal to R^2

CSG Data Structure

- Data structure does not define model shape explicitly but rather implies the geometric shape through a procedural description.
  - example: object is not defined as a set of edges & faces but by the instruction: union primitive 1 with primitive 2
- This procedural data is stored in a data structure referred to as a CSG tree.

CSG Models are not unique.

- More than one procedure (and hence database) can be used to arrive at the same geometry.
- Therefore, CSG representation not unique (expressive)
- The following illustrates this concept in a simple 2D example

Same object with different databases

+ - =
CSG Tree

- Model data for CSG model can be represented by a tree structure
  - CSG tree provides a procedural description of the model in a binary tree format.
  - The outer leaf nodes of tree represent the primitives.
  - The interior nodes represent the Boolean operations performed.
  - The following shows a graphical representation of the tree structure

CSG Representation

- CSG representation is **unevaluated**
  - Faces, edges, vertices not defined in explicit (mathematical) form
- CSG models are **always valid**
  - since built from solid elements
- CSG Models are **complete and unambiguous**

CSG data structure/storage

- Display data not actively stored
  - The edges, faces, and vertices of the model are *implied* rather being *explicitly defined* in the data structure.
  - Vertices, edges, etc. are defined as needed

Boundary evaluation for display:

- The unevaluated format of CSG representation does not store a display model
- a boundary evaluation must be performed to determine the edges of the model in order for a model display to be created

CSG Models

- Domain and expressive power
  - Domain is limited by the primitives available
  - primitives limited by half-spaces used to define them.
    - typically planar, quadratic, toroidal
    - also implicit combinations of these may approximate sculptured surfaces
CSG Models

• Validity
  – provided primitives themselves are valid, combinations of primitives will be valid

• Uniqueness
  – tree will unambiguously model a solid
  – trees are not unique descriptions
    • not necessarily a bad thing, but realize it is true

CSG Modeling

• CSG is powerful, with high level commands
• CSG models created with a minimum of steps
• CSG modeling techniques lead to a concise database.
  – Complete history of model is retained, and can be altered at any point.

Who uses CSG?

• Most of the model systems currently popular in industry do not use strict CSG model definitions.
• However, many of these systems do use similar constructions techniques.
  – Many provide the user with a library of basic 3D shapes.

Who Uses CSG

• Many systems provide the user with “Boolean type” commands for combining solids. This is because Booleans:
  – are easy to implement
  – tend to be intuitive
  – relate well to fabrication

Who uses CSG?

• The Army Research Lab does use a CSG system called: BRL-CAD
• Here is a simple CSG Java Applet
  http://www.nbb.cornell.edu/neurobio/land/OldStudentProjects/cs490-96to97/indira/csg/doc.html

Some Links

• Here are a couple of interesting links… there are many more if you search “constructive solid geometry”
  http://astronomy.swin.edu.au/pbourke/geometry/fittoy/
  http://www.iem.pw.edu.pl/~sawickib/artykul1/