

Geometric Modeling System

Human-centered CAD Lab.

Geometric modeling system

- ▶ Software enabling shape creation and visualization in the design process
- ▶ Designer realizes the shape in his mind while the shape data are stored inside
 - ▶ Wireframe Modeling System
 - ▶ Surface Modeling System
 - ▶ Solid Modeling System
 - ▶ Non-manifold Modeling System

History of Geometric Modeling

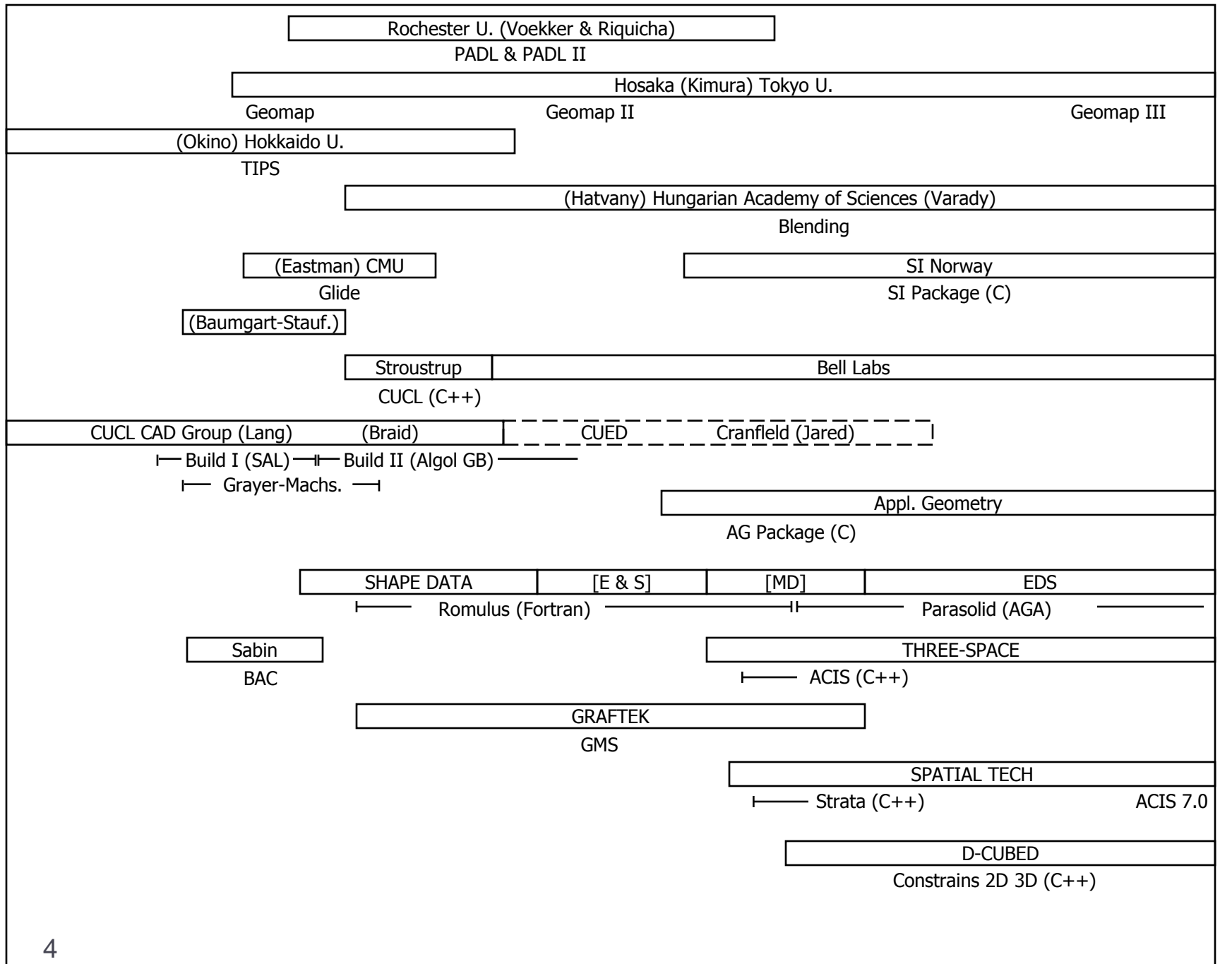
- ▶ **Tips**

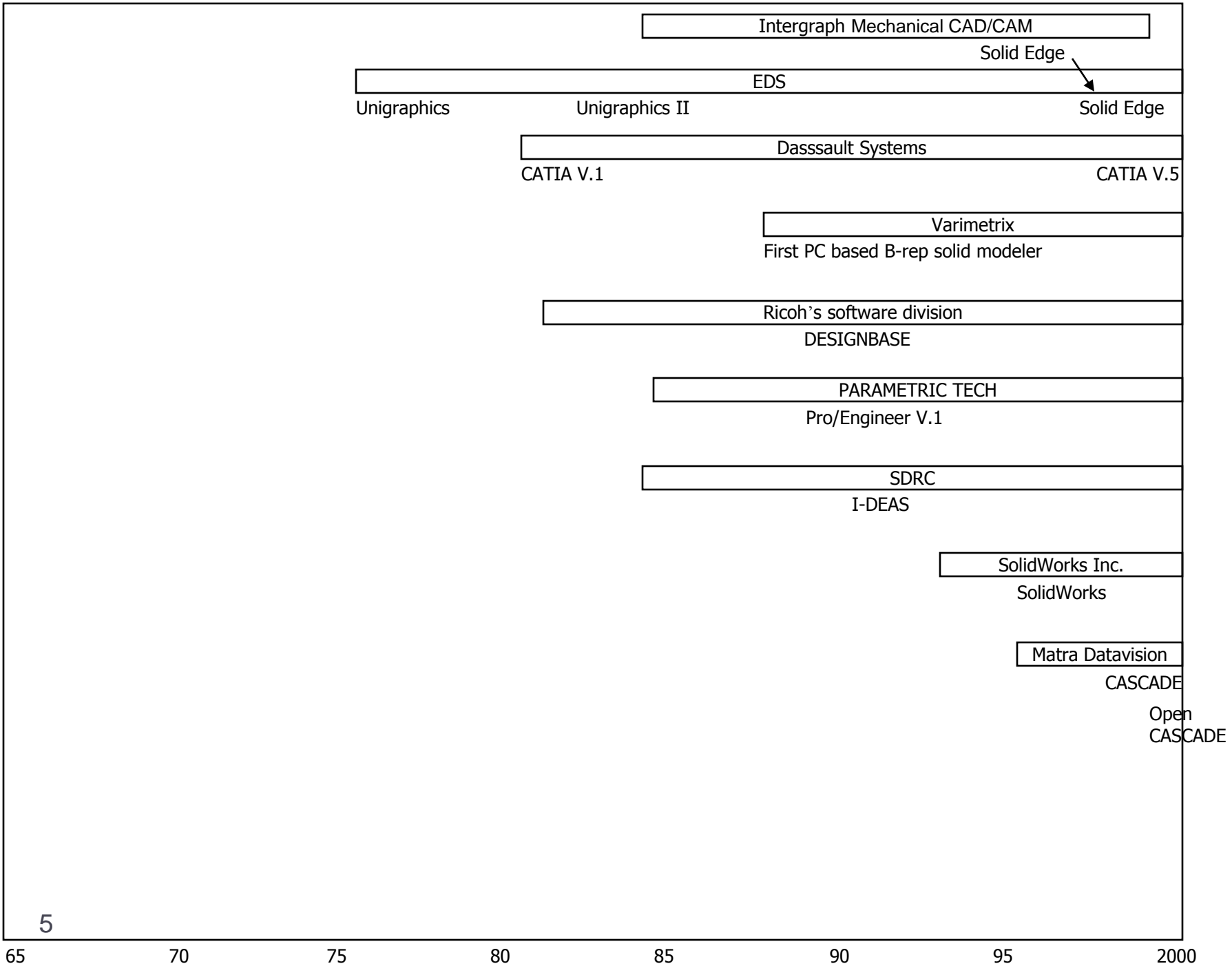
- ▶ Okino, Kubo at Hokaido University, 1973
- ▶ Constructive Solid Geometry (CSG)

- ▶ **Build**

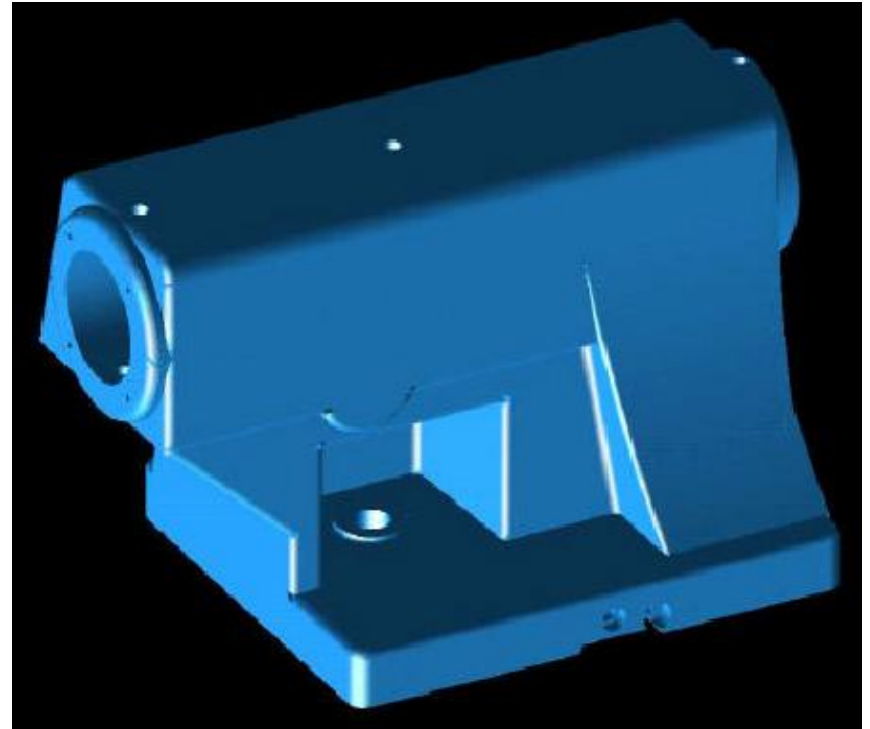
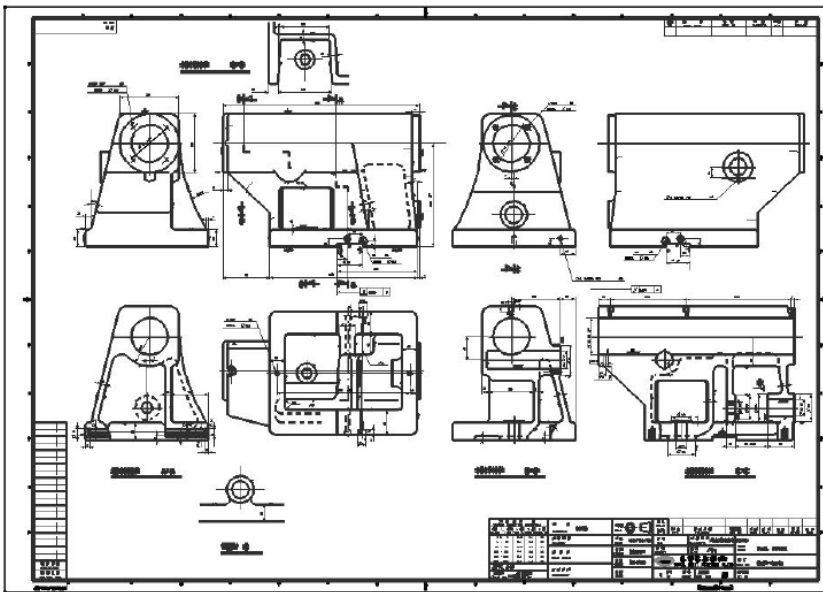
- ▶ Braid, Lang at Cambridge University, 1973
- ▶ Boundary Representation (B-rep)

- ▶ CADAM, Unigraphics, CATIA, I-DEAS, BRAVO, ME10/30, Pro/ENGINEER, DesignBASE, SolidEdge, SolidWorks, ...





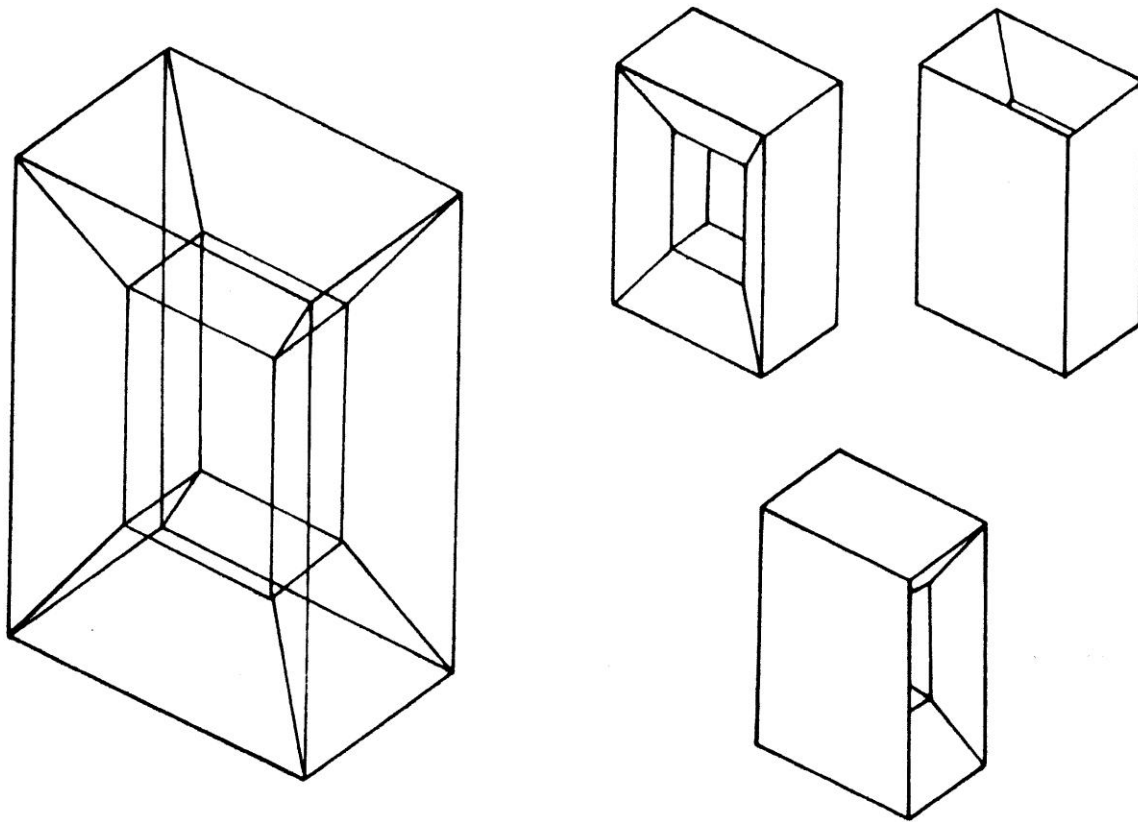
Why 3 Dimensional Model?



Wireframe Modeling System

- ▶ User inputs characteristic points and curves
- ▶ Good for simple visualization
- ▶ Ambiguous situations may occur
- ▶ Impossible to automatically calculate mass properties, NC tool paths, and finite elements

Ambiguous wireframe models



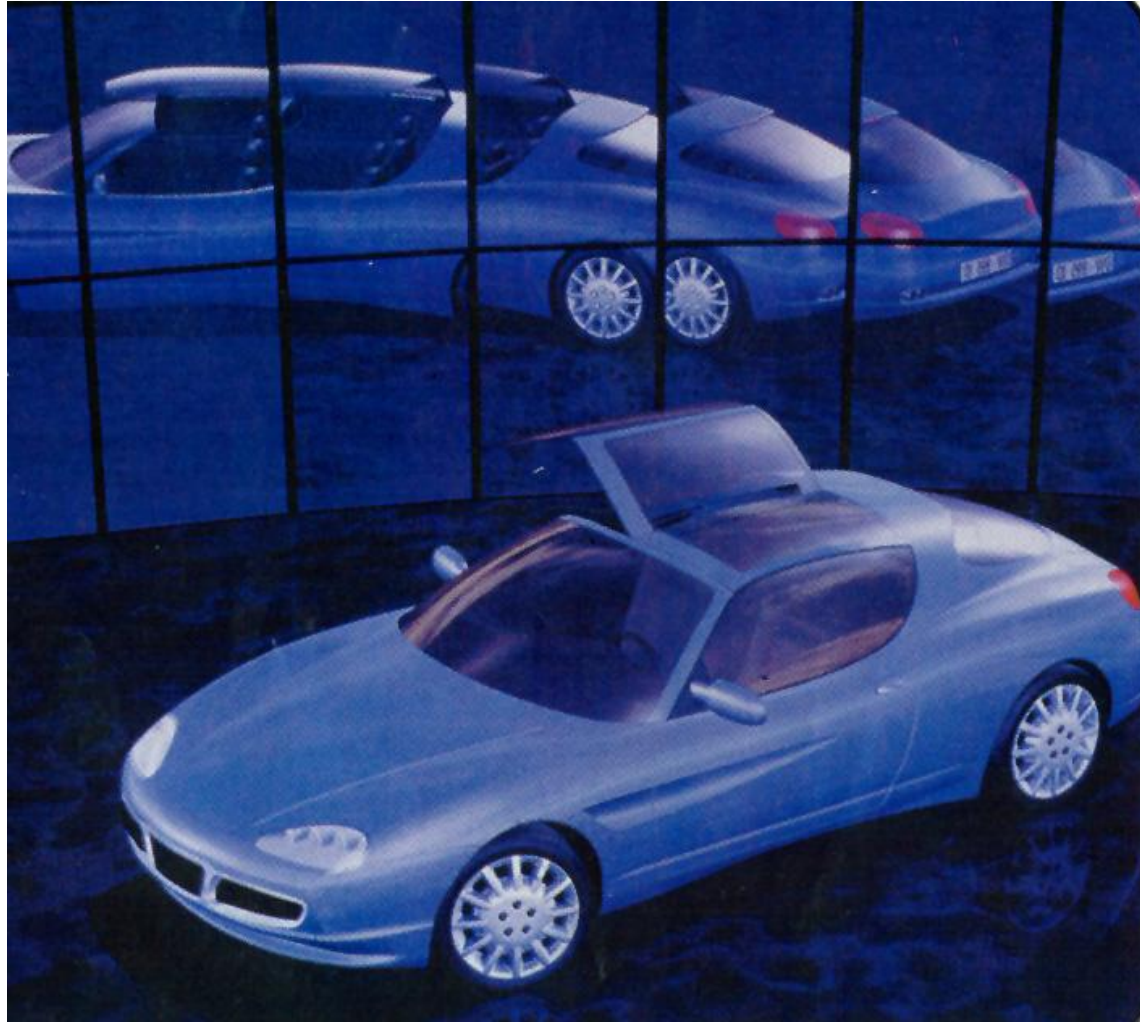
Surface Modeling System

- ▶ Surface information in addition to wireframe model
- ▶ Usually user specify the curves on a surface, then system stores the surface equation
- ▶ Adjacency information between surfaces are not stored in general
- ▶ Intersection calculation is needed to derive the boundary curves
- ▶ Some surface modeling systems store boundary curves also

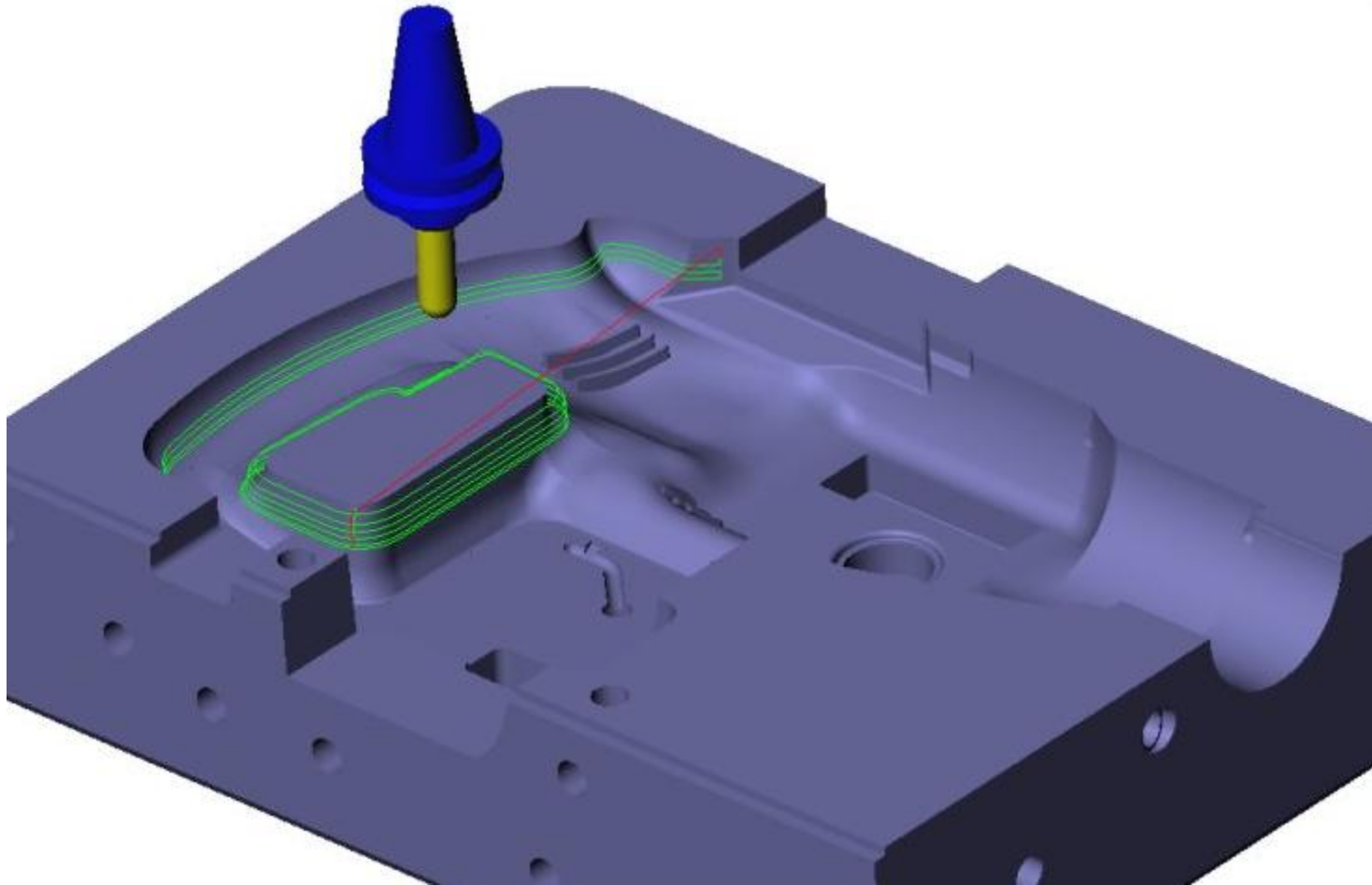
Surface Modeling System – cont'

- ▶ Point set
- ▶ Curve net
- ▶ Curve movement (Sweeping, Skinning)
- ▶ Good for aesthetic evaluation, Styling CAD
- ▶ Input for NC tool path generation
- ▶ Good for modeling object bounded by complicated surfaces

Modeling of automobile body by surface modeling system



Calculation and verification of NC tool paths



Solid Modeling System

- ▶ Adjacency information between faces, and inside-outside information of each face are stored in addition
- ▶ Volume inside modeled object is defined
- ▶ Volumetric operations are possible
 - ▶ Automatic generation of solid elements for FEA
 - ▶ Automatic generation of tool paths for rough cut

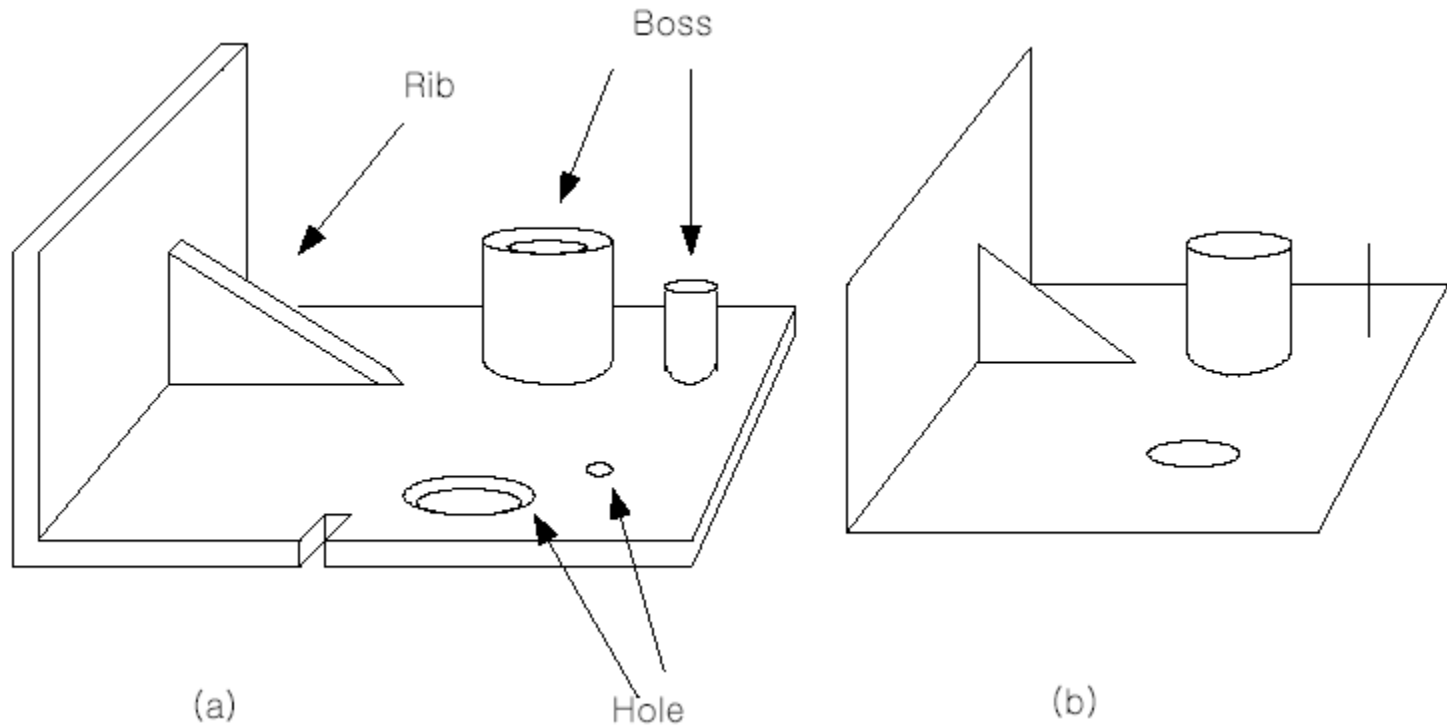
Solid Modeling System – cont'

- ▶ Partial modeling is not allowed, complete solid model should be made
- ▶ More modeling tasks
- ▶ Many convenient modeling commands are provided
 - ▶ Face adjacency, in-out information, etc. are generated by the system

Non-manifold Modeling System

- ▶ Accommodate all different levels of geometric model
 - ▶ Wireframe model : Wireframe modeling system
 - ▶ Surface model : Surface modeling system
 - ▶ Solid model : Solid modeling system
- ▶ Models of mixed dimension, incomplete models are allowed (support design process, analysis model)

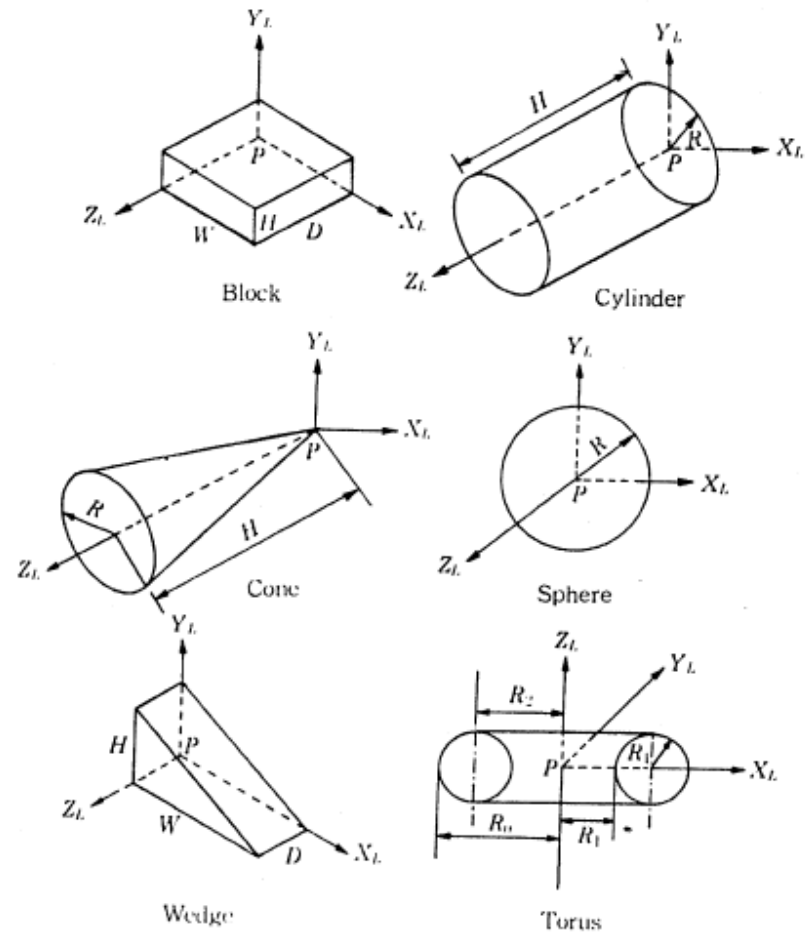
Non-manifold Modeling System



Modeling Functions

(1) Primitives Creation

- ▶ Retrieves a solid of a simple shape
- ▶ Primitives are stored by the procedures how they are created.
- ▶ Parameters specifying the size are passed to the corresponding procedure as arguments.



Primitives generally supported

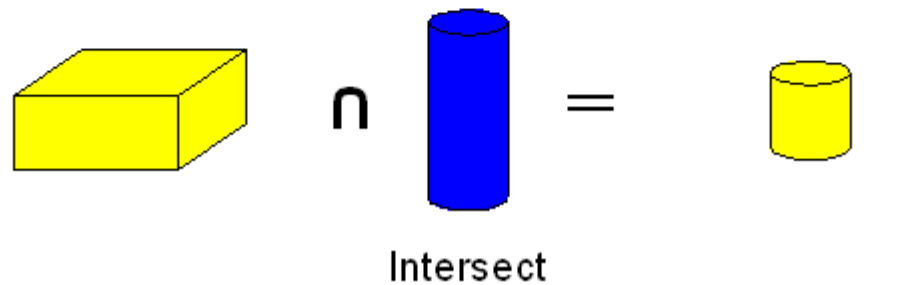
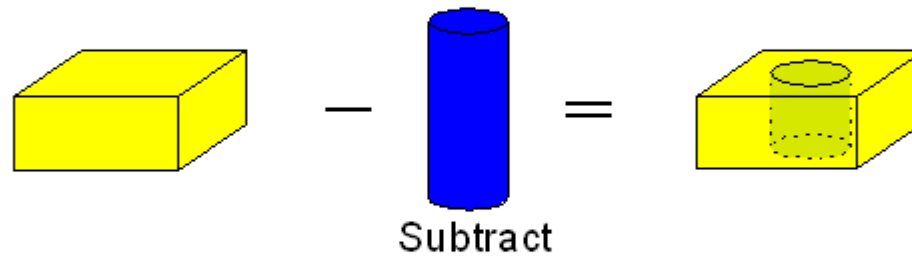
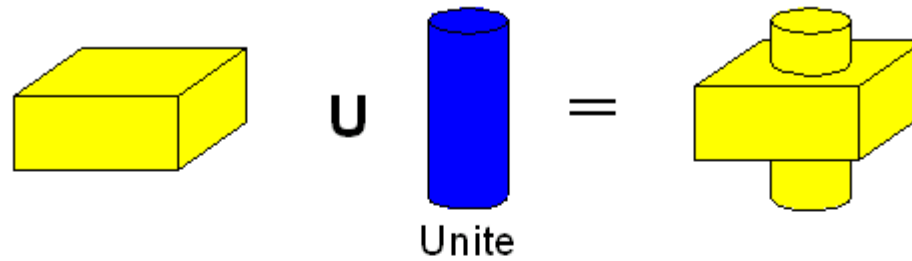
Modeling Functions

(2) Boolean operation

- ▶ Primitive solid is assumed to be a set of points
- ▶ Boolean operation is performed between the point sets
- ▶ The result is the solid composed of the points resulting from the set operation.

Modeling Functions

(2) Boolean operation

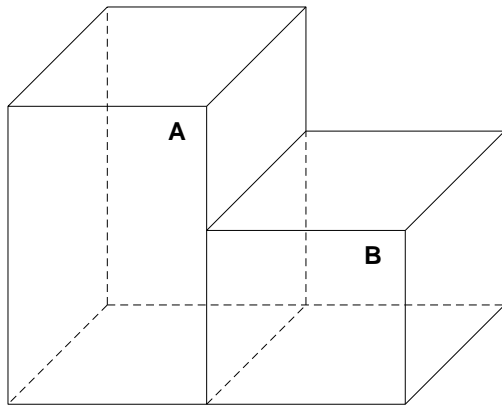


Modeling Functions

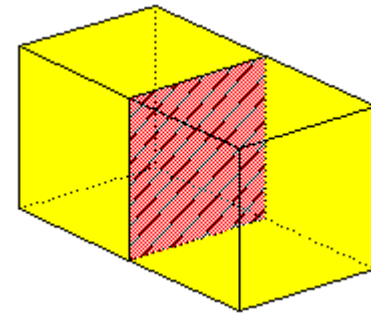
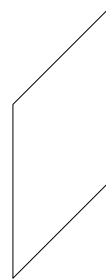
(2) Boolean operation

- ▶ Boolean operation may result an invalid solid
- ▶ Non-manifold modeling systems can handle Boolean operations between objects of mixed dimension.

Example of Boolean operation to be avoided

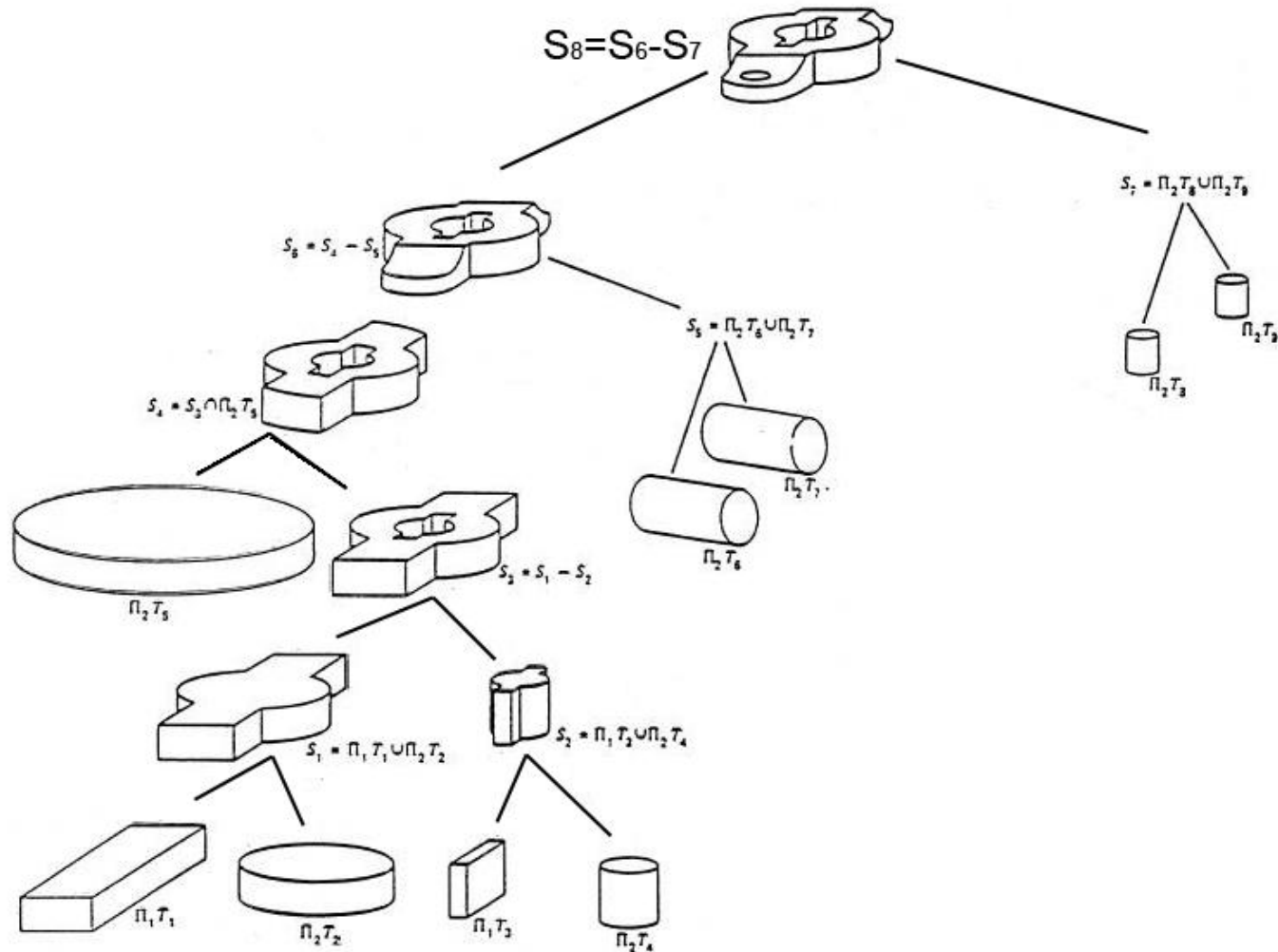


$A \cap B$
→



$A \cap B$

Example of modeling in CSG approach



Modeling Functions

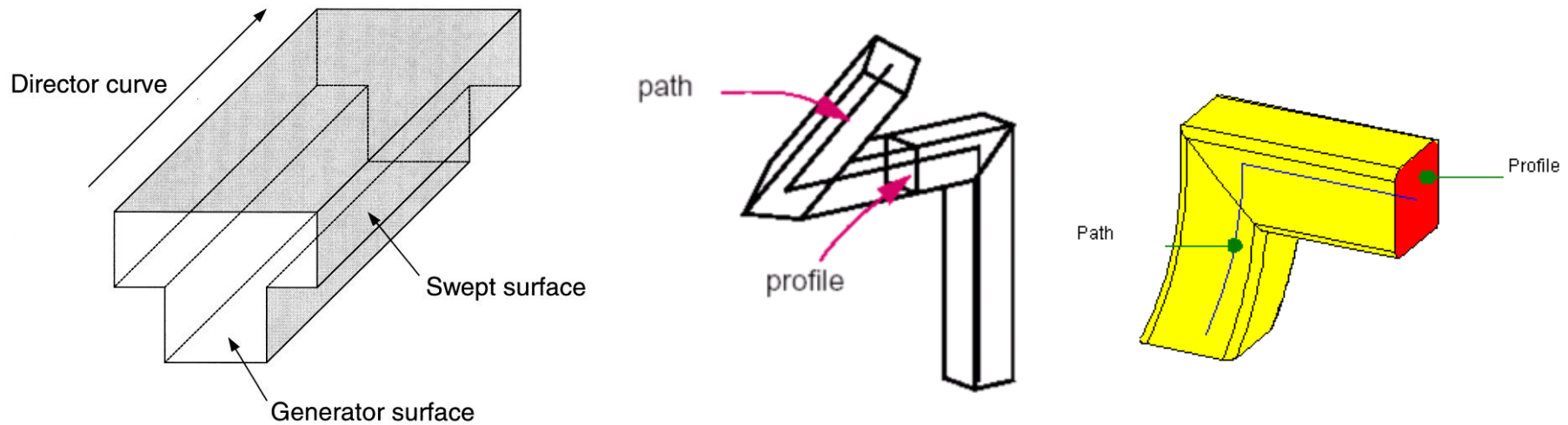
(3) Sweeping

- ▶ Planar closed domain is translated or revolved to form a solid
- ▶ When the planar shape is not closed, the result is a surface
 - ▶ Used in surface modeling system

Modeling Functions

(3) Sweeping – Example.1

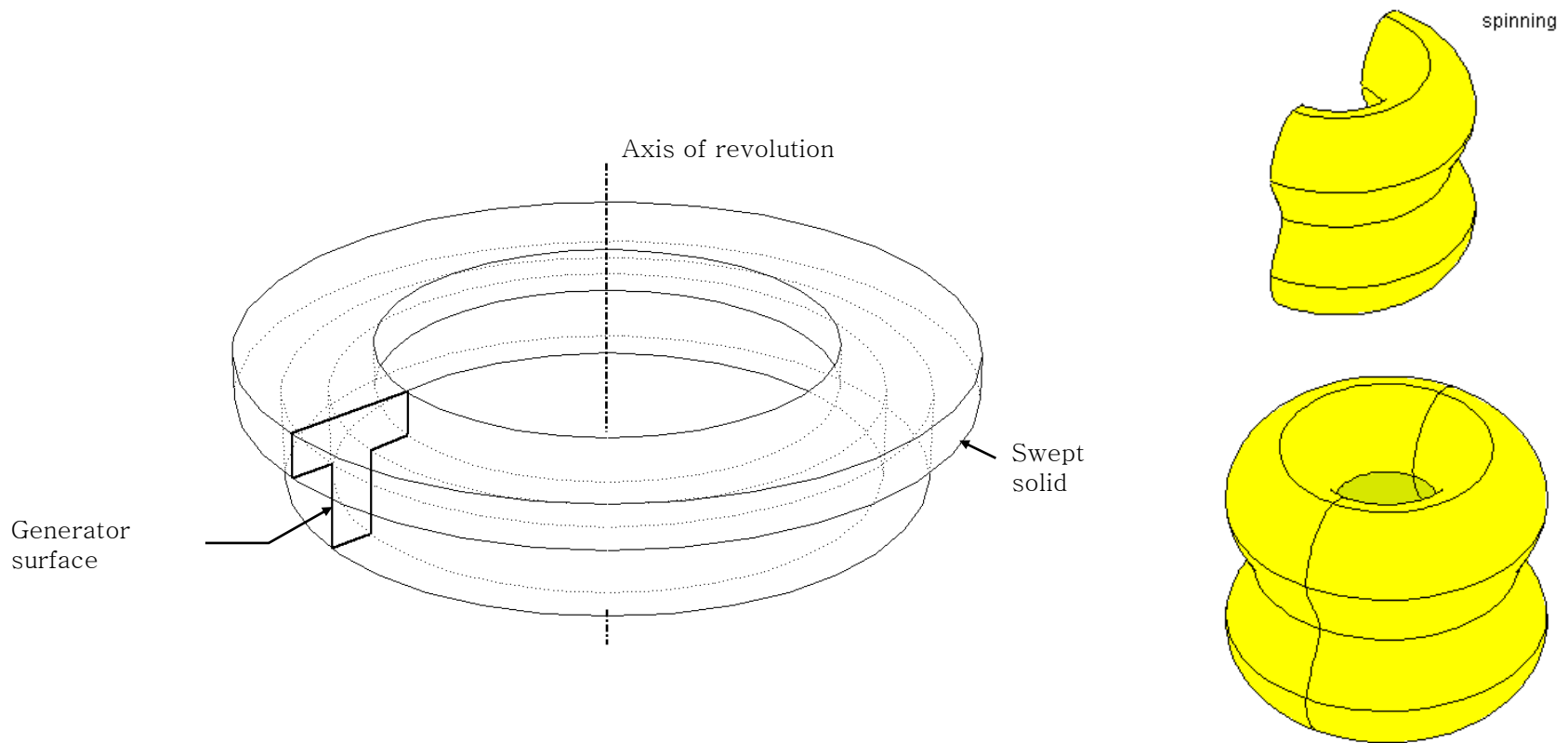
▶ Example of translation sweeping



Modeling Functions

(3) Sweeping – Example.2

▶ Example of rotational sweeping



Modeling Functions

(4) Skinning

- ▶ Form a closed volume by creating a skin surface over pre-specified cross sectional planar curves
- ▶ If two end faces corresponding to the two end cross sections are not added, the result would be a surface
 - ▶ Used in surface modeling system

Modeling Functions

(4) Skinning (Lofting) - Example

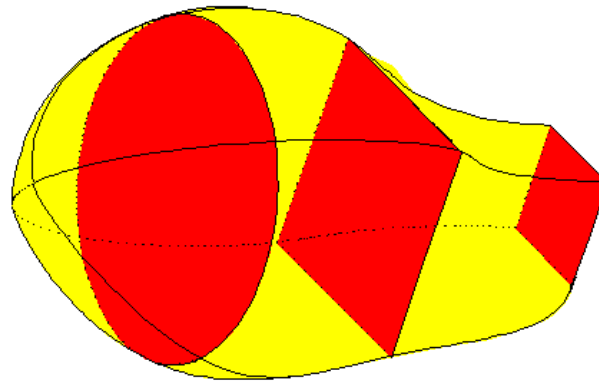
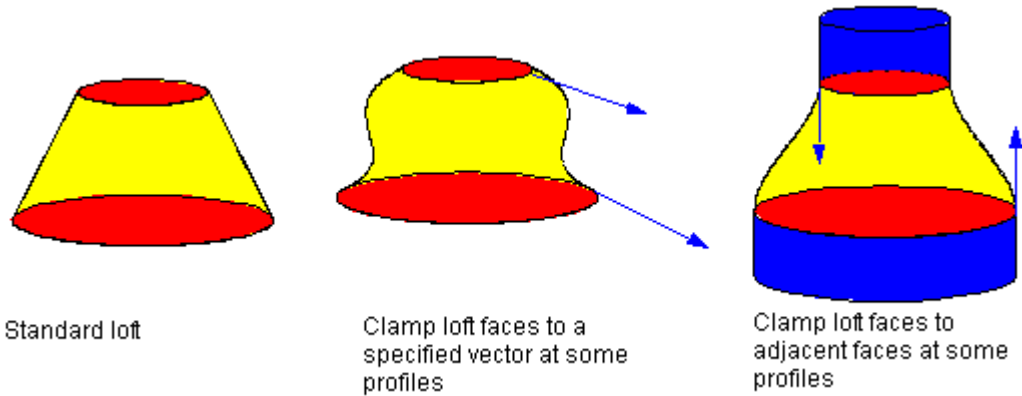


Figure 8-6 Creating a lofted body using several different profiles

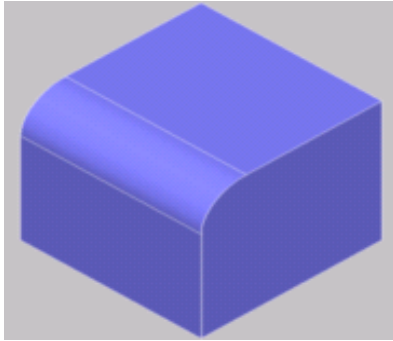
Modeling Functions

(5) Blending

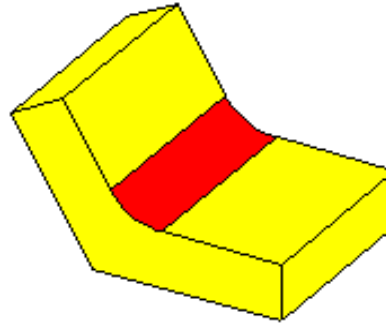
- ▶ Sharp edge or sharp vertex is replaced by a smooth curved surface
- ▶ Normal vector is continuous across the surfaces meeting at the original sharp edge or vertex

Modeling Functions

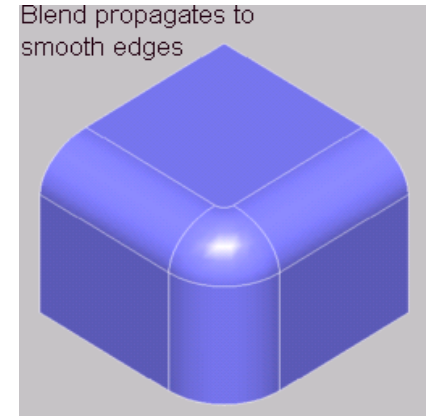
(5) Blending – Example



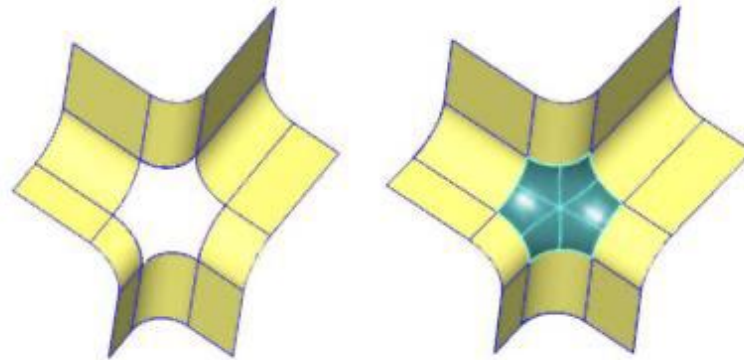
Edge rounding



Edge filleting



Vertex rounding

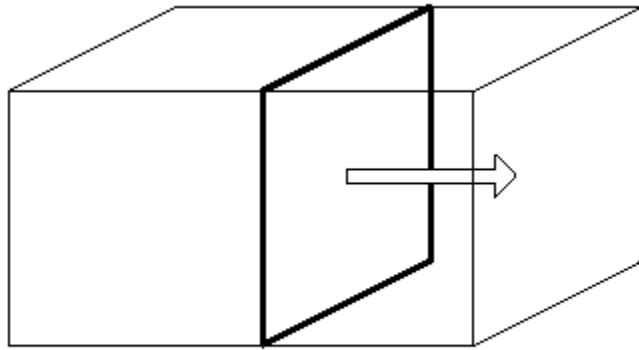


complex intersecting blends

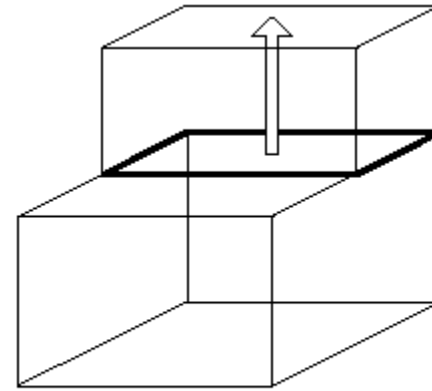
Modeling Functions

(6) Lifting

- ▶ Pull a portion or whole face of a solid



(a)



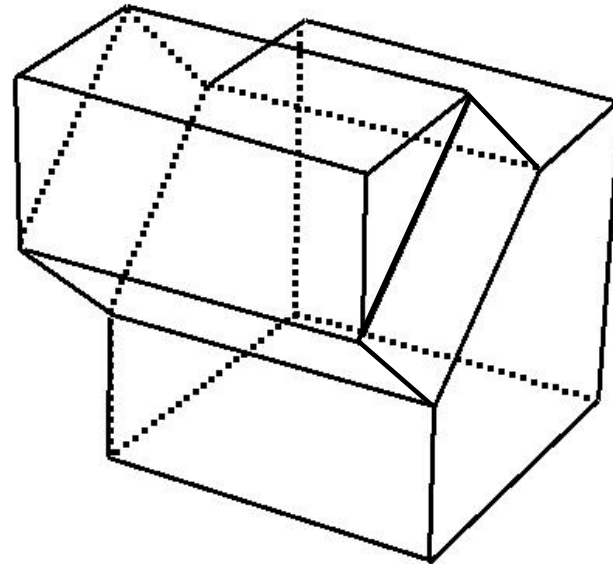
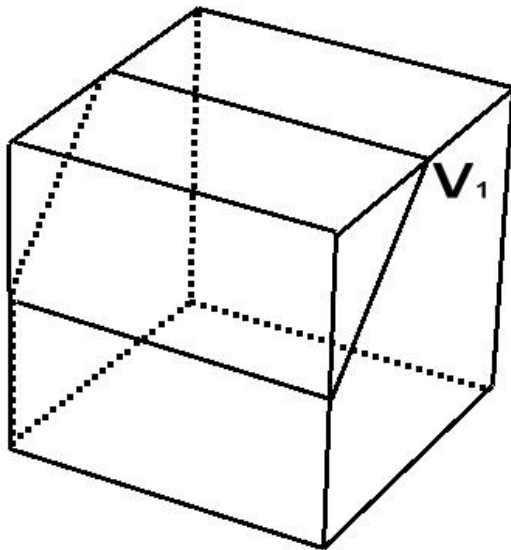
(b)

Example of lifting

Modeling Functions

(6) Lifting

► Face lifting



Modeling Functions

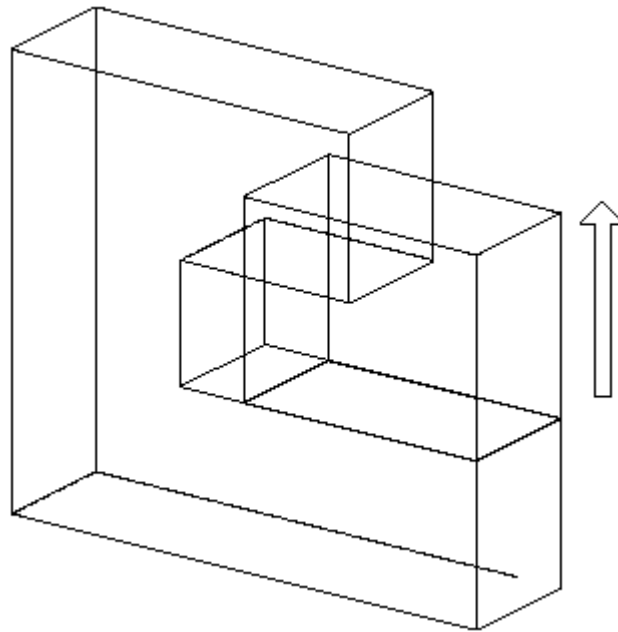
(6) Lifting

- ▶ When a portion of a face is lifted, the face should be split beforehand
 - ▶ Add a splitting edge
 - ▶ Update face connectivity
 - ▶ Update edge adjacency, ...
- ▶ Euler operators will handle these tasks

Modeling Functions

(6) Lifting

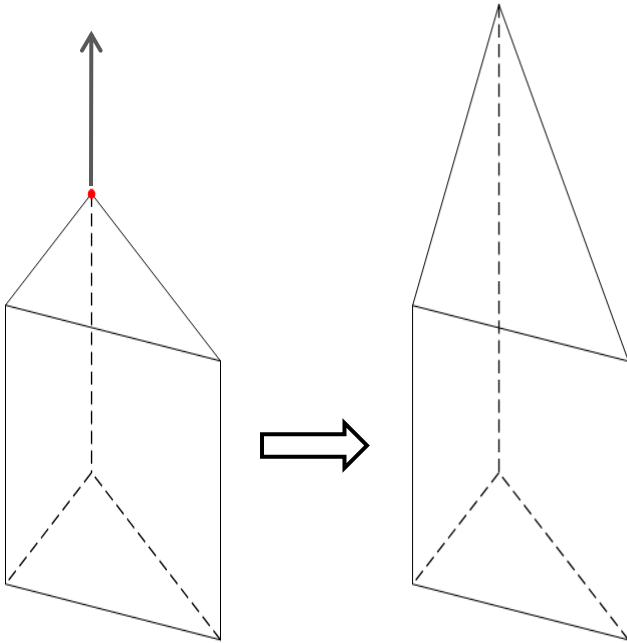
- ▶ Self interference caused by lifting



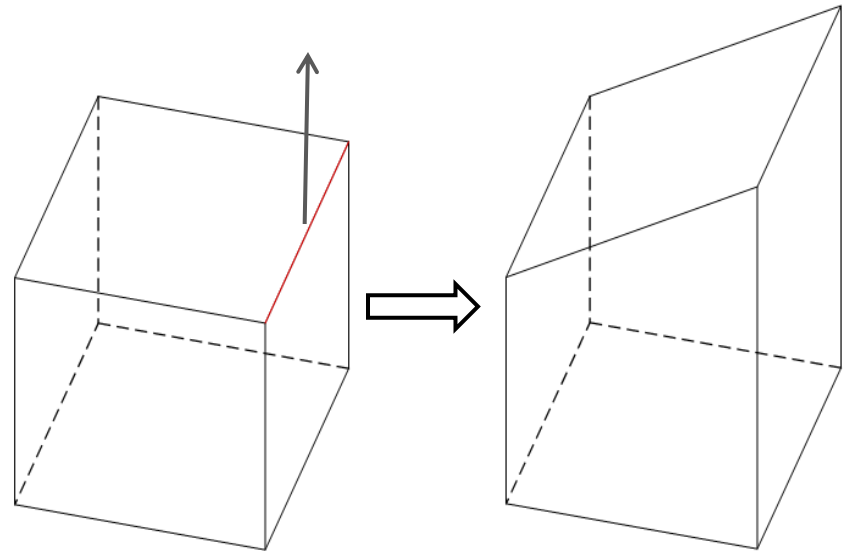
Modeling Functions

(7) Tweaking

▶ Vertex Tweaking

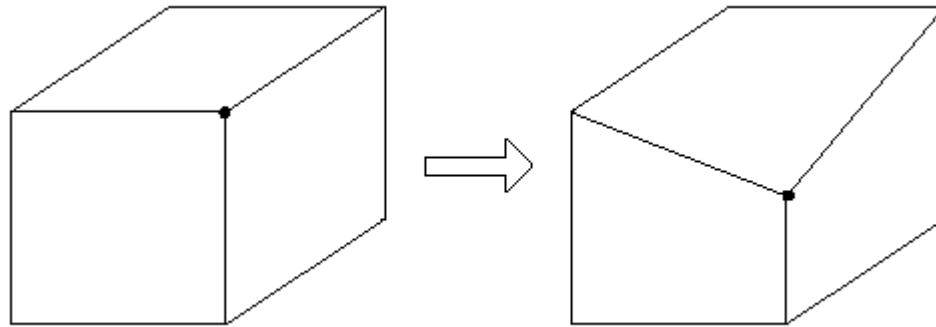


▶ Edge Tweaking

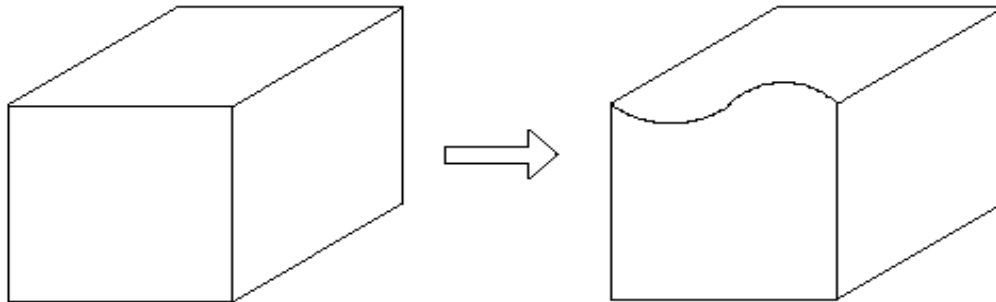


Modeling Functions

(7) Tweaking



Modification by vertex moving

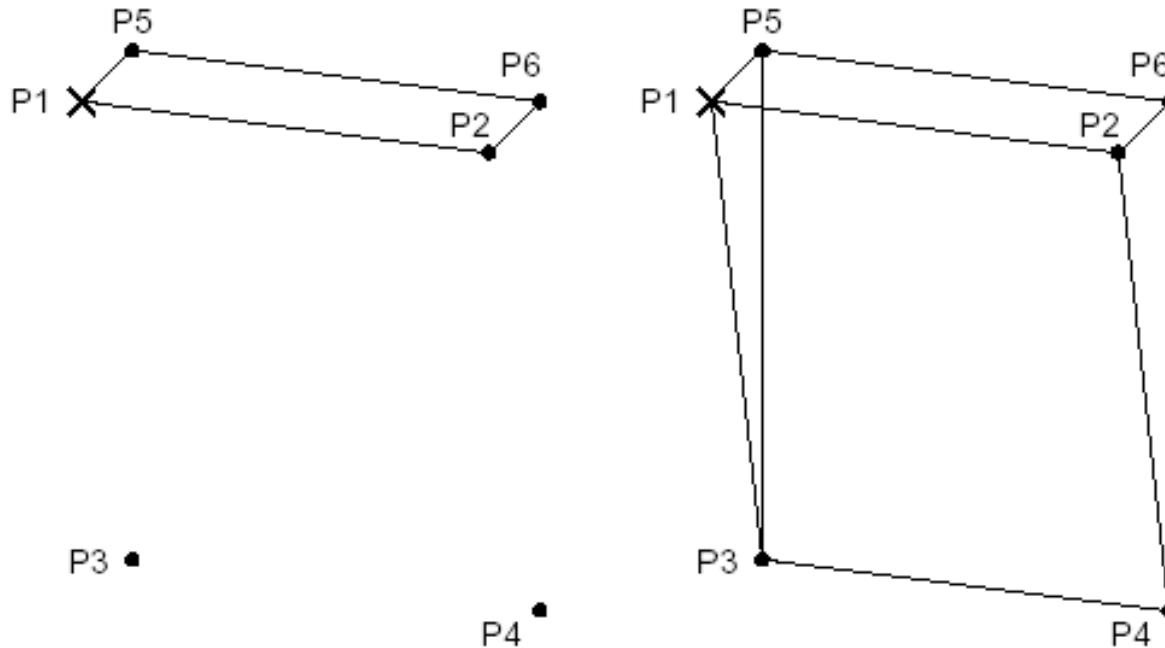


Modification by edge replacement

Modeling Functions

(8) Boundary Modeling

- ▶ Add, delete, modify entities such as vertices, edges, and faces directly



Modeling Functions

(8) Boundary Modeling

- ▶ Very tedious operation
- ▶ Boundary modeling functions are mainly used to create only up to two dimensional shapes which are used for sweeping or skinning
- ▶ Can be effectively applied to modify a shape of an existing solid
 - ▶ Tweaking operation

Modeling Functions

(9) Feature based modeling

- ▶ Let designers model a solid by the shape units familiar to them
- ▶ The resulting solid carries the information on the existence of these shape units in addition to the elementary shape entities such as vertices, edges, faces, etc.

Modeling Functions

(9) Feature based modeling

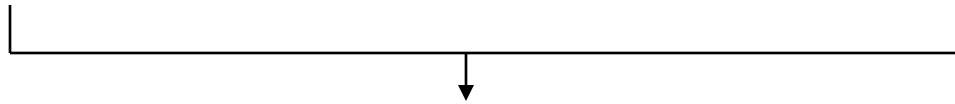
- ▶ E.g.
 - ▶ ‘ Make a hole of a certain size at a certain place ’
 - ▶ ‘ Make a chamfer of a certain size at a certain place ’
 - ▶ Existence of hole and chamfer is added to model information
- ▶ Set of features varies depending upon the frequent applications of the system

Modeling Functions

(9) Feature based modeling

- ▶ Popular feature

chamfer, hole, fillet, slot, pocket, ...

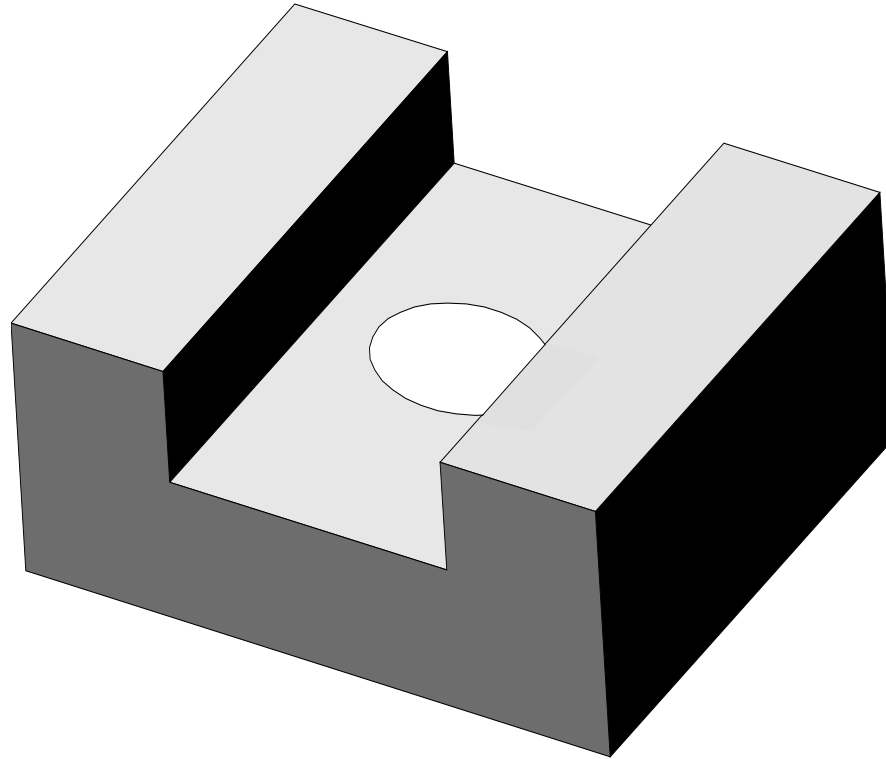


manufacturing features

These features can be matched to
a specific machining process

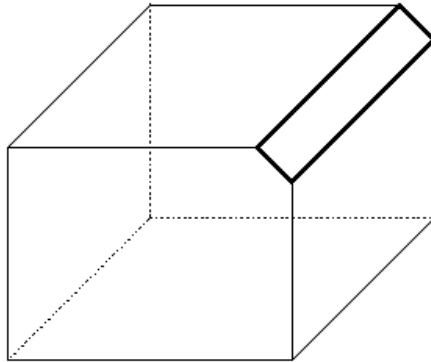
Modeling Functions

(9) Feature based modeling

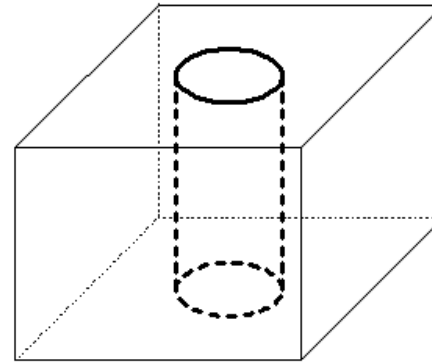


Example of modeling using “slot” and “hole” features

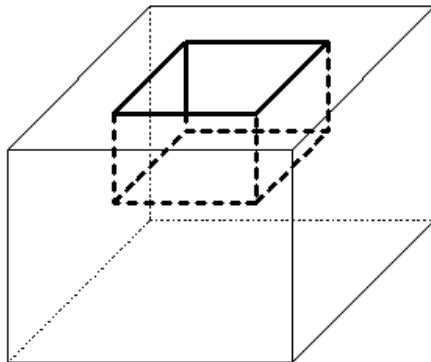
Example modeling using machining features



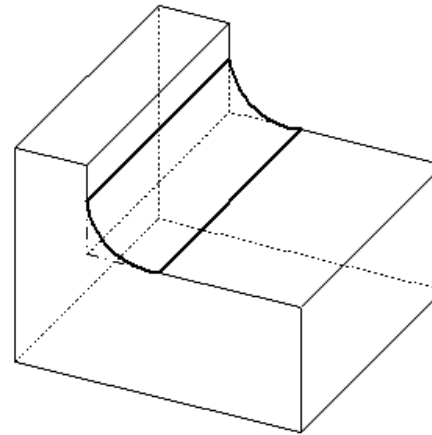
(a) Chamfering



(b) Hole



(c) Pocket



(d) Fillet

Modeling Functions

(9) Feature based modeling

- ▶ Any feature based modeling system cannot provide all the features necessary for all the specific applications
- ▶ The desirable set of features is different between applications
- ▶ Many systems provide feature definition language so that any specific feature can be defined
- ▶ When they are defined, they are parameterized as the primitives

Modeling Functions

(10) Parametric Modeling

- ▶ Model a shape by using the geometric constraints and the dimension data
- ▶ Geometric constraints describe the relation between shape elements
- ▶ Dimensional data include dimensions and relations between the dimensions

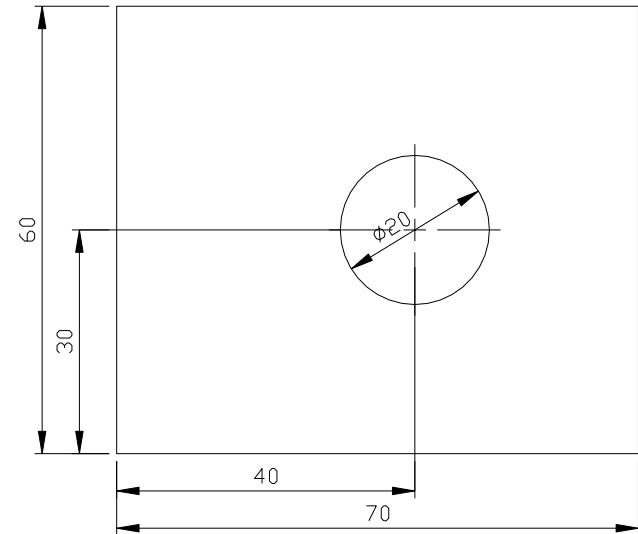
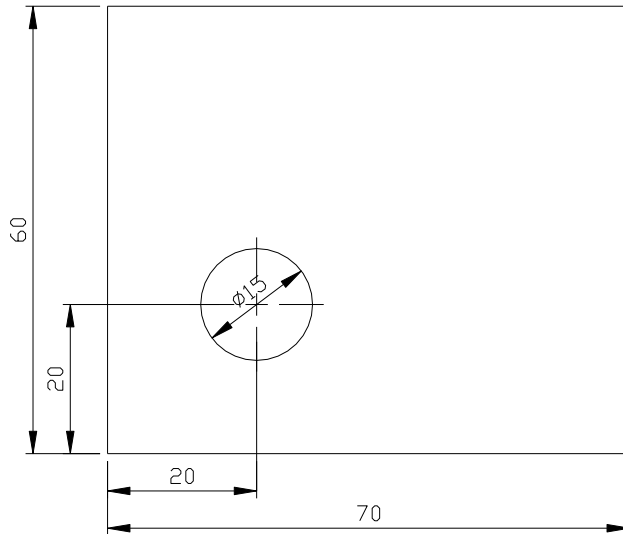
Modeling Functions

(10) Parametric Modeling

- ▶ Input two dimensional shape roughly
- ▶ Input geometric constraints and dimension data
- ▶ Reconstruct the two dimensional shape
- ▶ Create 3D shape by sweeping or swinging

Modeling Functions

(10) Parametric Modeling



Data structure of solid model

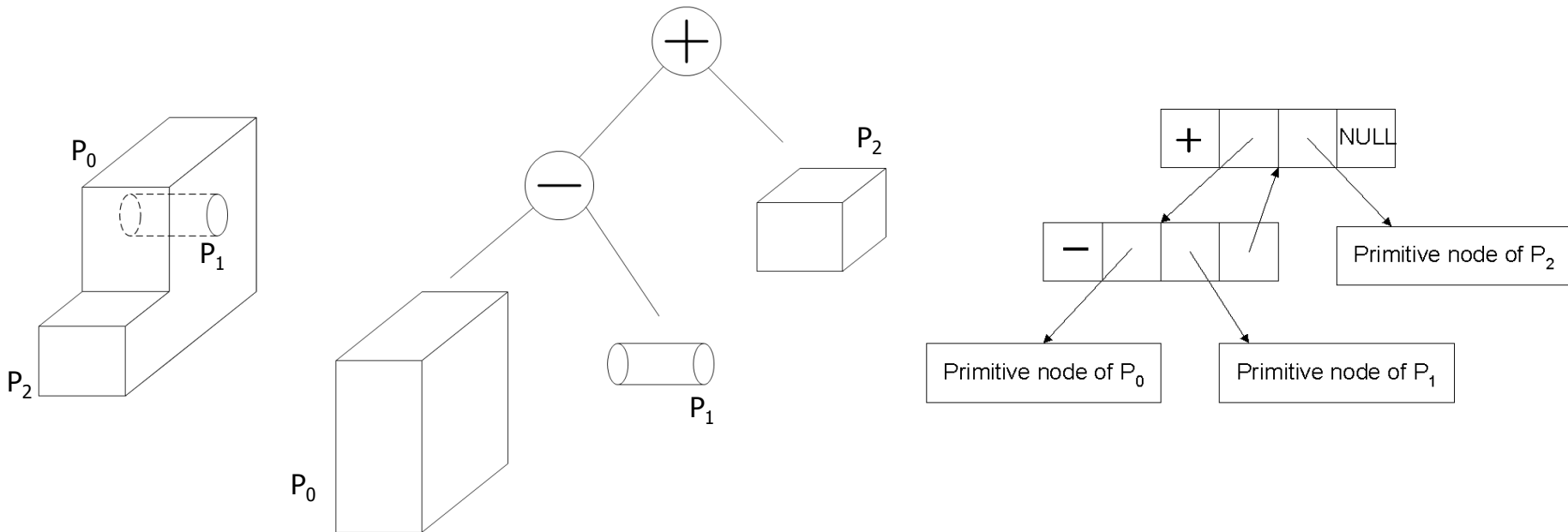
- ▶ **CSG Representation storing CSG tree**
 - ▶ Store procedure Boolean operation in tree structure
- ▶ **Boundary Representation (B-Rep)**
 - ▶ Data structure vertex, edge, face tables
 - ▶ Data structure using half edge
 - ▶ Data structure using Winged-edge

Data structure of solid model – cont'

- ▶ Data structure storing decomposition model
 - ▶ Octree representation
 - ▶ Voxel representation
 - ▶ Cell decomposition model
 - ▶ Similar to finite element

CSG tree

- ▶ Stores the procedure in which Boolean operations are applied



Example of CSG tree

Implementation of CSG tree structure in C language

```
struct operator {
    int      op_type,           /* union, intersection or difference operator */
           L_type;            /* left node type: 0=operator, 1=primitive */
           R_type            /* right node type: 0=operator, 1=primitive */
    void     *L_ptr;          /* left node */
           *R_ptr;          /* right node */
           *p_ptr;          /* parent node */
}
```

```
struct primitive {
    int      prim_type;        /* type of primitive */
    double   pos_x, pos_y, pos_z; /* position of instance */
    double   ori_x, ori_y, ori_z; /* orientation of instance */
    void     *attribute;      /* the value of dimensions of the primitive */
}
```

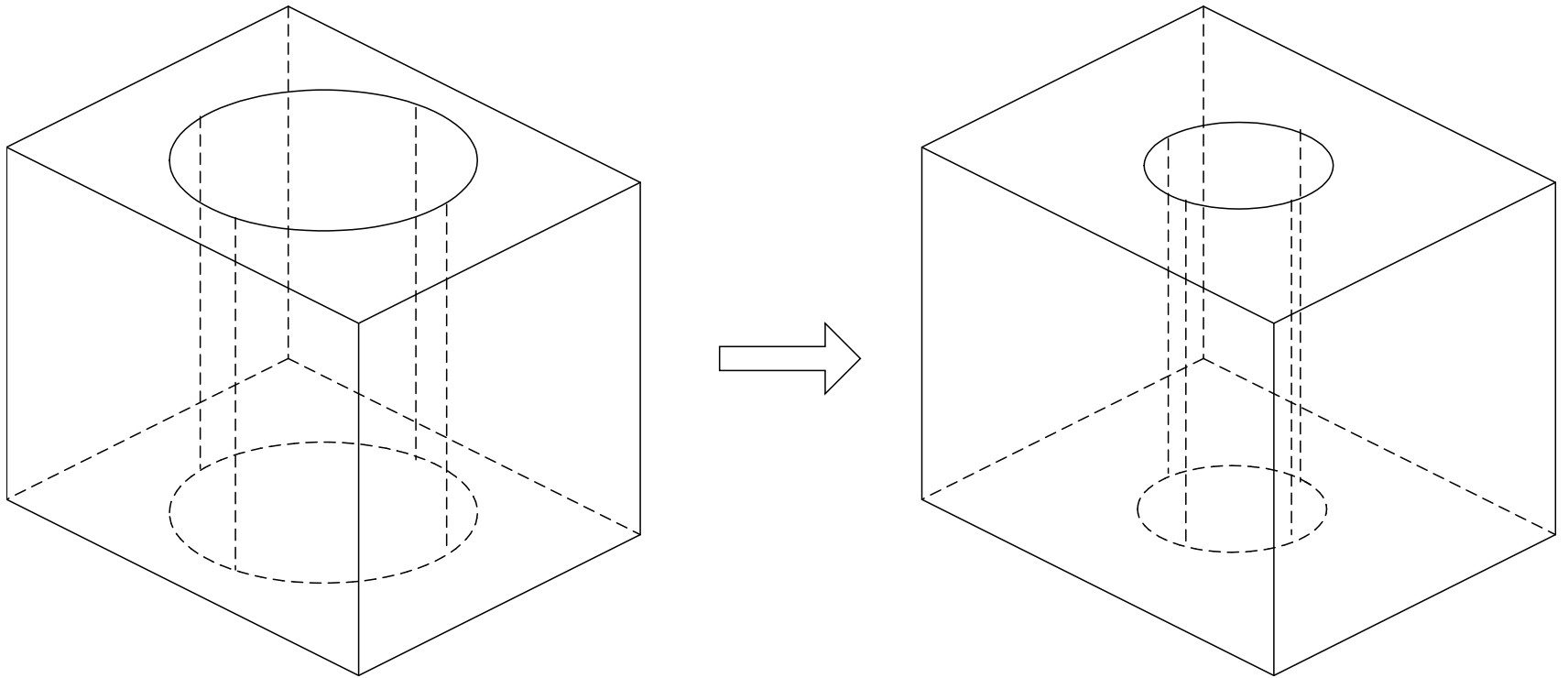
CSG tree representation– advantages

- ▶ Compact data, Easy to maintain
- ▶ Represent only valid object
- ▶ Possible to be converted to B-Rep
 - ▶ Many applications can be integrated
 - ▶ Model can be easily changed by changing parameter values of primitives

CSG tree representation – disadvantages

- ▶ Allows only Boolean operations
- ▶ Shapes to be modeled are limited
- ▶ Impossible to modify locally
- ▶ Significant computation is required for boundary evaluation
 - bad for interactive display
- ▶ Trends are to store B-Rep and Feature tree together

Modification of solid by changing parameters

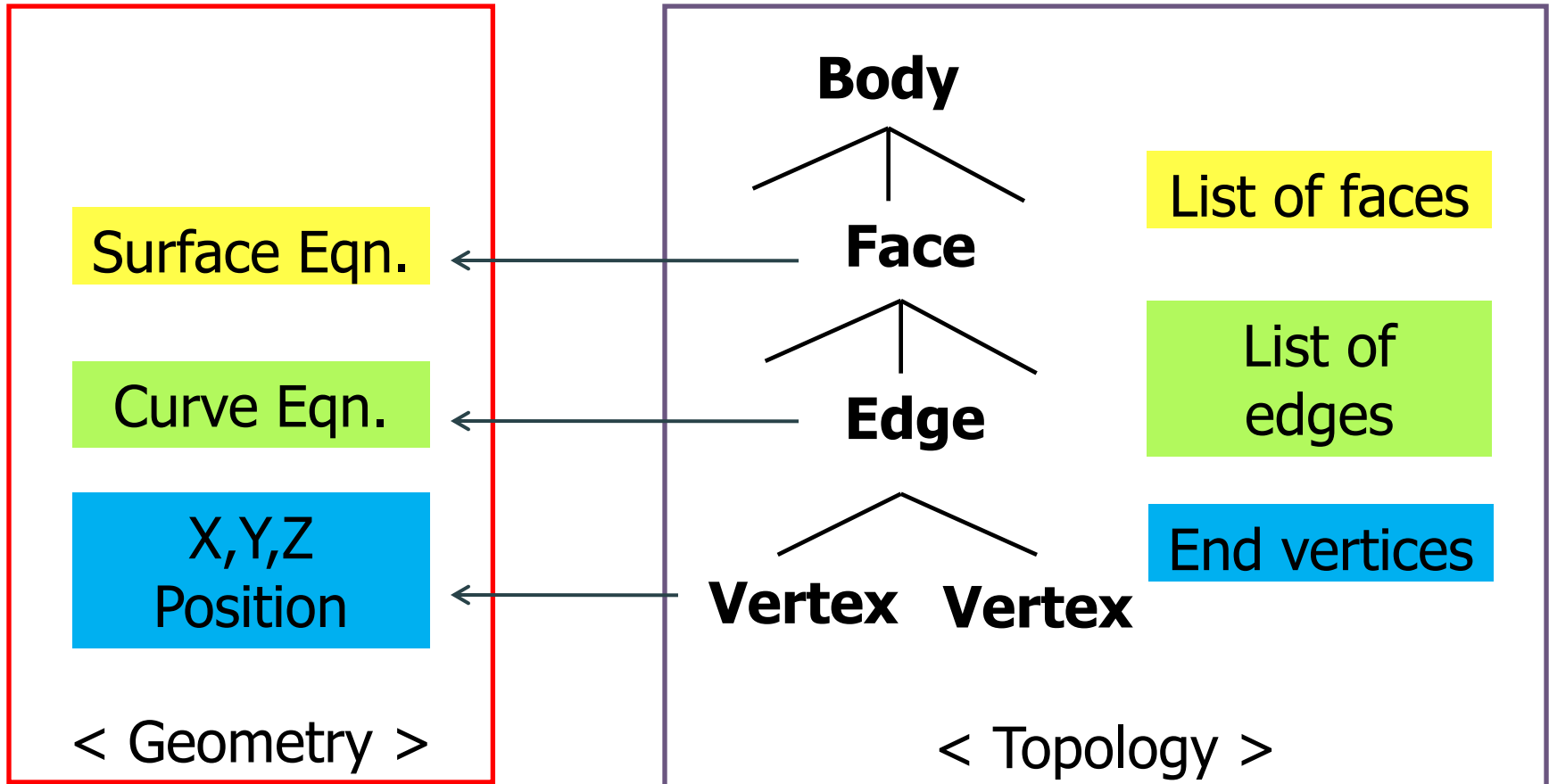


B-Rep(Boundary Representation)

- ▶ Shape is expressed by its bounding entities such as faces, edges, and vertices
- ▶ Bounding entities and their connectivity are stored in graph structure
 - Graph-based model

B-Rep Structure – cont'

Topology vs. Geometry



B-Rep – advantages

- ▶ Boundary data are stored explicitly and enables quick interactive response
- ▶ Topology information can be easily derived
- ▶ Supports various modeling commands (local operations in addition to Boolean)

B-Rep – disadvantages

- ▶ Complicated data structure with a large amount of data
- ▶ Invalid solid may result

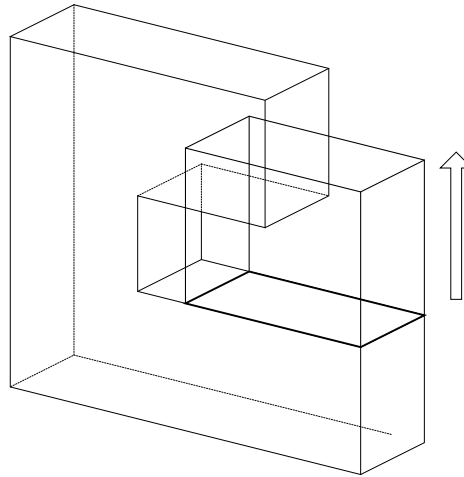
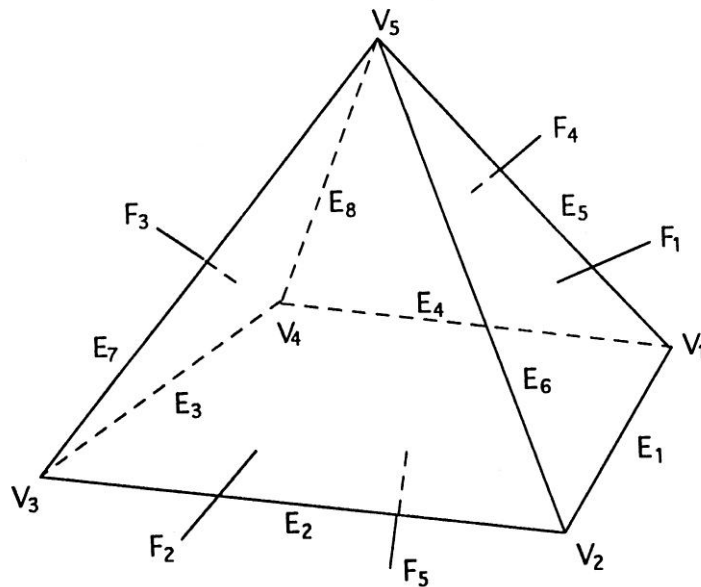


Table-based structure for storing B-Rep

Face table	
Face	Edges
F ₁	E ₁ , E ₅ , E ₆
F ₂	E ₂ , E ₆ , E ₇
F ₃	E ₃ , E ₇ , E ₈
F ₄	E ₄ , E ₈ , E ₅
F ₅	E ₁ , E ₂ , E ₃ , E ₄

Edge table	
Edge	Vertices
E ₁	V ₁ , V ₂
E ₂	V ₂ , V ₃
E ₃	V ₃ , V ₄
E ₄	V ₄ , V ₁
E ₅	V ₁ , V ₅
E ₆	V ₂ , V ₅
E ₇	V ₃ , V ₅
E ₈	V ₄ , V ₅

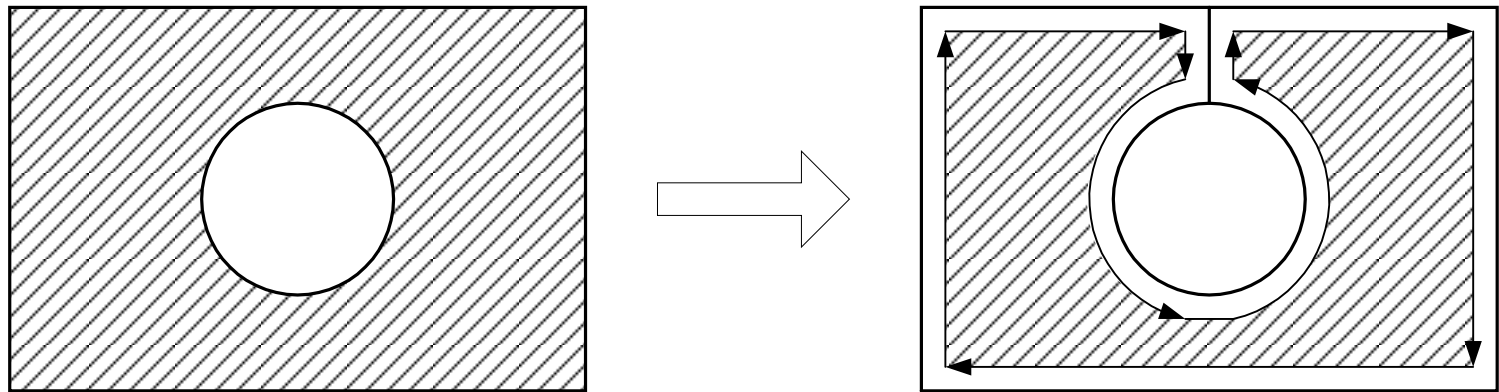
Vertex table	
Vertex	Coordinates
V ₁	x ₁ , y ₁ , z ₁
V ₂	x ₂ , y ₂ , z ₂
V ₃	x ₃ , y ₃ , z ₃
V ₄	x ₄ , y ₄ , z ₄
V ₅	x ₅ , y ₅ , z ₅
V ₆	x ₆ , y ₆ , z ₆



Things to be considered

- ▶ Balance between structure compactness and effectiveness in data retrieval
- ▶ Basically used for polyhedron models
- ▶ For objects with curved surfaces and curved edges, information on surface equations are stored in the Face table, information on curve equations are stored in the Edge table
- ▶ If there are faces with holes, the current Face table cannot be used

Treatment of face with multiple boundaries



Adding bridge-edge is one way to handle hole

B-Rep – Things to be considered

- ▶ Length of edge table in the Face table varies
→ Loss of memory usage
- ▶ Deriving adjacency among Vertex, Edge, Face requires a heavy search

Ex) Which faces share a given edge?

Which edges share a given vertex?

Decomposition Model Data Structure

- ▶ **Decomposition model:**
 - ▶ Represent an object as an aggregation of simple objects such as cubes

Voxel model (Exhaustive enumeration)

- ▶ Space of interest is represented by a set of cubes (voxels) after being subdivided by grid planes
- ▶ Only the voxels embodied by the object are stored
- ▶ Use 3D array $C(i, j, k)$, $C(i, j, k)$ corresponding to the embodied voxels is set to 1. Others set to 0
- ▶ Popular in digital image processing

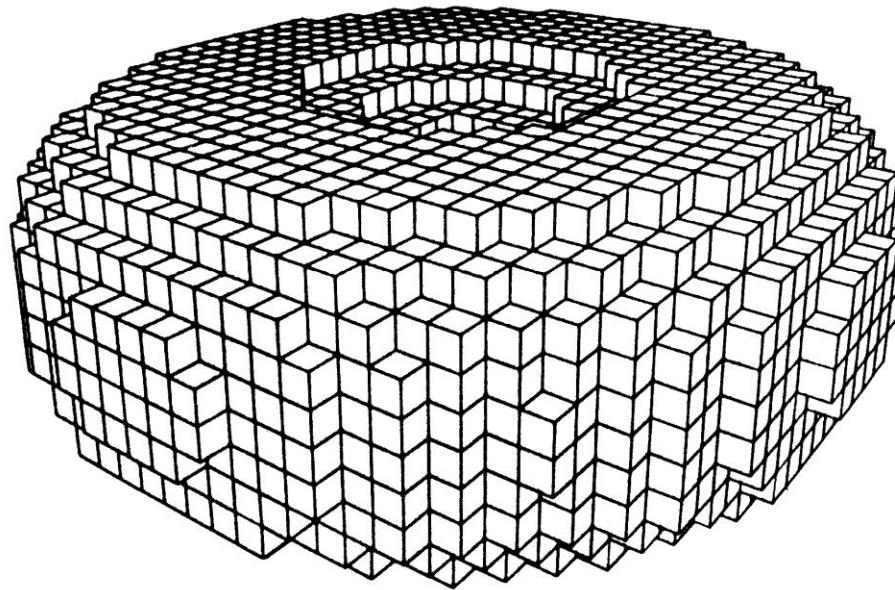
Voxel model – cont'

- ▶ Any shape can be represented, approximately at elast
- ▶ Used to model human bones and organs from digital topography
- ▶ Easy to implement mass property calculation and Boolean operation
- ▶ Information on empty space is also available

Voxel model – cont'

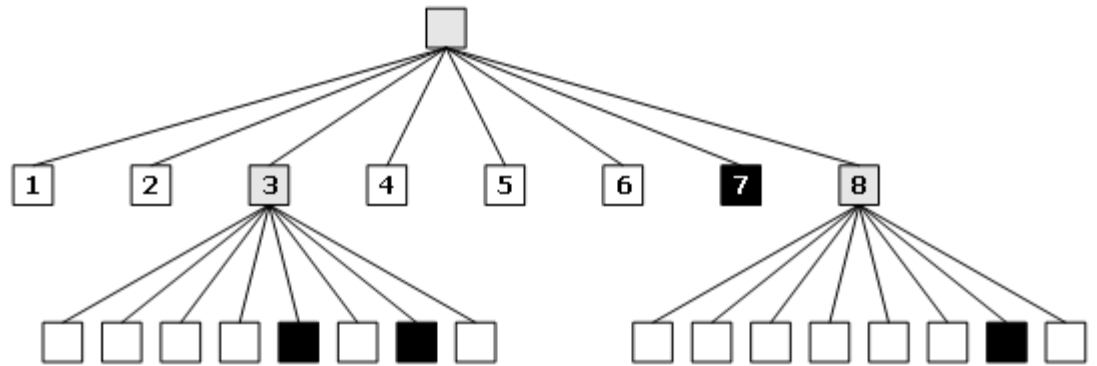
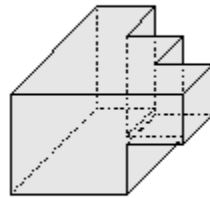
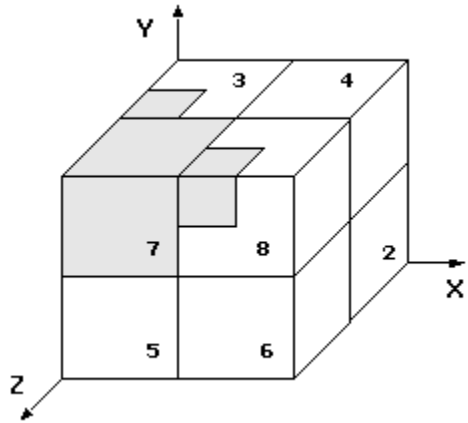
- ▶ Memory requirement varies drastically depending upon desired resolution
- ▶ Used as a secondary representation for computation convenience

Visualization of voxel representation



Octree representation

- ▶ Only voxels occupying the object space are subdivided, Extension of Quadtree to 3D



Data structure for storing octrees

```
struct    octreeroot
{
    float   xmin, ymin, zmin;      /* space of interest */
    float   xmax, ymax, zmax;
    struct  octree   *root;        /* root of the tree */
};

struct    octree
{
    char           code;           /* BLACK, WHITE, GREY */
    struct  octree *oct[8];        /* pointers to octants, present if
    GREY */
};
```

Procedure of octree generation

```
make_tree( p, t, depth )
primitive    *p;          /* p = the primitive to be modeled */
octree       *t;          /* t = node of the octree, initially
    the initial tree with one grey node */
int depth;    /* initially max. depth of the recursion */
{
    int i;
    switch( classify( p, t ) )
    {
        case WHITE:
            t->code = WHITE;
            break;

        case BLACK:
            t->code = BLACK;
            break;
```

Procedure of octree generation – cont'

```
case GREY:
    if( depth == 0 )
    {
        t->code = BLACK;
    }
    else
    {
        subdivide( t );
        for( i = 0; i < 8; i++ )
            make_tree( p, t->oct[i],
depth-1 );
    }
    break;
}
}
/* classify octree node against primitives */
classify( ... );
/* divide octree node into eight octants */
subdivide( ... );
```

Cell decomposition model

