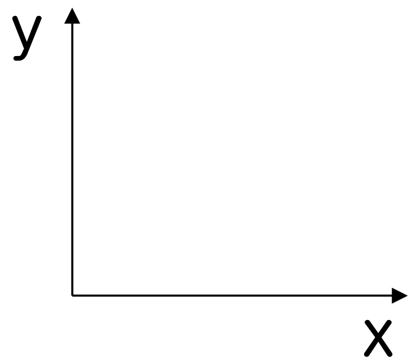
Geometric Modeling

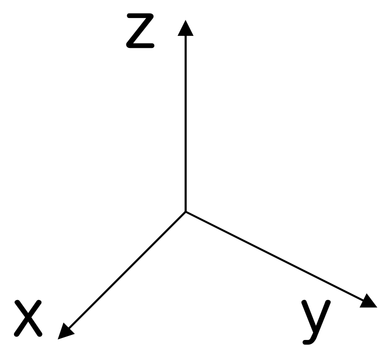
- 기하학적 형상을 만드는 행위
- Modeling
- Model
  – 물체의 특성을 완전하고 정확하게 표현해야 함
  – Physical Model vs. Mathematical Model

Modeling Space

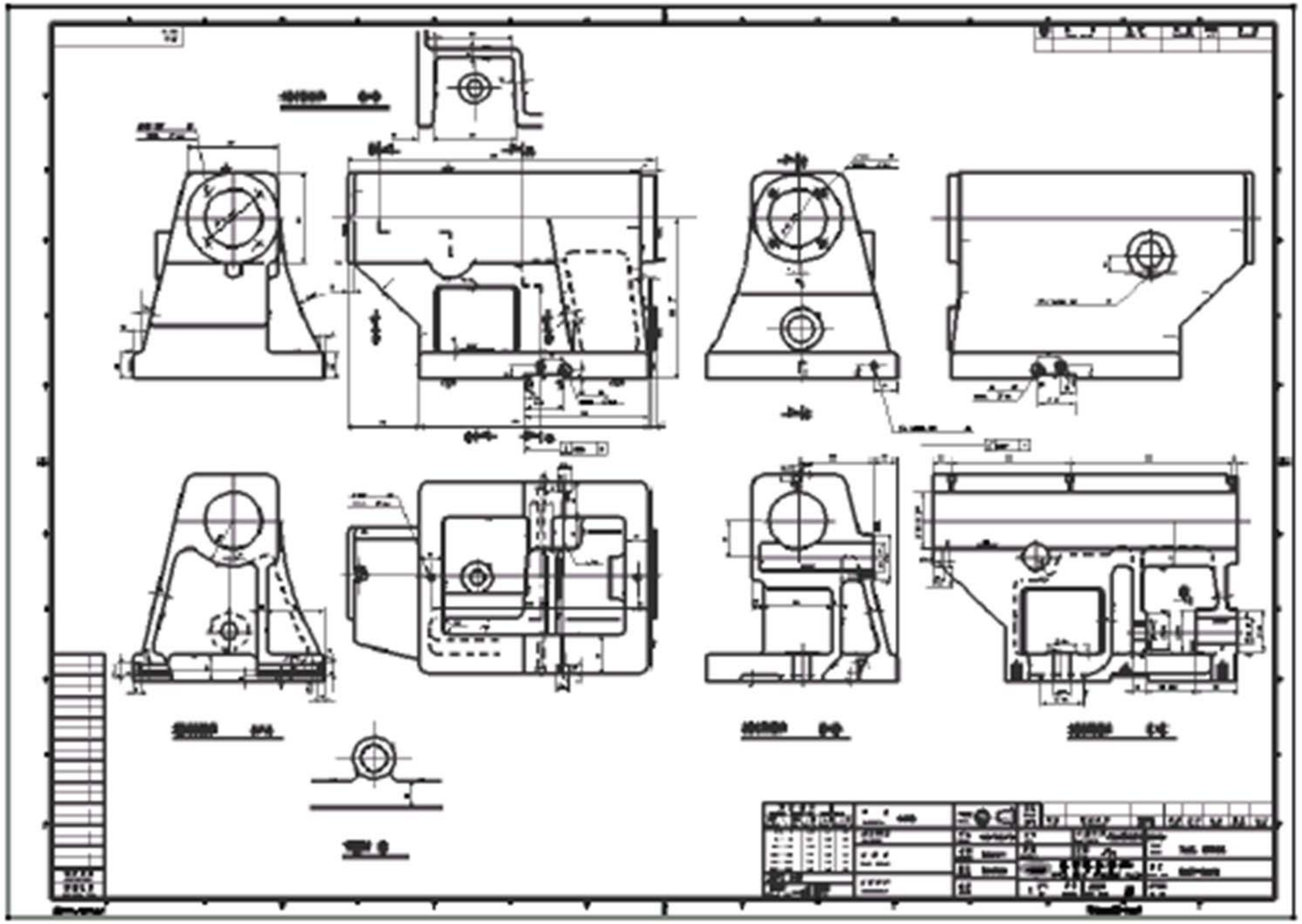
- 2D space (x,y)
  - Computer aided drafting



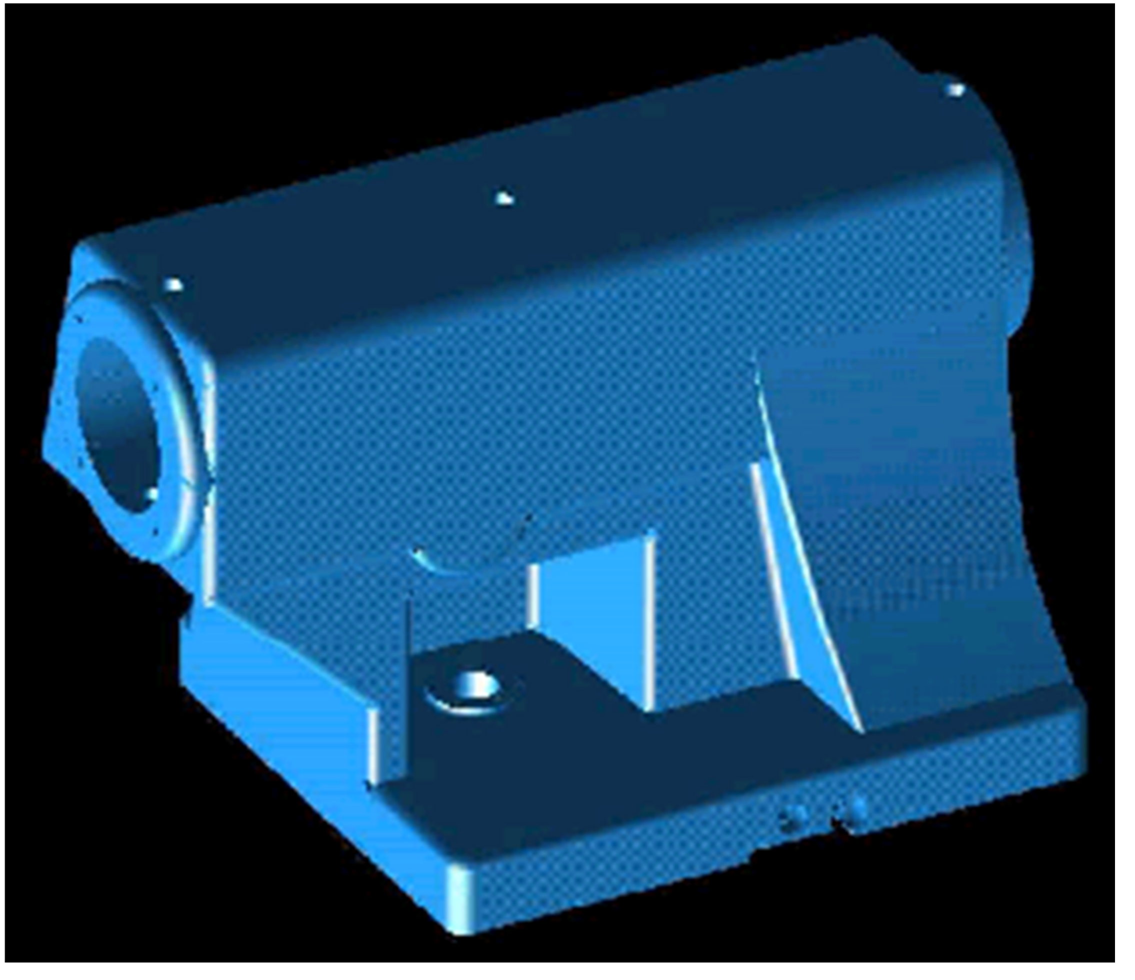
- 3D space (x,y,z)



Why 3D Model? (1)



Why 3D Model? (2)



Geometric Entity

- Point : 좌표값
- Curve : 곡선의 방정식 (planar/space/ freeform)
- Surface : 곡면의 방정식 (plane/sculptured)
- Solid : 부피를 둘러싸고 있는 곡면들의 방정식

Geometric entity를 편리하게 정의하는 기능
해당좌표나 방정식을 유도하고 저장하는 기능

수학적 이론

CAD system

점(point) 정의

- 직접입력
  - 좌표값 입력
- 간접입력
  - 두 곡선의 교점 이용
  - 점을 곡면 위에 투영
  - 곡선과 곡면의 교점 이용
  - 주어진 곡선 위에 일정한 간격으로 n개의 점 생성

2D 곡선(curve) 정의

- 기본 2D 곡선 이용
  - 직선(line), 원(circle), 타원(ellipse), 포물선(parabola), 쌍곡선(hyperbola)
  - 몇 개의 파라미터값으로 간단하고 정확하게 표현
- 두 곡선 사이를 Blending
  - Rounding/Chamfer
- 순서가 정해진 여러 개의 2D 점 이용
  - 자유곡선을 정의
  - Interpolation/Approximation

3D 곡선(curve) 정의

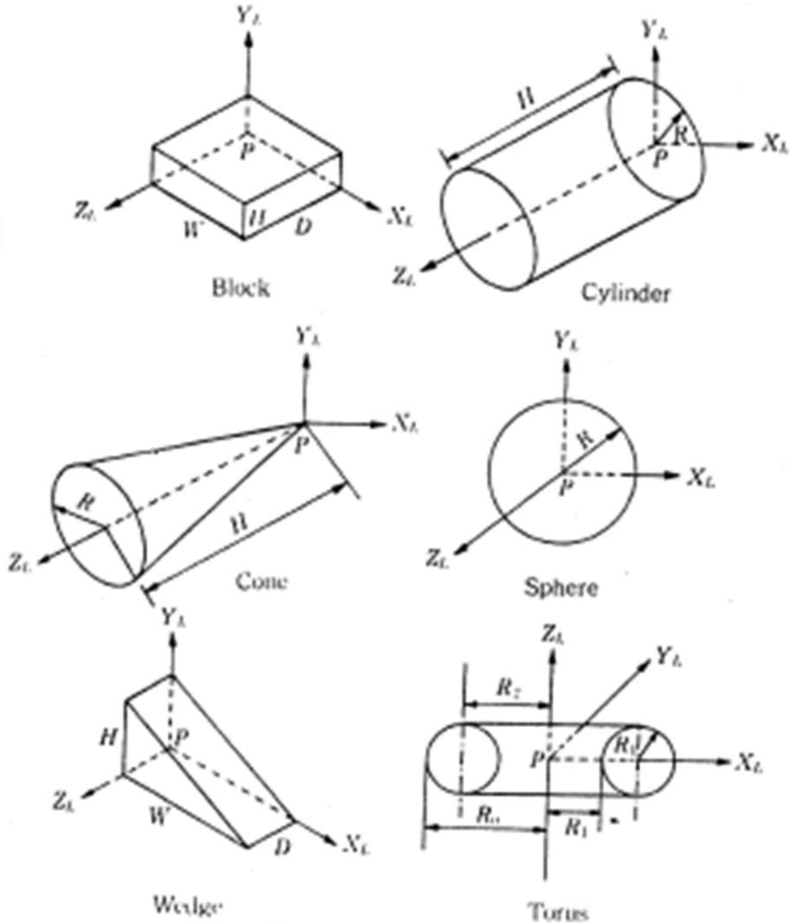
- 순서가 정해진 여러 개의 3D 점 이용
  - Interpolation/Approximation
- 두 곡면의 교선 이용
- 3D 곡선을 곡면 위에 투영시킨 곡선 이용
- 여러 view에 그려진 2D 곡선들로 부터 3D 곡선을 역으로 계산

곡면(surface) 정의

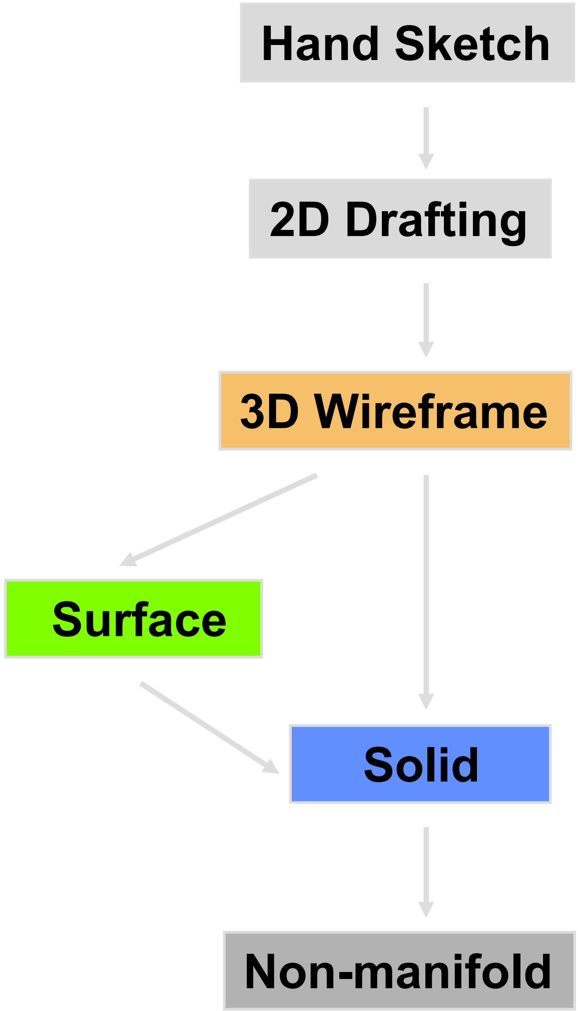
- 기본곡면 이용
  - 원통(cylinder), 원추(cone), 구(sphere), 타원체(ellipsoid), 원환체(torus)
- 두 곡면사이를 Blending
- 윤곽곡선이나 단면곡선 Sweeping
- 그물과 같은 곡선망을 이용
  - Curve-net fitting
- 3D 점들의 집합을 이용
  - Surface fitting

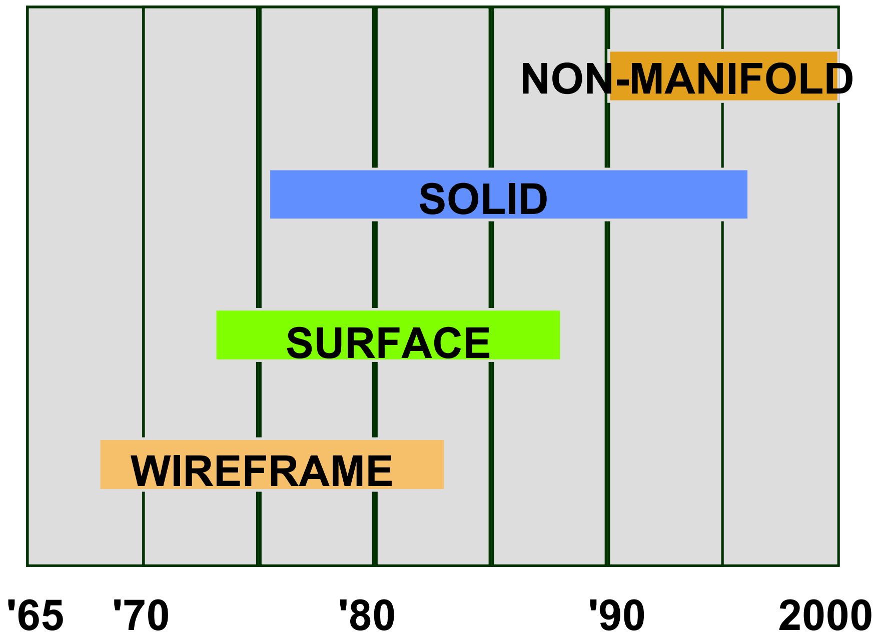
솔리드(solid) 정의

- 곡면 + 부피(닫힌 형태)
- Primitives
- Complex Solids
  - Boolean operations of primitives



Evolution of Geometric Modeling Systems

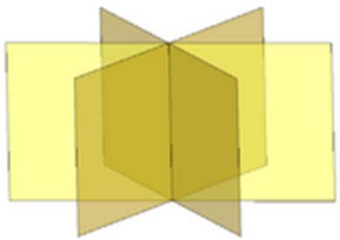




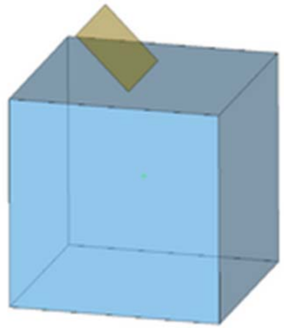
Non-manifold

- In manifold model,
  - Every point on a surface is two-dimensional
  - Every point has a neighborhood that is homeomorphic to 2D disk

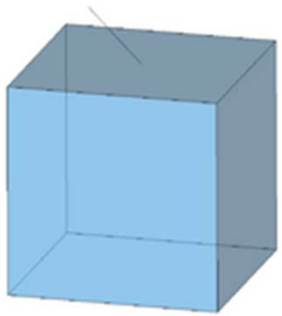
Vertex in a wire body



Edge in a sheet body



Edge

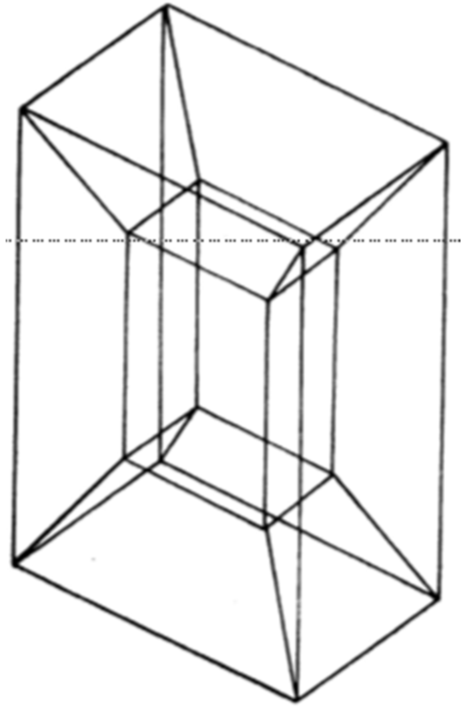


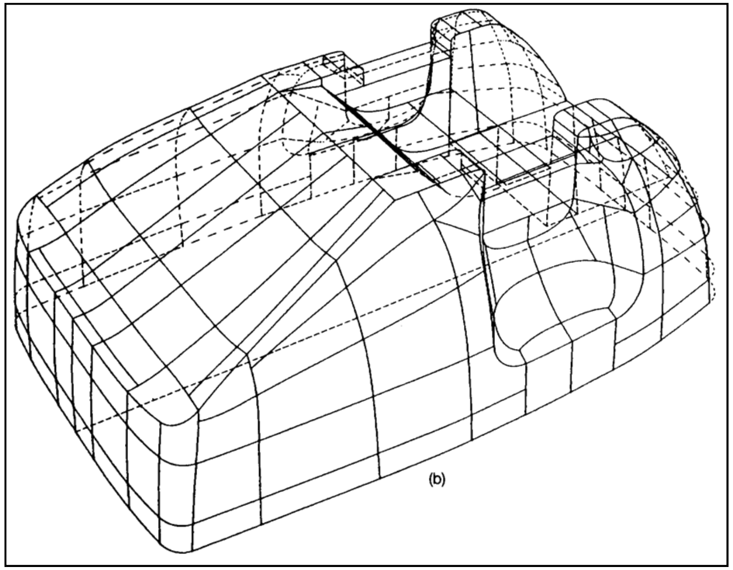
Vertex

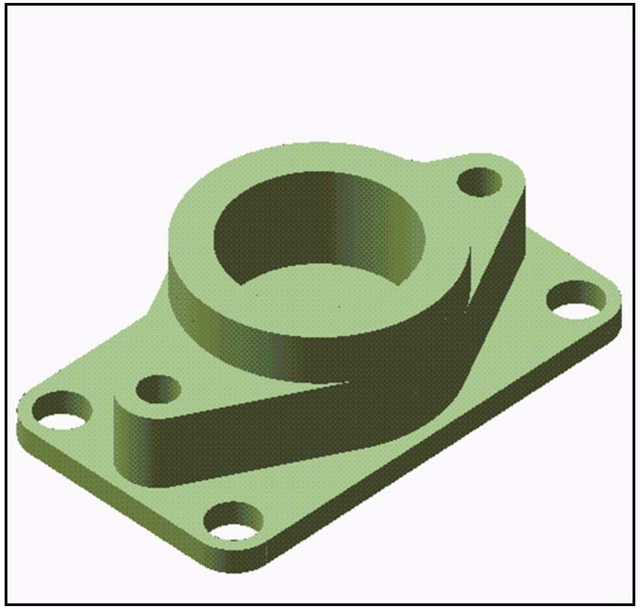
Vertex

Geometric Model

- Wireframe/ Surface/ Solid Model





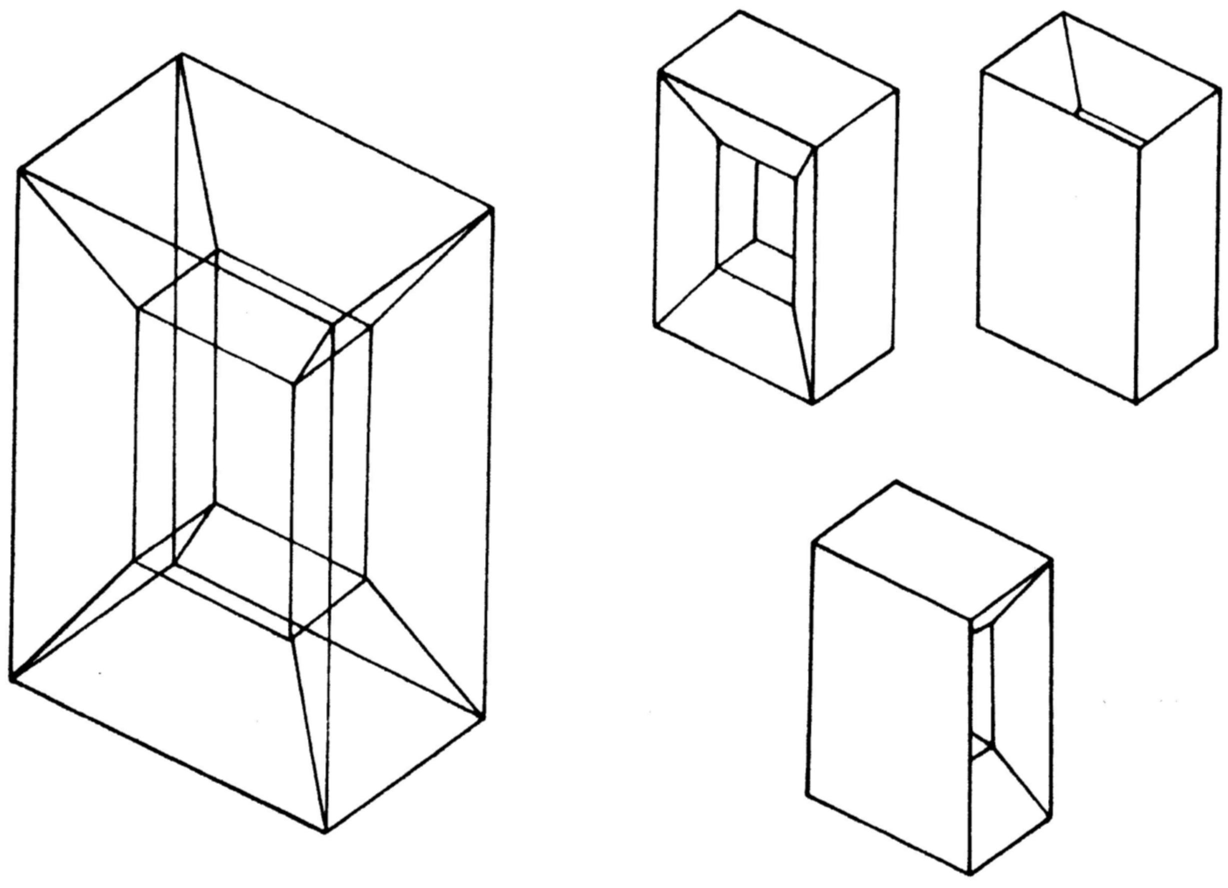


Contents

Wireframe Model

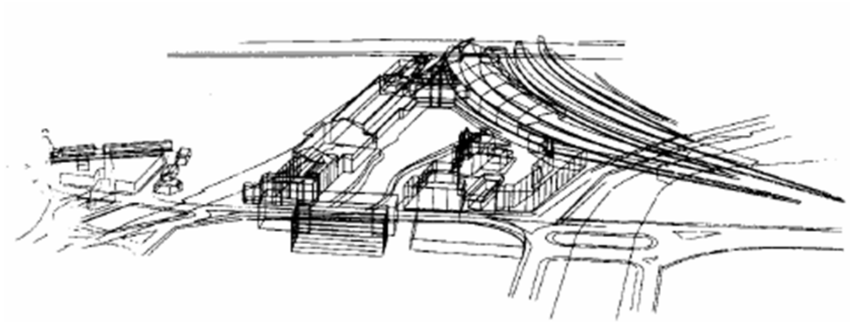
- Database
  - Represent a shape by its characteristic lines and end points
- Advantages
  - Require simple user input to create a shape
  - Easy to develop systems
- Disadvantages
  - Models can be ambiguous
  - No boundary surfaces and volume information
  - Impossible to calculate mass properties, drive NC tool paths, generate FEM meshes
- Products
  - Sketchpad, Steerbear

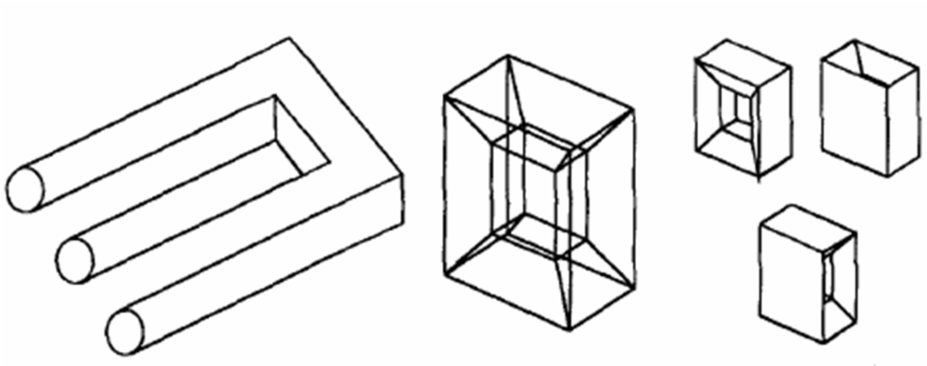
Ambiguous Wireframe Model

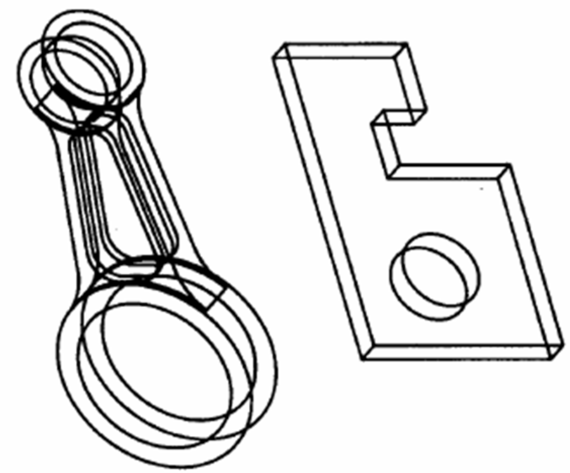


Basic 3D Models: Wireframe (1)

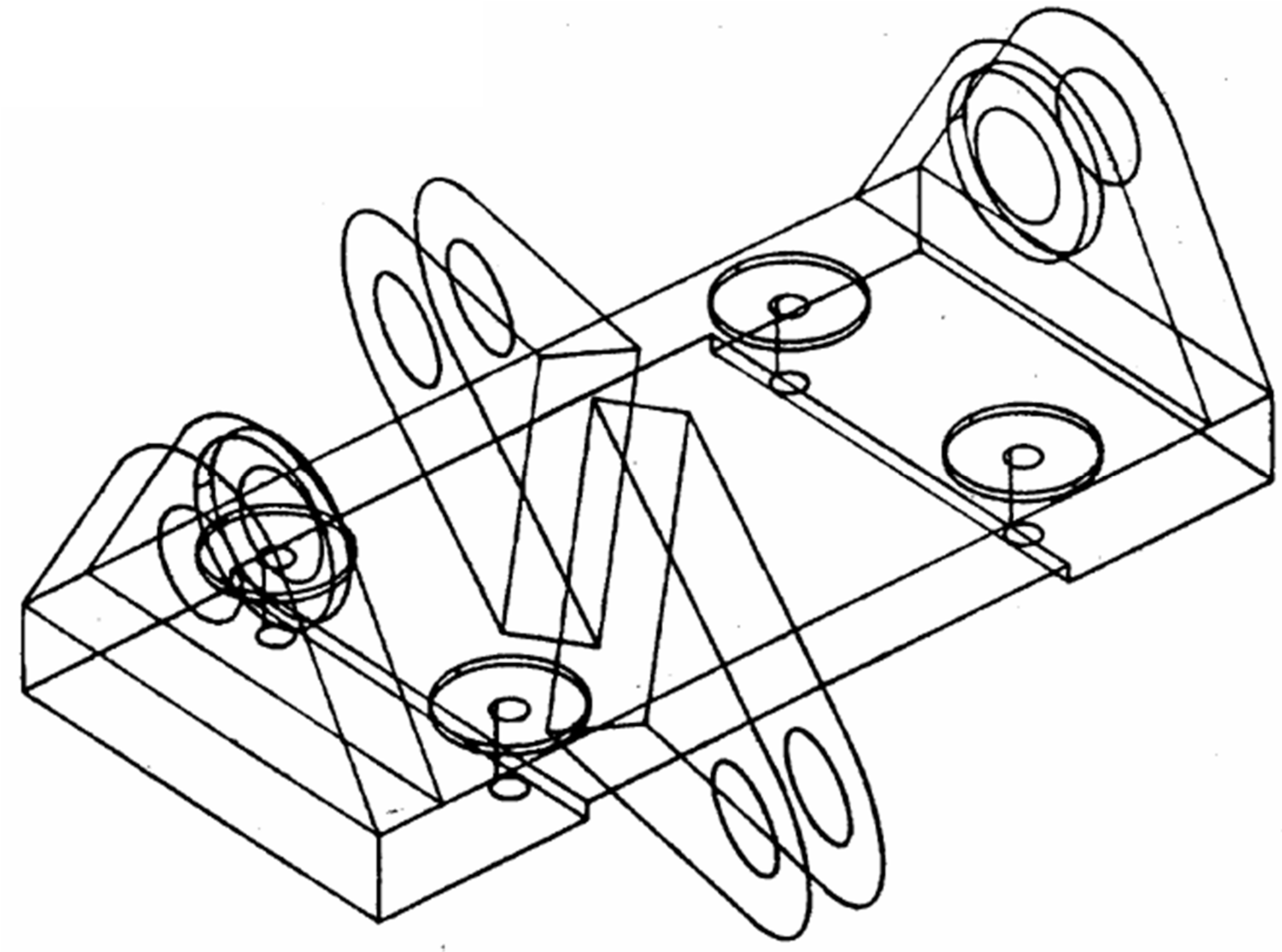
- Wireframes
  - easiest of all to create
  - nothing hidden
  - visually ambiguous
  - topological problems?





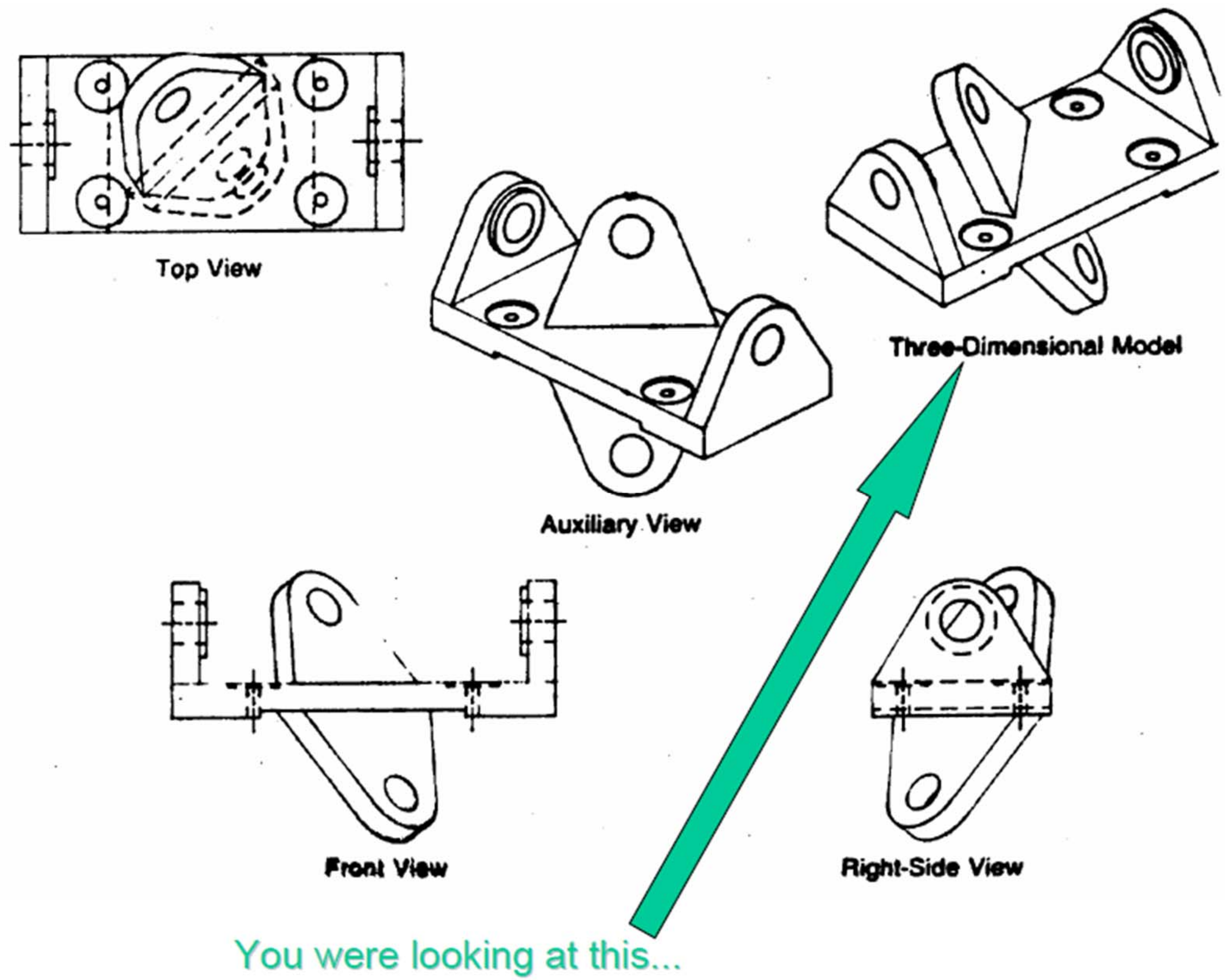


Basic 3D Models: Wireframe (2)



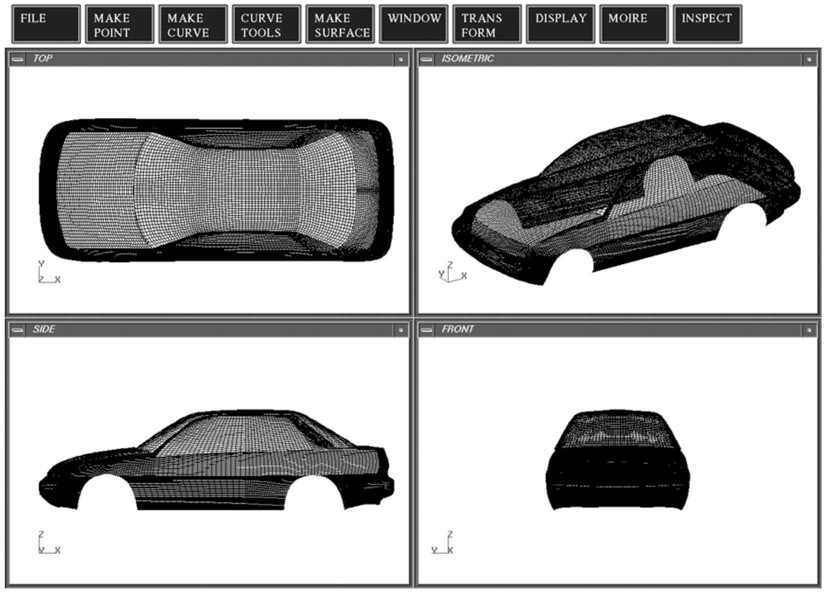
Can you figure out what this is? (It’s really a valid wireframe model…)

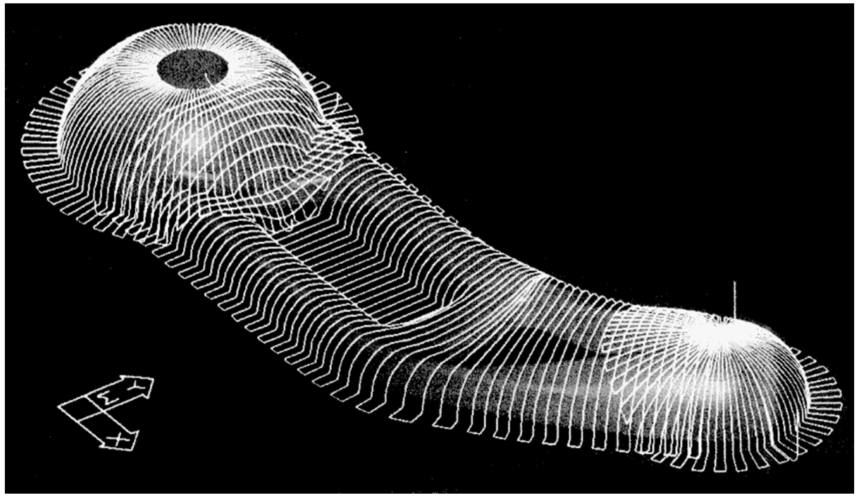
Basic 3D Models: Wireframe (3)



Surface Model (1)

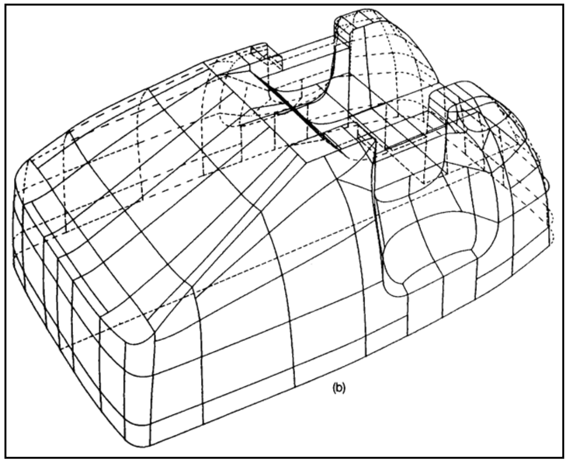
- Purpose
  - Visual model for aesthetical evaluation
  - Mathematical description to generate the NC Tool Paths

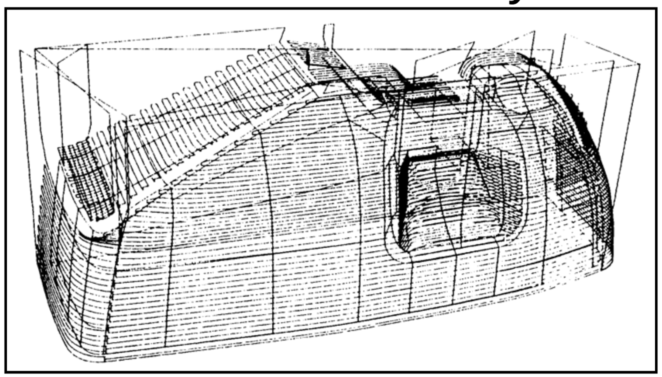




Surface Model (2)

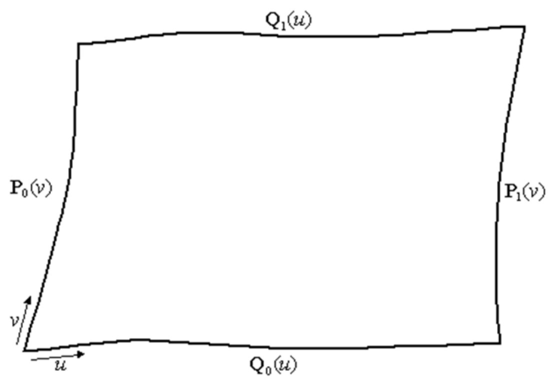
- Database
  - Characteristic lines and end points + surface (+surface connectivity) information
  - surface connectivity information
    - Useful for checking gouging of a surface adjacent to the surface being machined
    - If the system includes only a list of surface equations of infinite surfaces without connectivity information, the application should derive the surface boundaries and their connectivity information
  - NC Tool Path Generation with Surface Connectivity Information

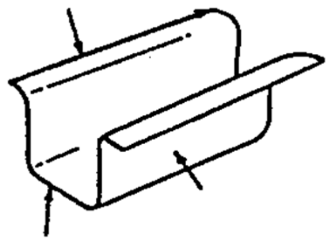


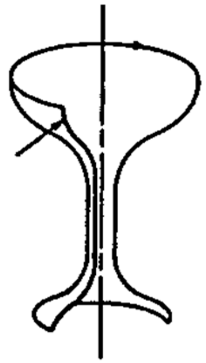


Surface Model (3)

- Creating methods
  - Interpolating the input points
  - Interpolating the curve nets
  - Translating or Revolving a specified curve





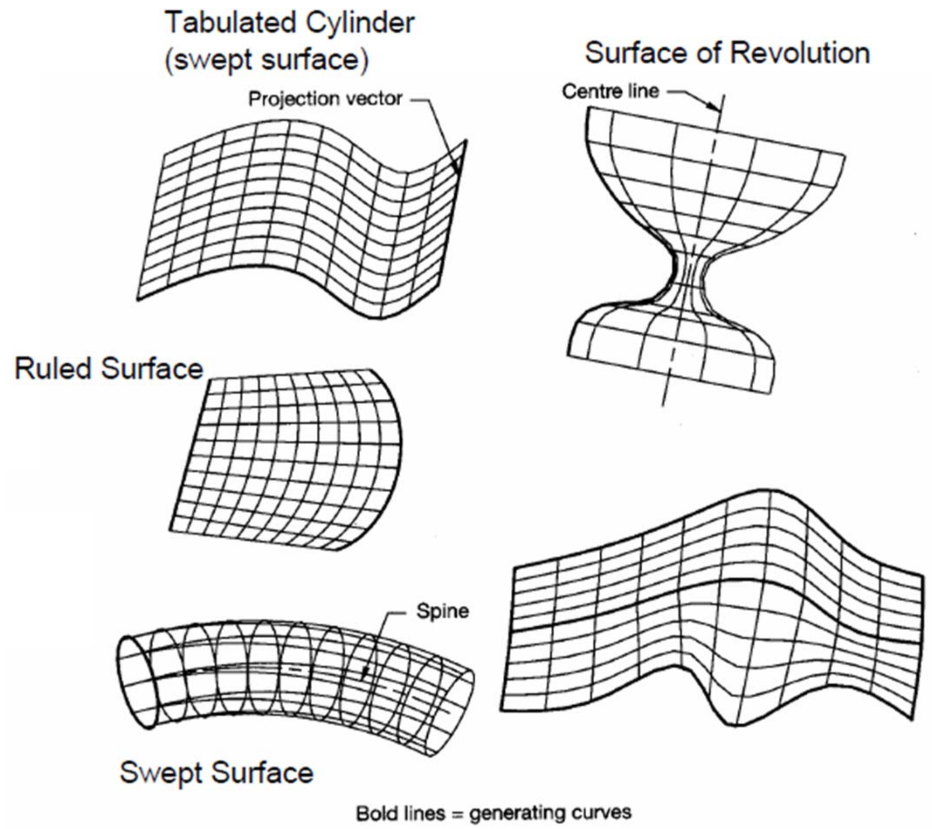


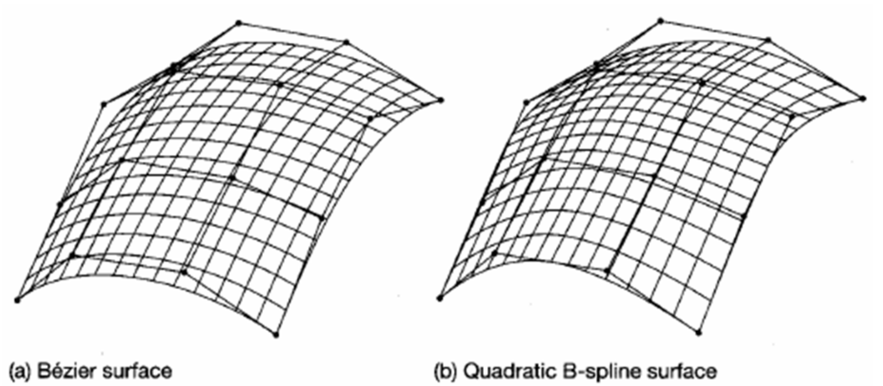
Surface Model (4)

- Advantages
  - Automatic NC tool path generation
  - Visual model colored and shaded
- Disadvantages
  - Cannot calculate mass properties
  - Cannot generate FEM meshes
- Products
  - CATIA, ALIAS, OMEGA, SPEED+
  - Applications for CAM and CG

Basic 3D Models: Surface (1)

- Surface models
  - accurate surface definition
  - enclose a volume
  - topologically difficult to handle…

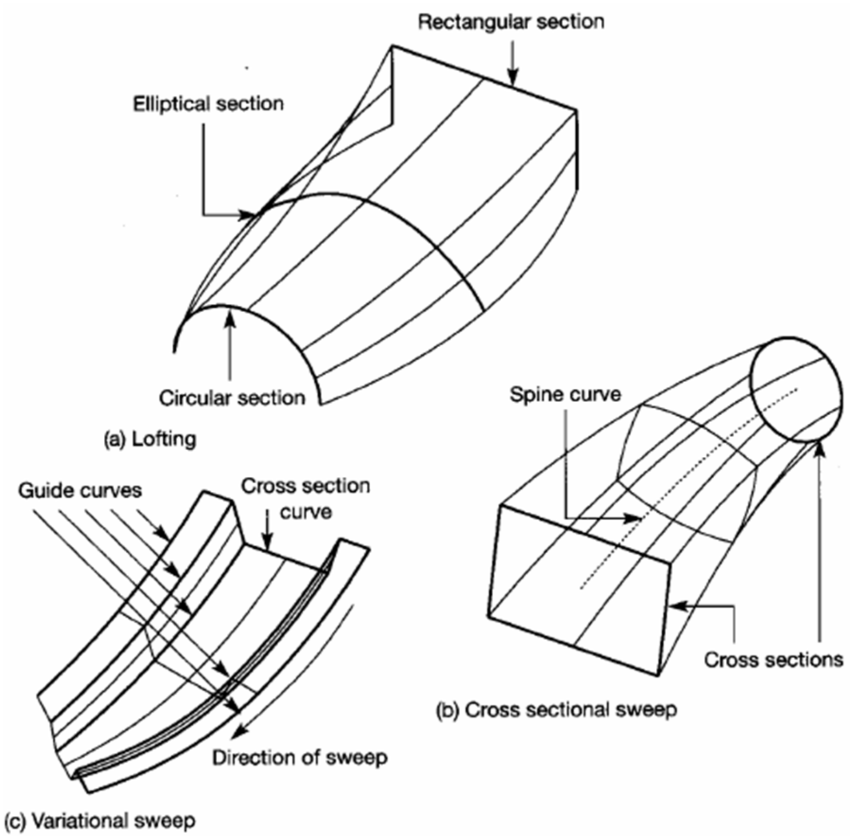


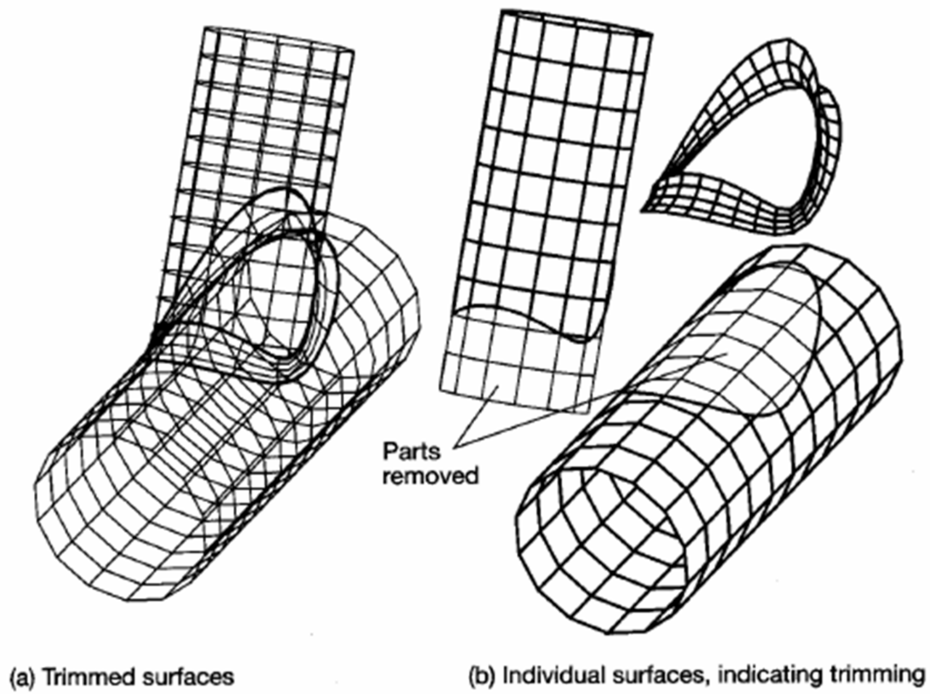


NOTE: control points define shape of surface

Basic 3D Models: Surface (2)

- More complex surfaces





Solid Model (1)

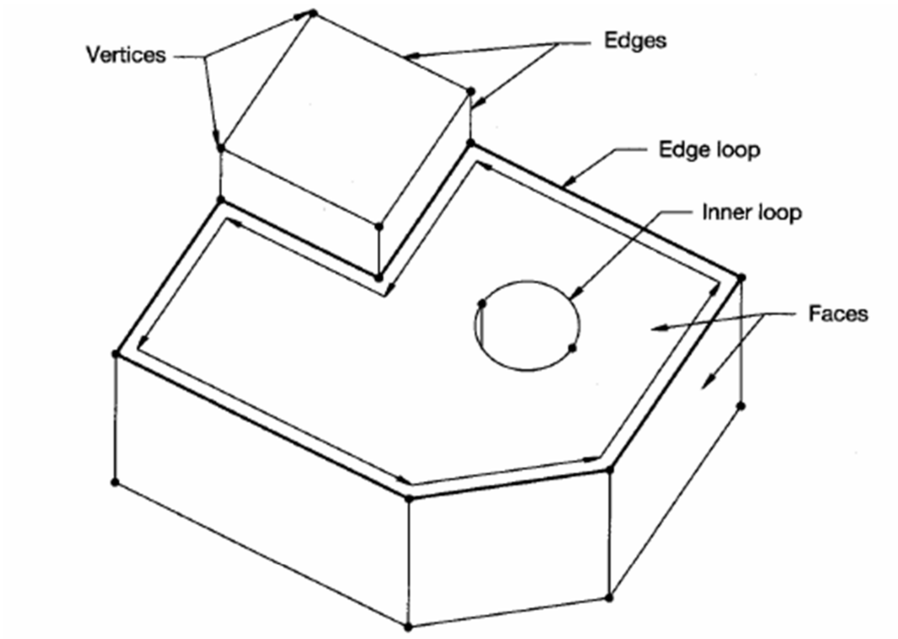
- Database
  - Store a closed volume
  - Surface information + In/out information
  - Not allow a simple set of surfaces or characteristic lines if it can not form a closed volume
- Products
  - TIPS, PADL-2, BUILD2, ROMULUS, DESIGNBASE, Pro/Engineer, SolidWorks, SolidEdge, CATIA, ParaSolid

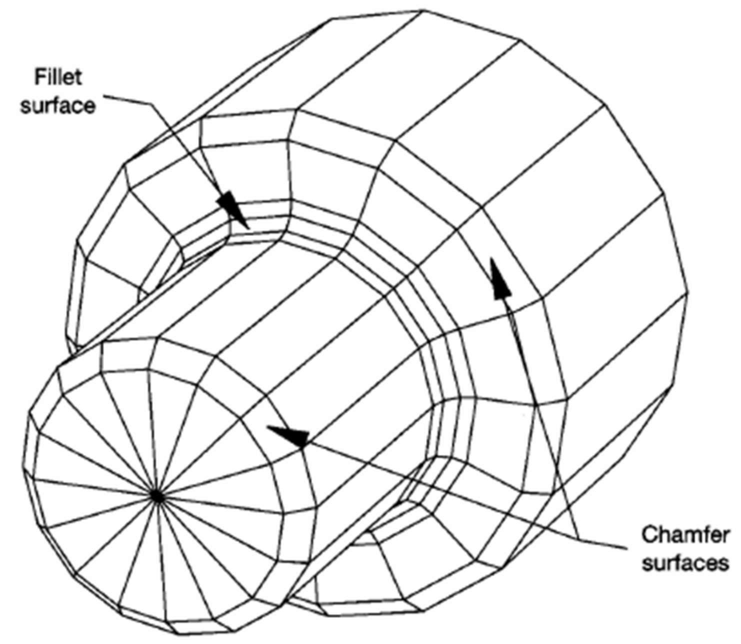
Solid Model (2)

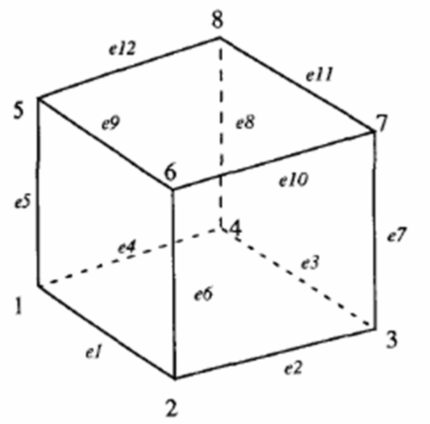
- Advantages
  - Calculate mass properties
  - Generate FEM solid meshes
  - Interference checking between objects
  - 3D visual model colored and shaded
  - NC tool path generation and simulation
- Disadvantages
  - Permit only a complete solid model
  - Require a large amount of input data (complicated and difficult)
  - Large amount of data storage

Basic 3D Models: Solid (1)

- Volumes
  - combine surfaces together
  - topology is a problem
  - boundary representation models (B-rep)





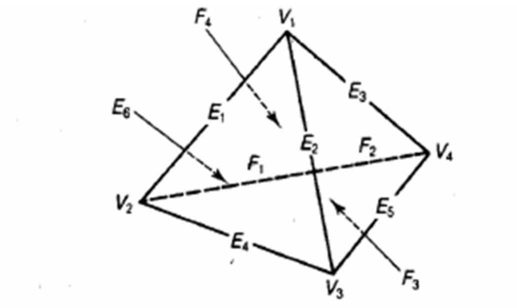


v1	x1	y1	z1		f1	v1	v2	v3	v4
v2	x2	y2	z2		f2	v6	v2	v1	v5
v3	x3	y3	z3		f3	v7	v3	v2	v6
v4	x4	y4	z4		f4	v8	v4	v3	v7
v5	x5	y5	z5		f5	v5	v1	v4	v8
v6	x6	y6	z6		f6	v8	v7	v6	v5
v7	x7	y7	z7						
v8	x8	y8	z8						

A vertex-based boundary model

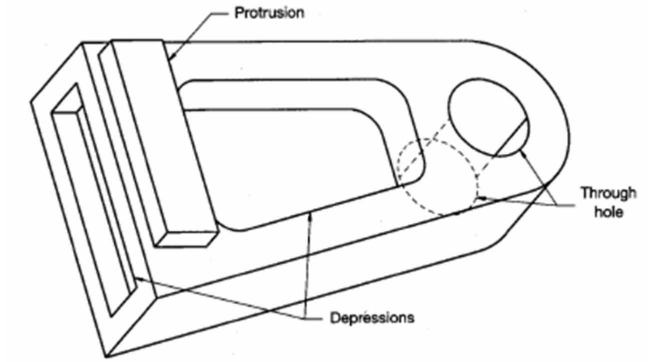
Basic 3D Models: Solid (2)

- Boundary models (b-rep)
  - aka: graph-based models
  - graph nodes & edges



Vertices	Edges	Faces
V_1	E_1	F_1
V_2, V_3, V_4 E_1, E_2, E_3 F_1, F_2, F_4	V_1, V_2 E_2, E_3, E_4, E_6 F_1, F_4	V_1, V_2, V_3 E_1, E_4, E_2 F_2, F_3, F_4
V_2	E_2	F_2
V_1, V_3, V_4 E_1, E_4, E_6 F_1, F_3, F_4	V_1, V_3 E_1, E_3, E_4, E_5 F_1, F_2	V_1, V_3, V_4 E_2, E_5, E_3 F_1, F_3, F_4
V_3	E_3	F_3
V_1, V_2, V_4 $E_2, \ldots$	V_1, V_4 $E_1, E_2, \ldots$	V_4, V_3, V_2 $E_4, \ldots$

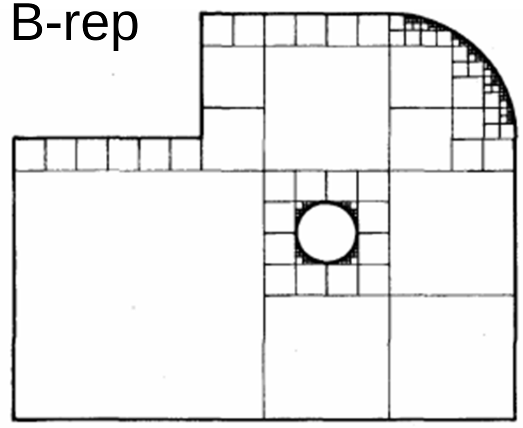
- Rules
  - faces bounded by single loop or ring of edges
  - edge joins exactly 2 faces and is terminated by vertices
  - at least 3 edges meet at each vertex
  - Euler's Rule applies: V-E+F=2 (extended: V-E+F-H=2(S-P))



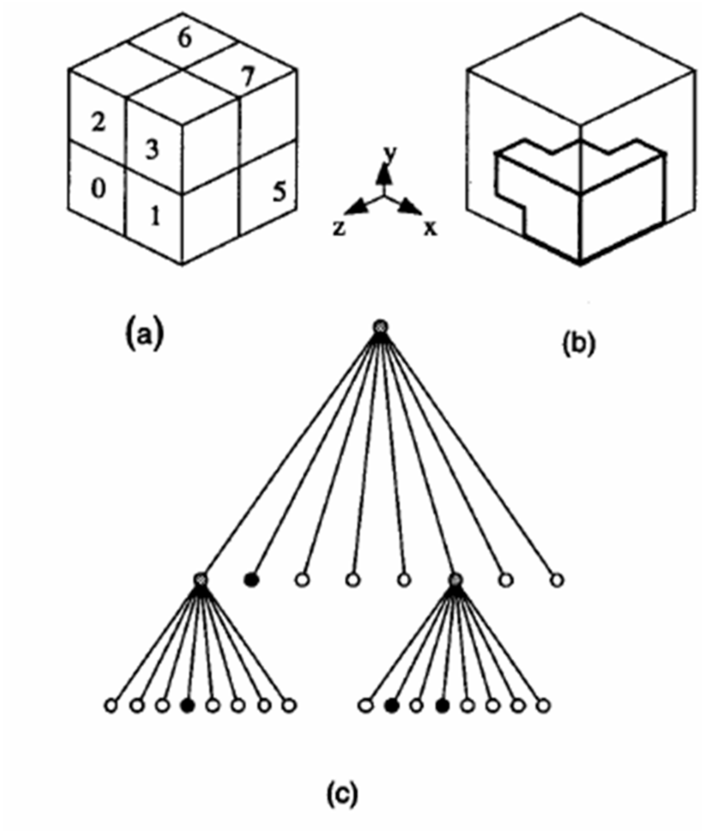
	Vertices	Edges	Faces	Holes	Passages
Basic shape	8	12	6		
Protrusion adds	8	12	5	1	
Depression with sharp corners adds	8	12	5	1	
Depression with filleted corners adds	16	24	9	1	
Through hole adds	4	6	2	2	1

Basic 3D Models: Solid (3)

- Solid models
  - Notion of inside vs. outside
  - Analytical models (extend surface to 3-parameters)
  - Spatial decomposition or cell enumeration
  - Constructive solid geometry (CSG)
  - B-rep

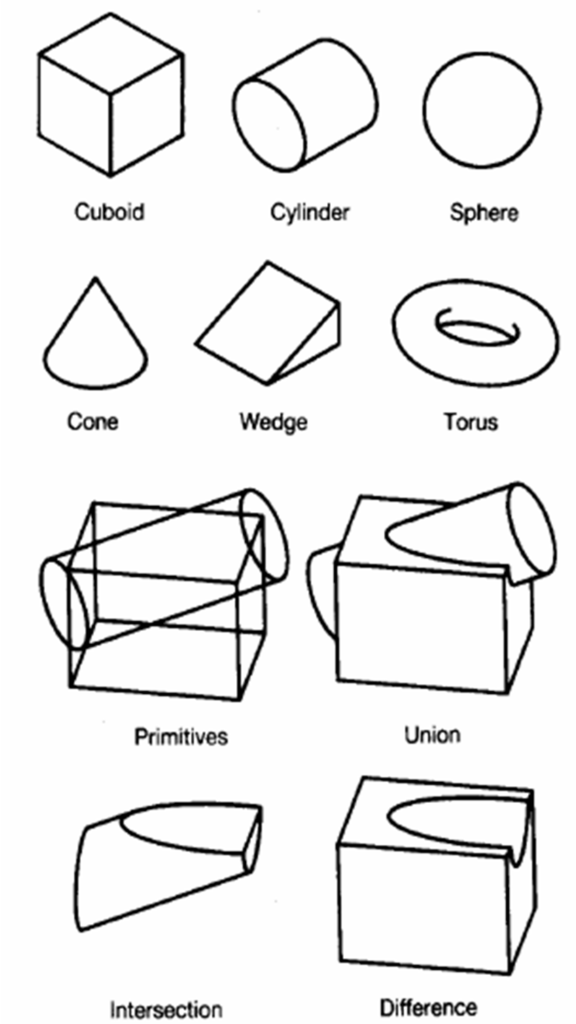


Quadtree (2D)

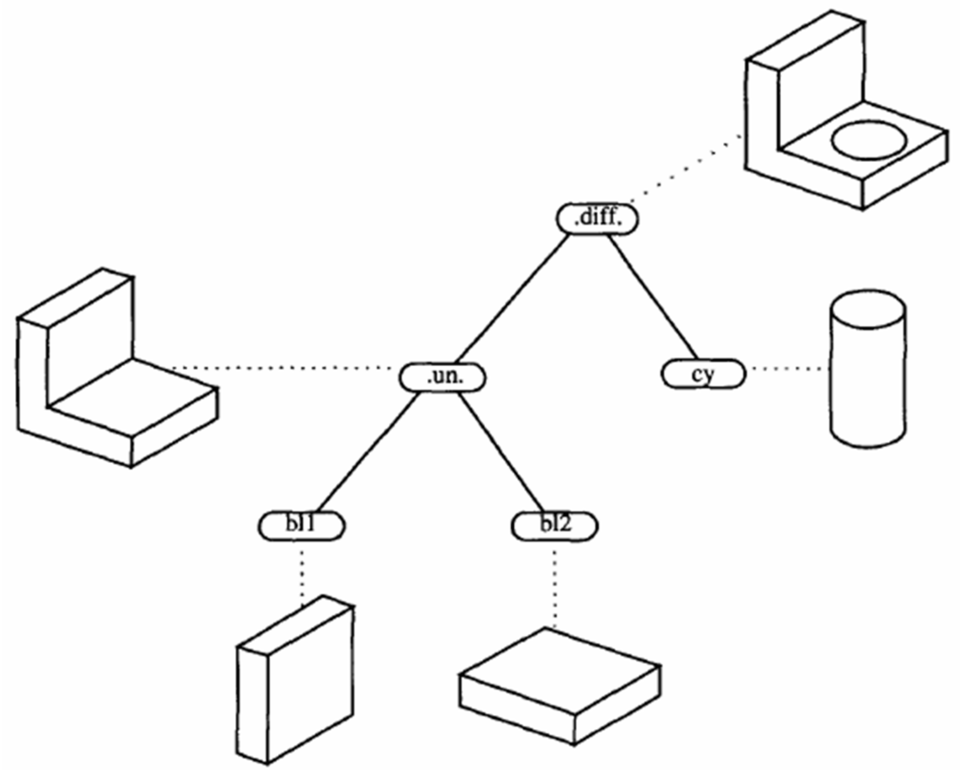


Quadtree (3D)

Basic 3D Models: Solid (4)



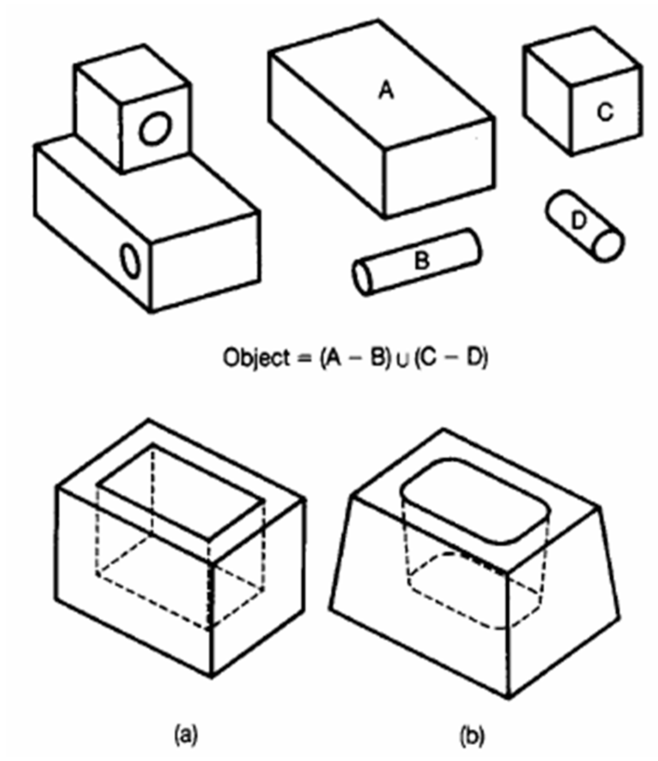
CSG Primitives and Boolean Operations



Graph-based CSG representation

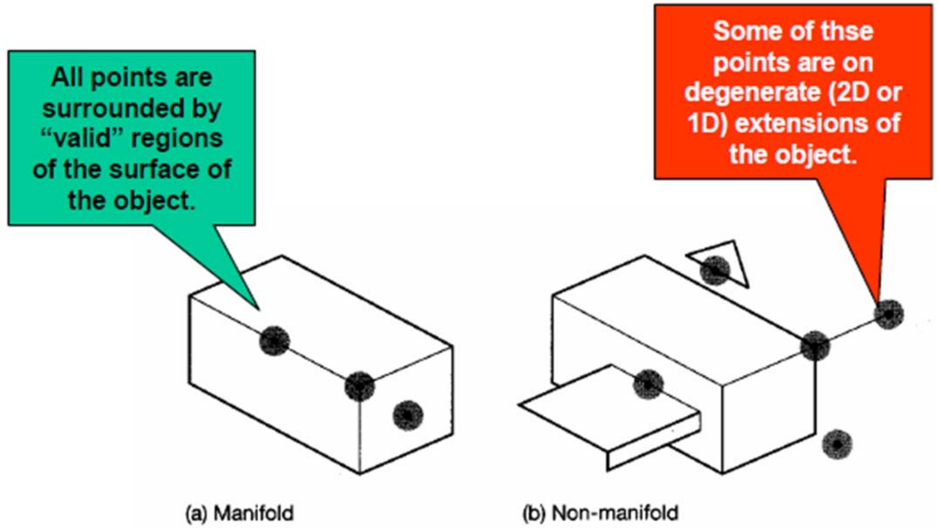
Basic 3D Models: Solid (5)

This is a "simple" CSG object...

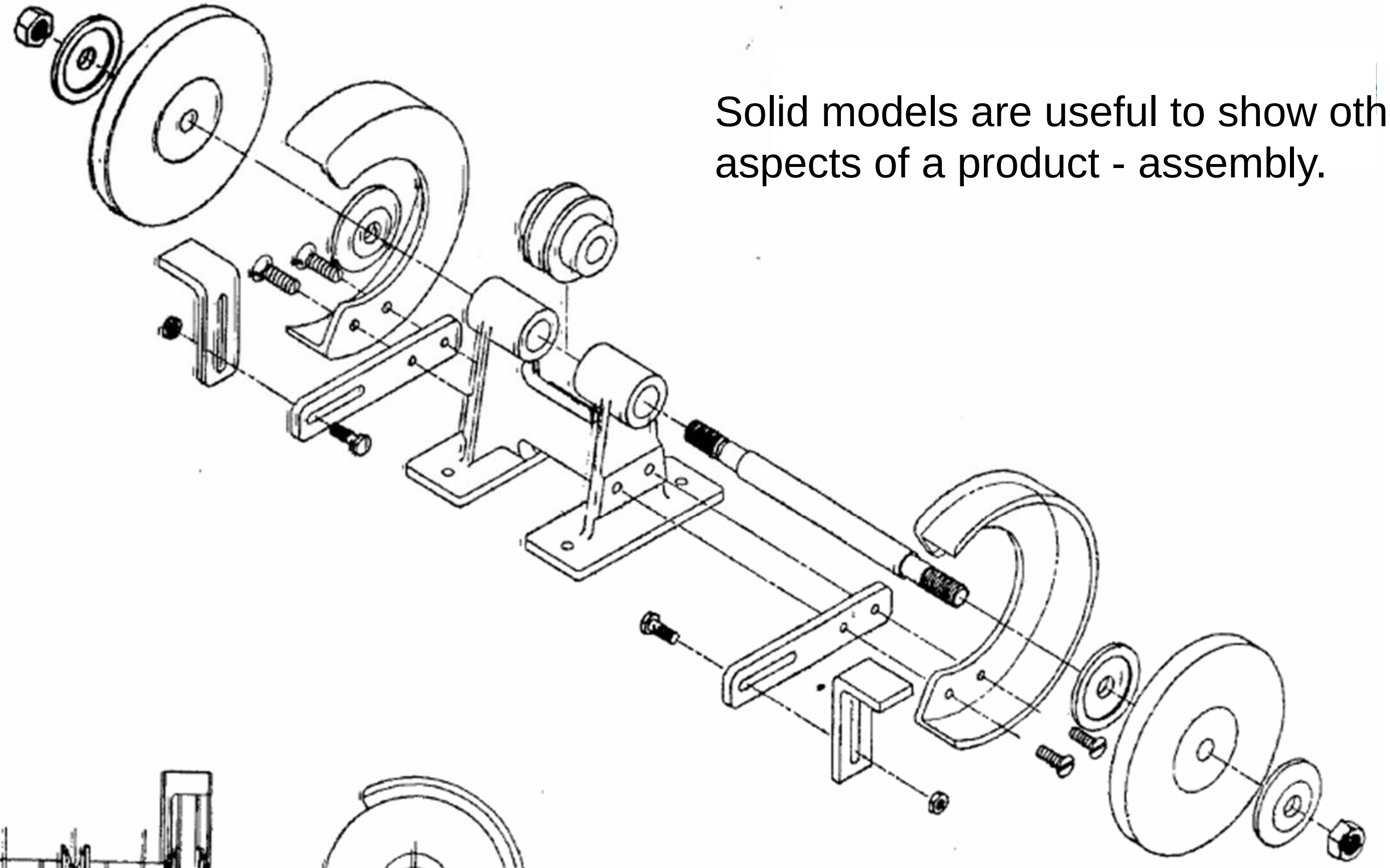


This is a much more complex CSG object!
(can you see why this is so?)

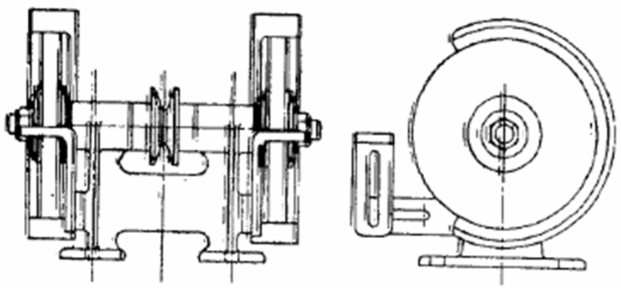
Solid models can have problems...



NOTE: By restricting the types of Boolean operations that are allowed, we can avoid most of these degeneracies.



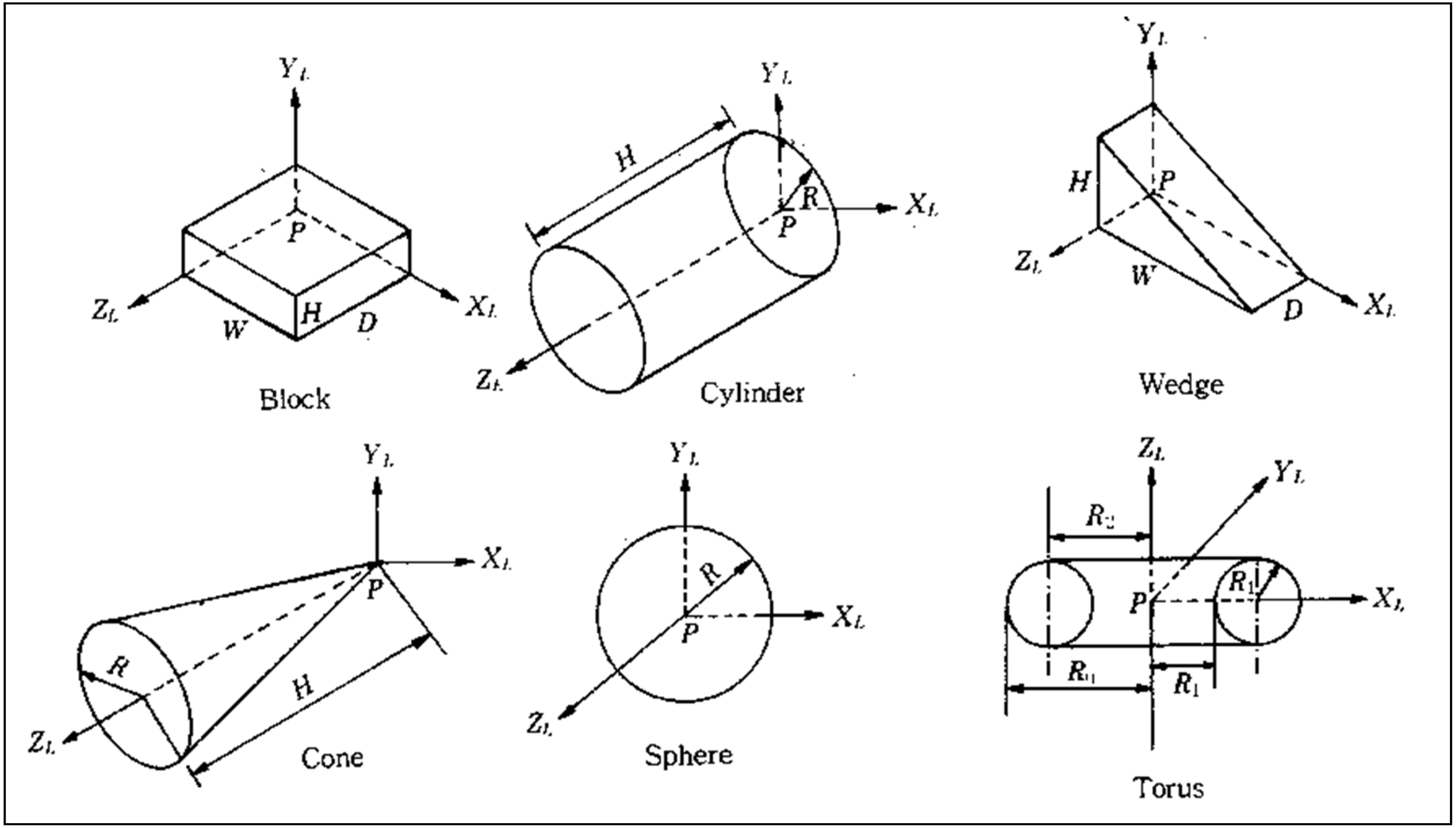
Solid models are useful to show other aspects of a product - assembly.



Solid Modeling Functions

- Primitive Creation Functions + Boolean Operations
- Surface Moving Functions
  - Sweeping, Swinging (used for Parametric Modeling)
  - Skinning
- Local Modification Functions
  - Rounding(or Blending, or Filleting), Lifting
- Boundary Modeling
- Feature-Based Modeling
- Parametric Modeling

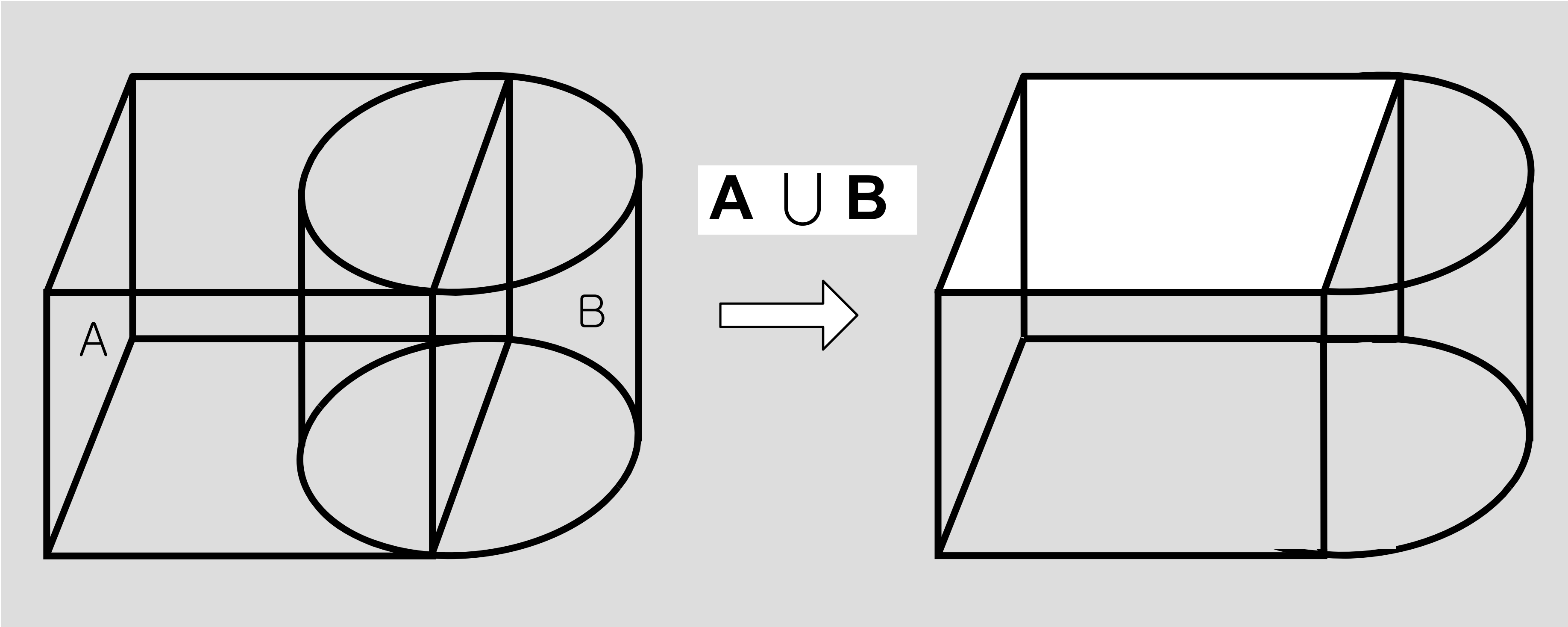
Primitive Creation Functions



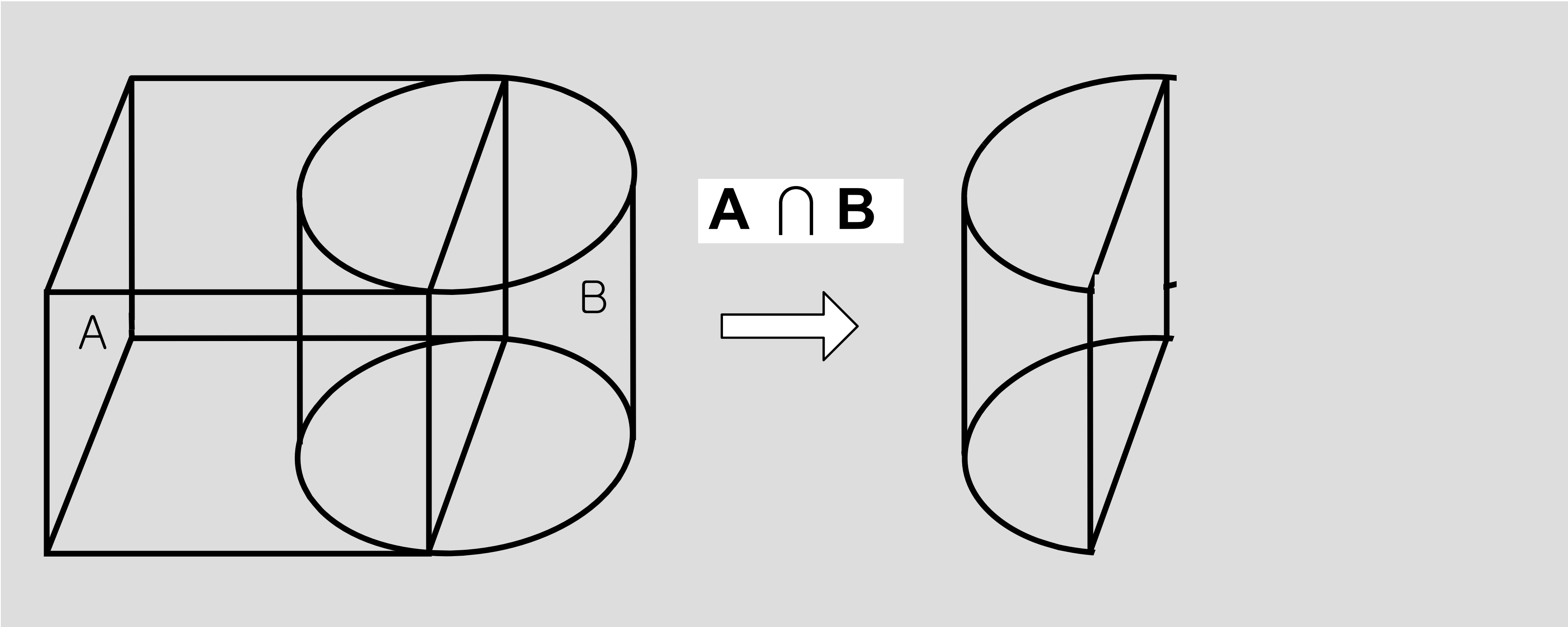
Boolean Operations

- Basic Idea:
  - Each primitive solid is assumed to be a set of points, a Boolean operation is performed on point sets, and the result is a solid composed of the points resulting from the operation.
- Boolean Operations
  - Union
  - Intersection
  - Difference
  - (Similar Operations: Sectioning and Gluing)

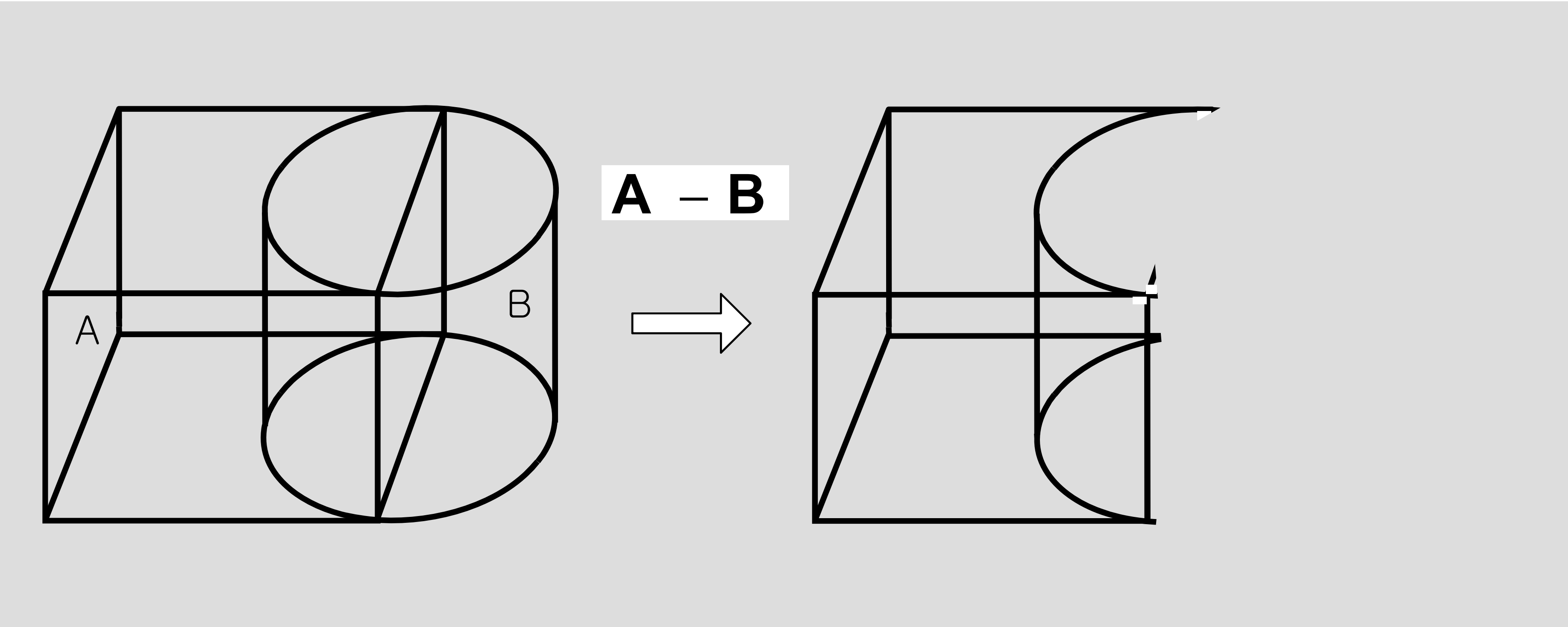
Union Operation



Intersection Operation

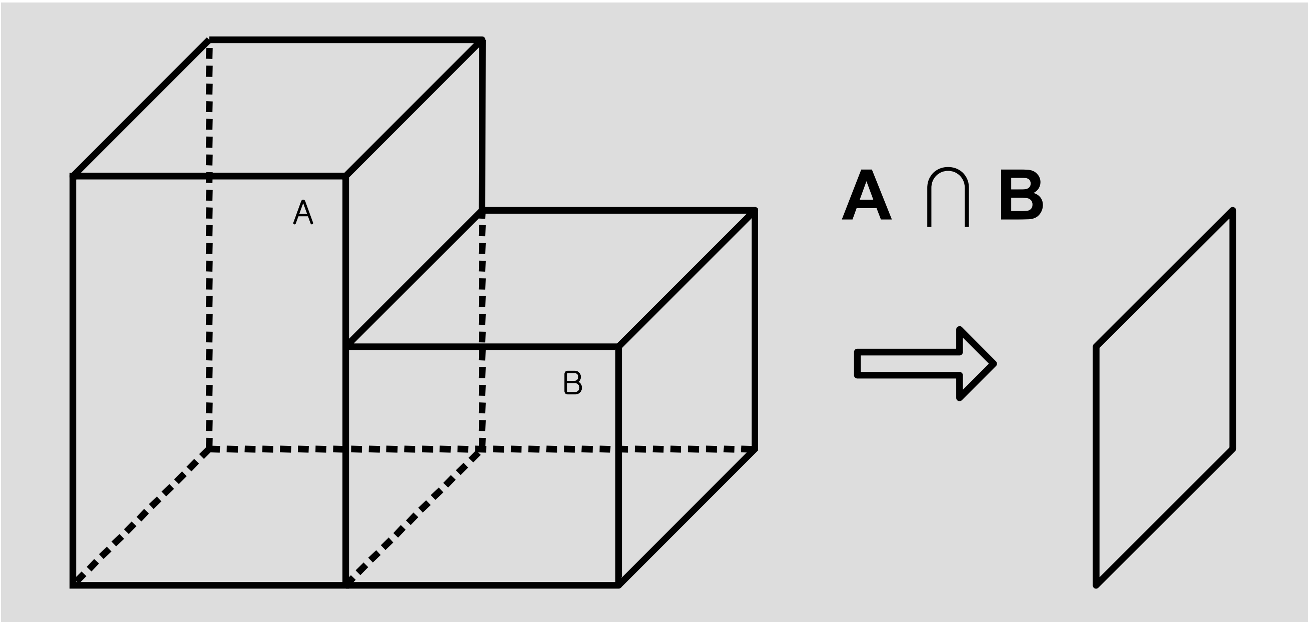


Difference Operations



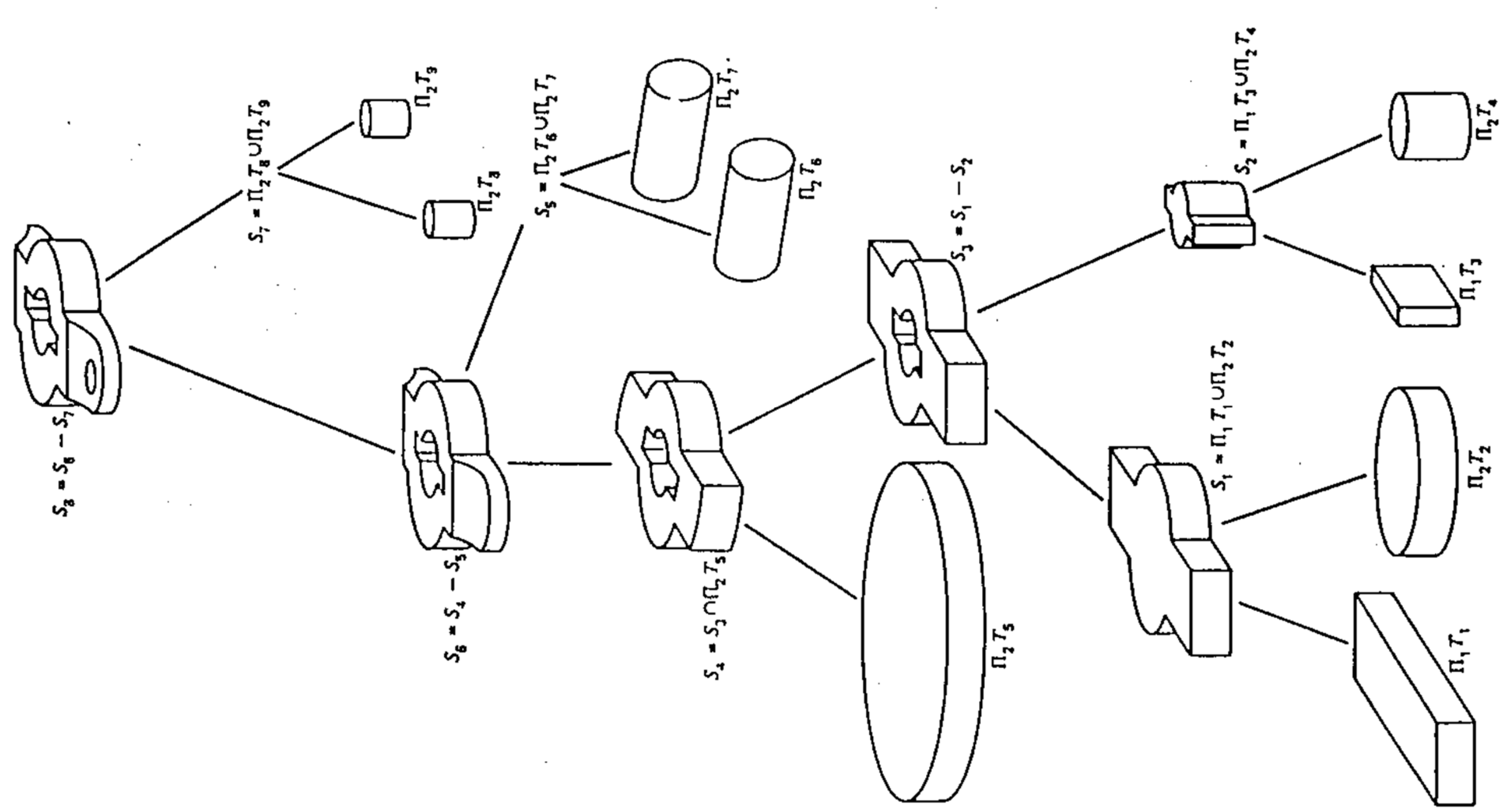
Limitation of Solid Models in Boolean Operations

- Solid models are not closed to Boolean operations

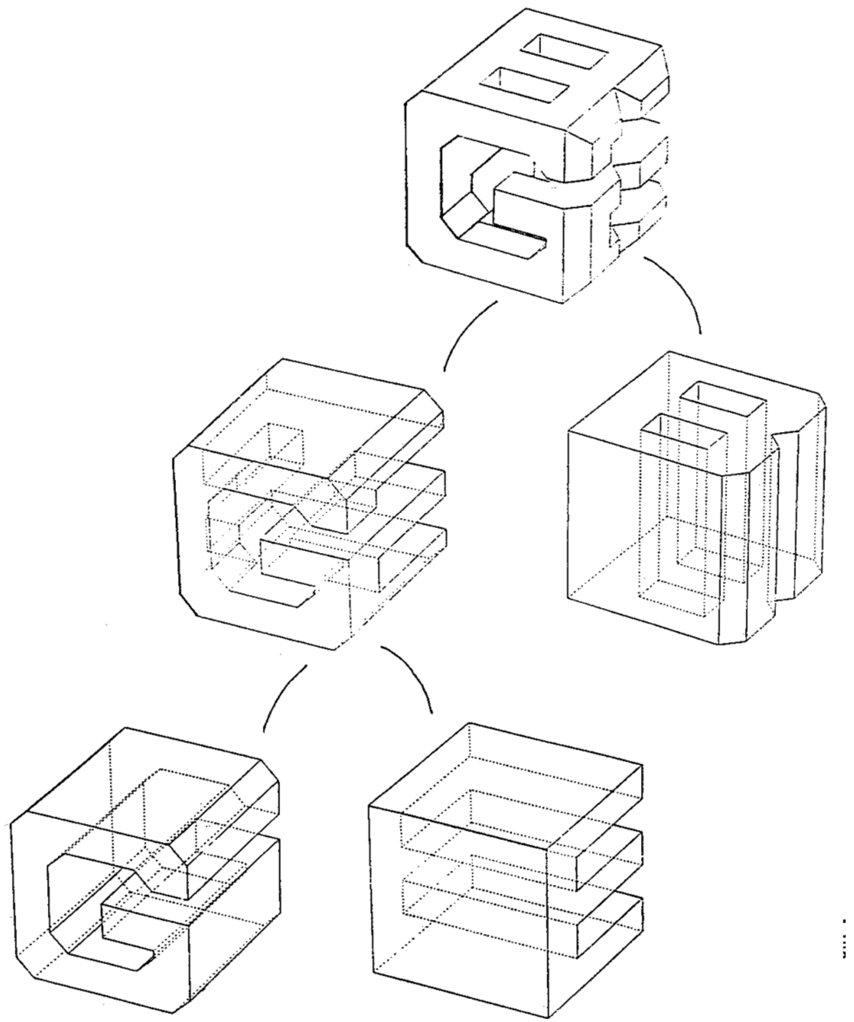


Result is $\varnothing$ in Solid Modeler

Example of Boolean Operations (1)

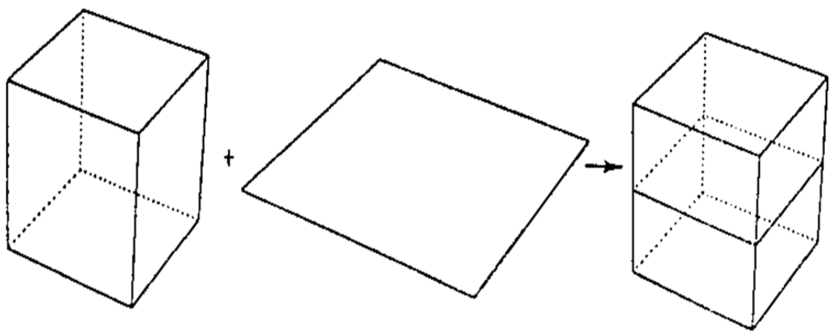


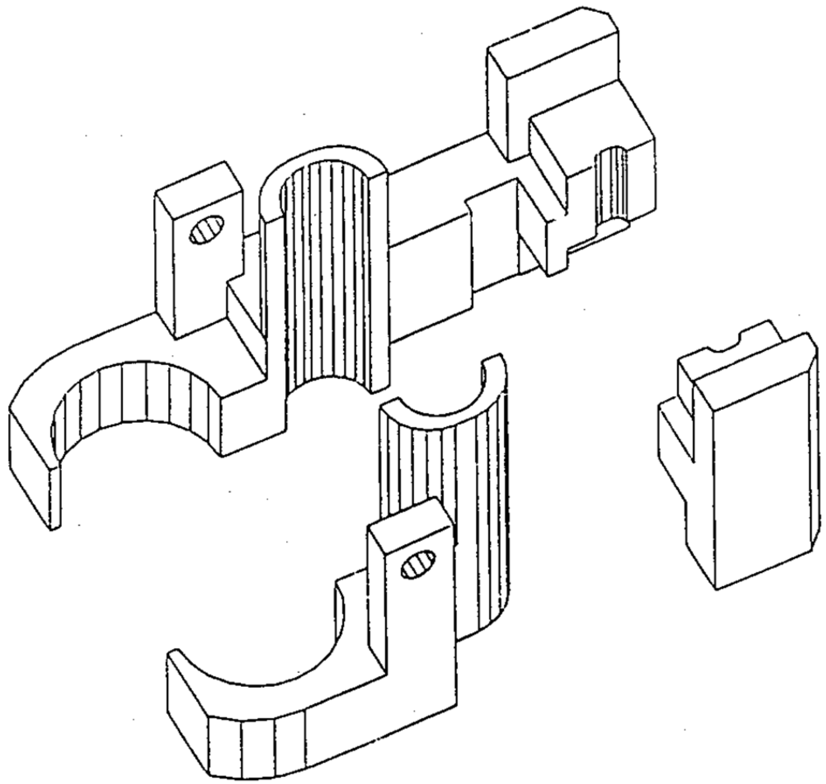
Example of Boolean Operations (2)



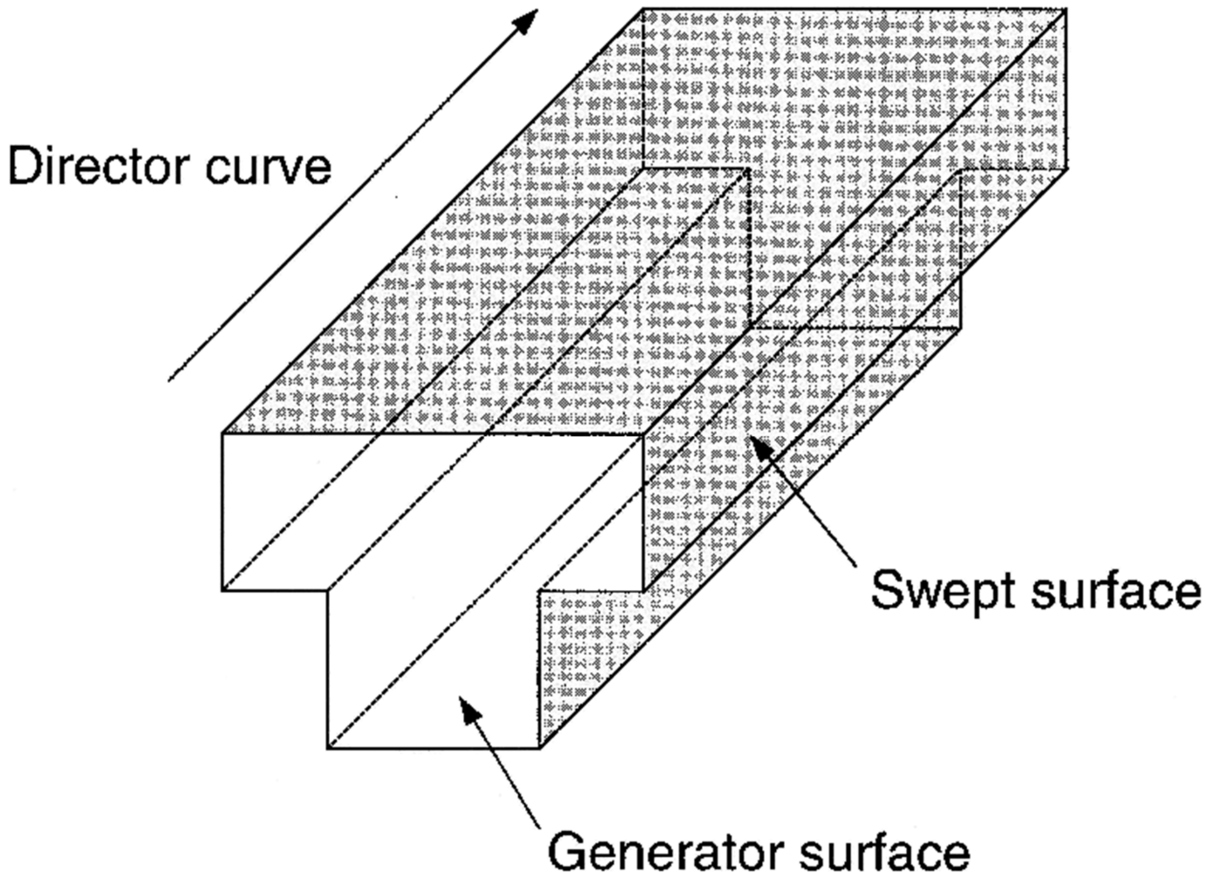
Sectioning (or Cutting)

- Useful for Cross-Sectional View

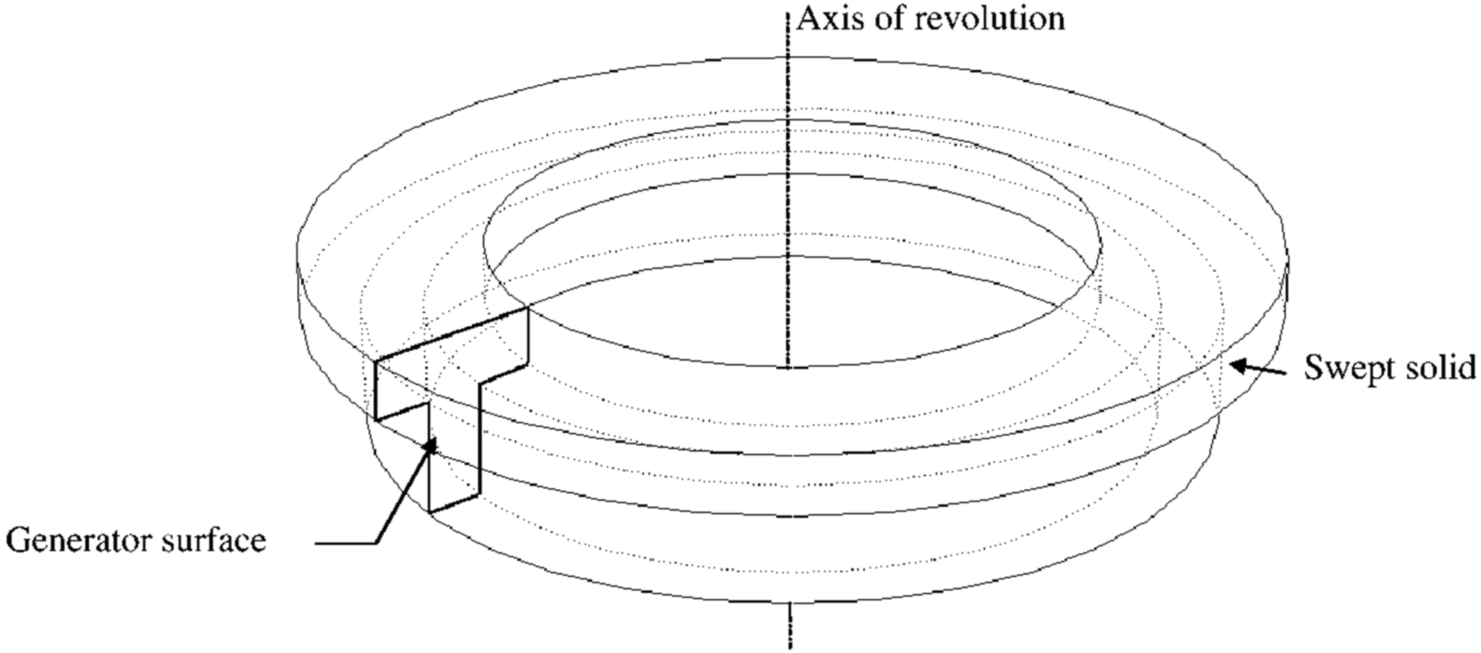




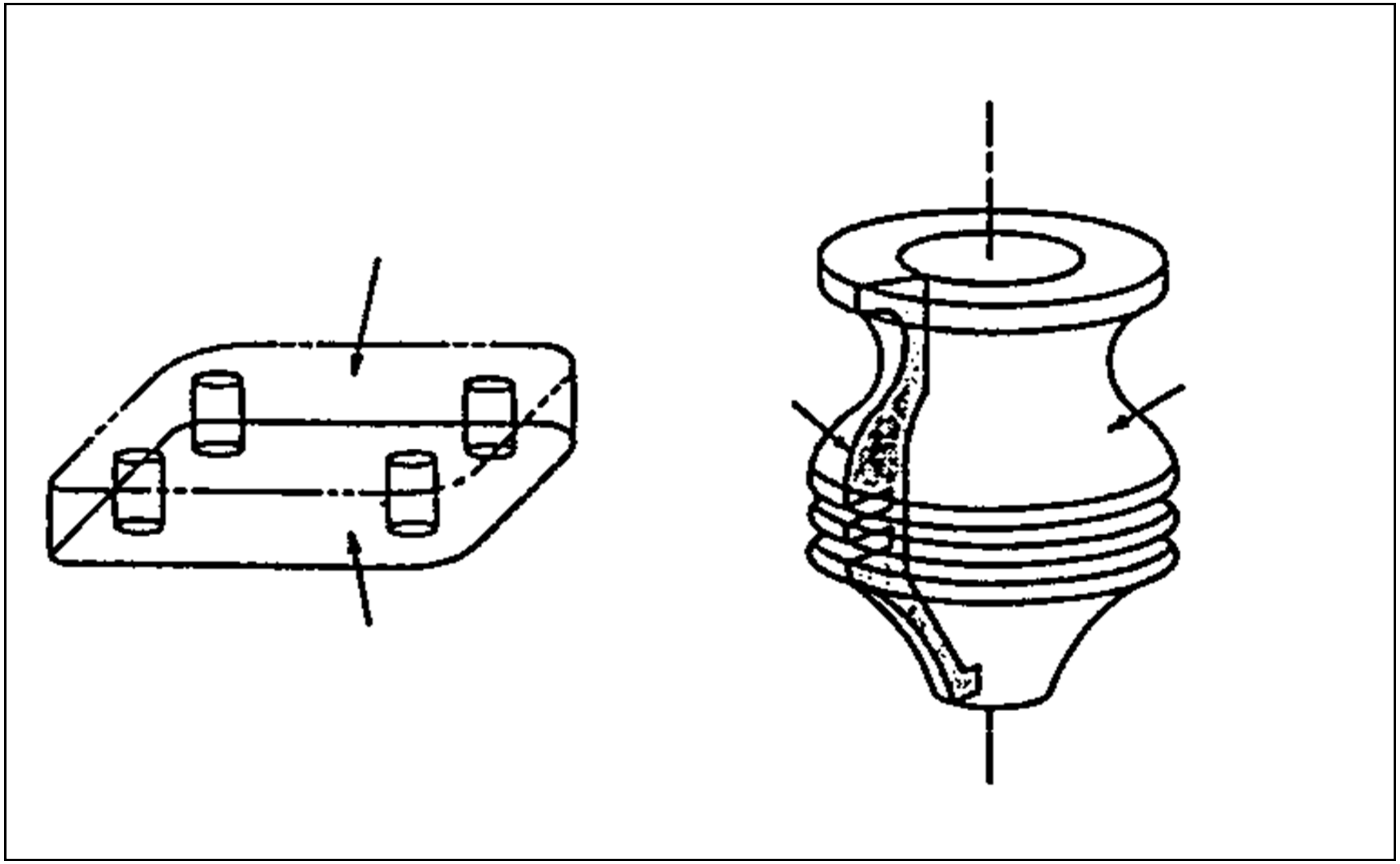
Translational Sweeping



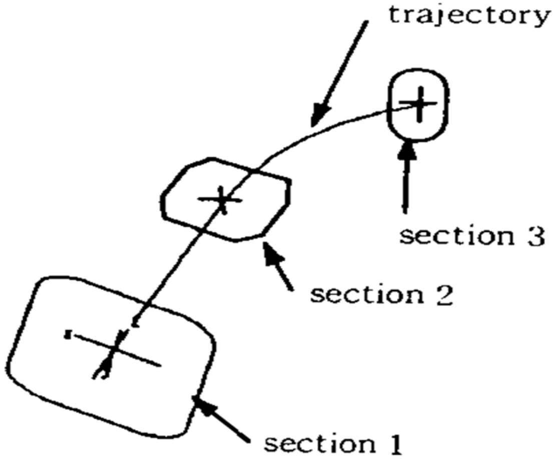
Rotational Sweeping (Swing)

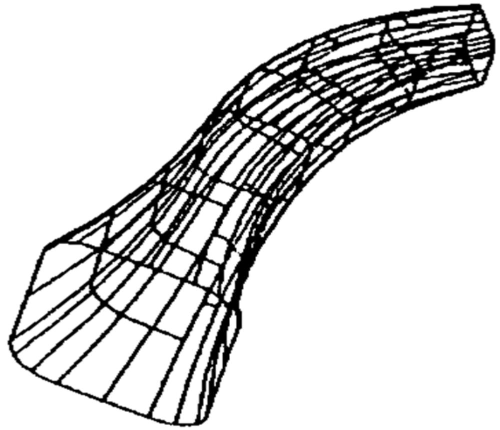


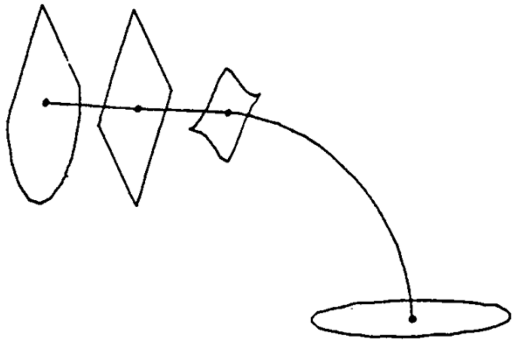
Examples of Sweeping Operations

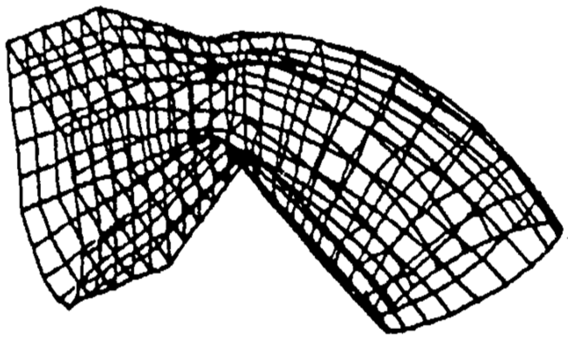


Skinning

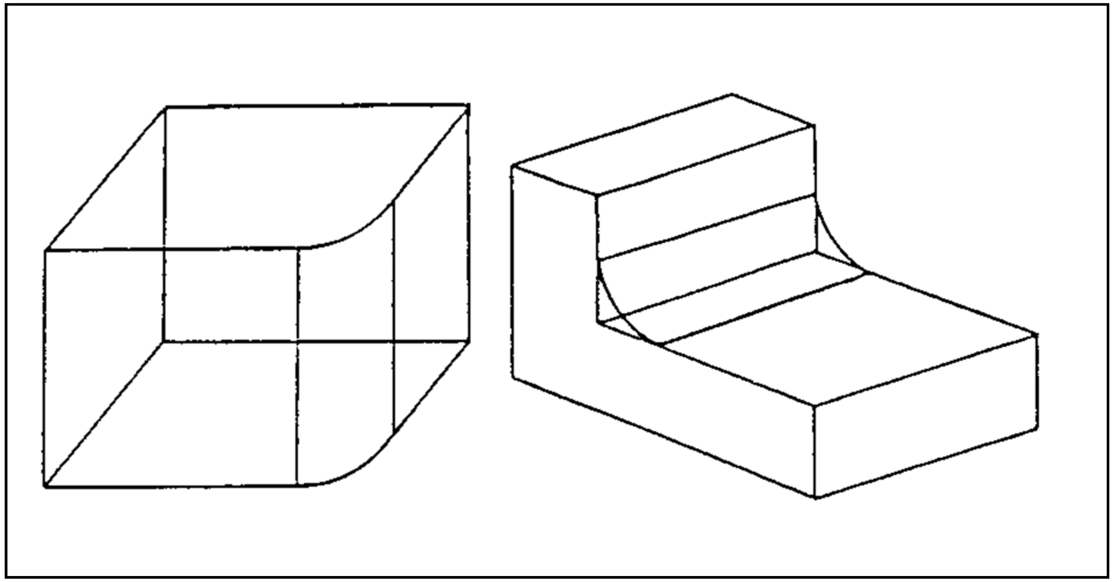




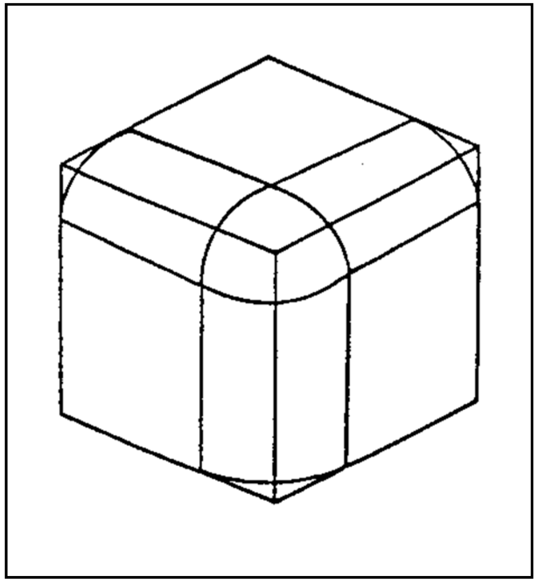




Rounding

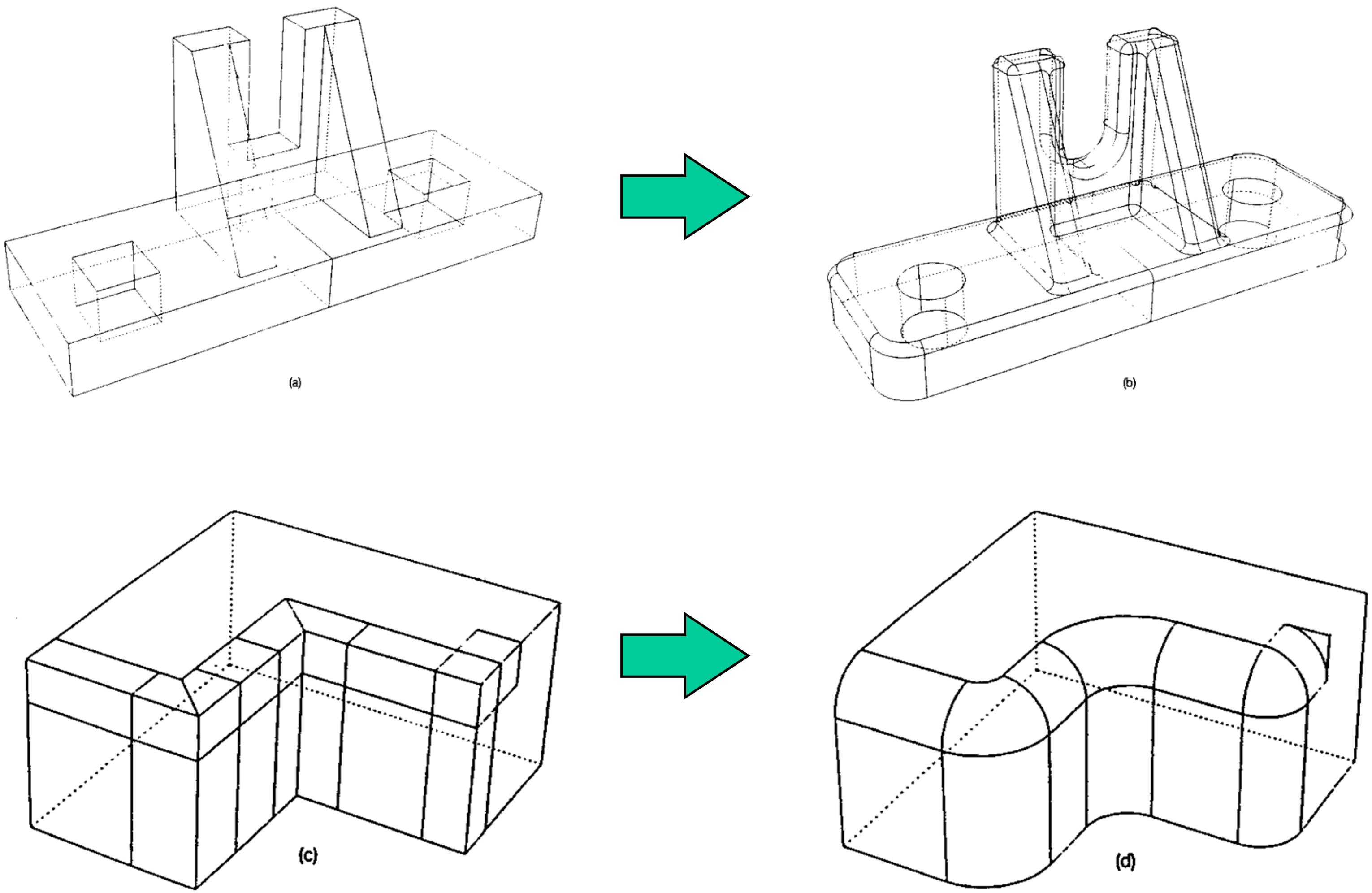


Edge Rounding

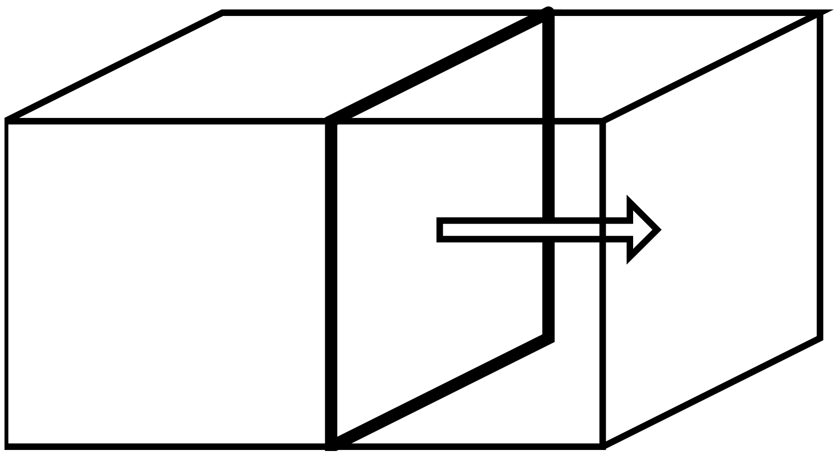


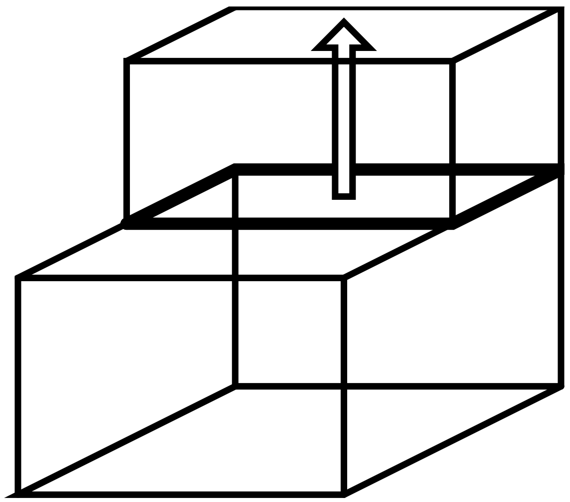
Vertex Rounding

Examples of Rounding Operation

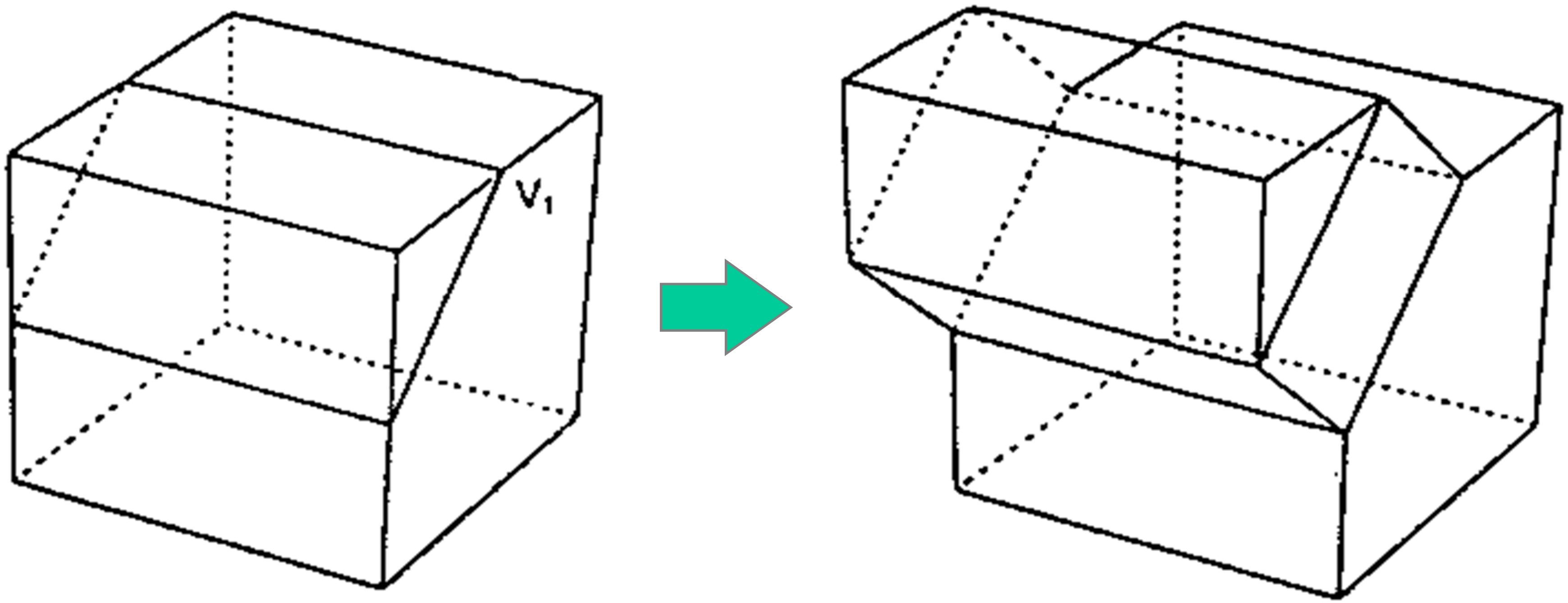


Lifting

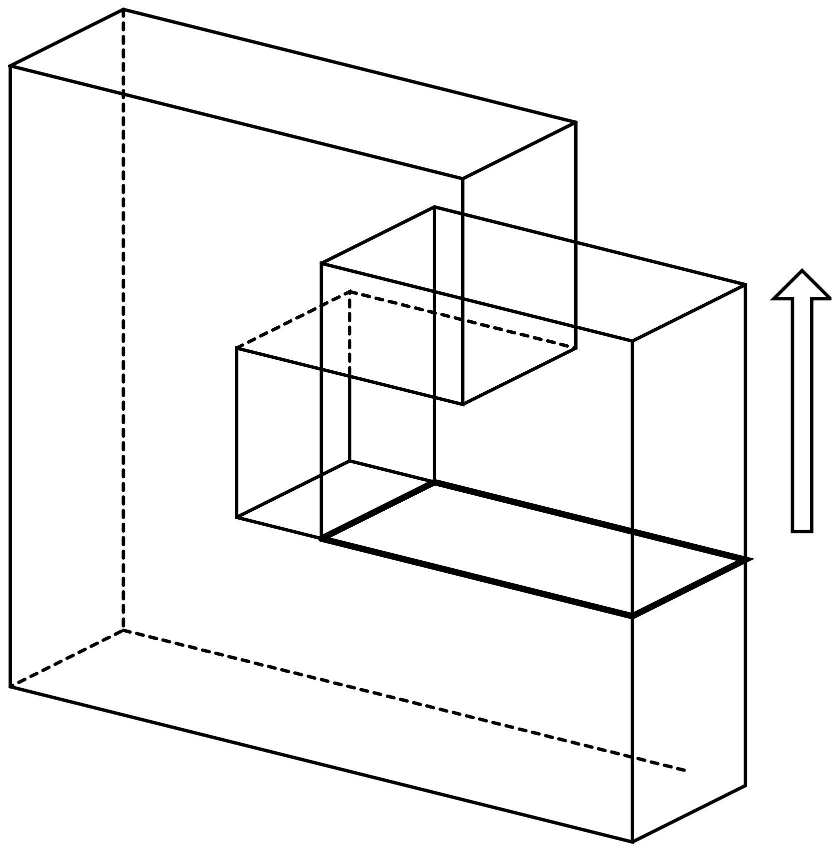




Lifting a Face Group

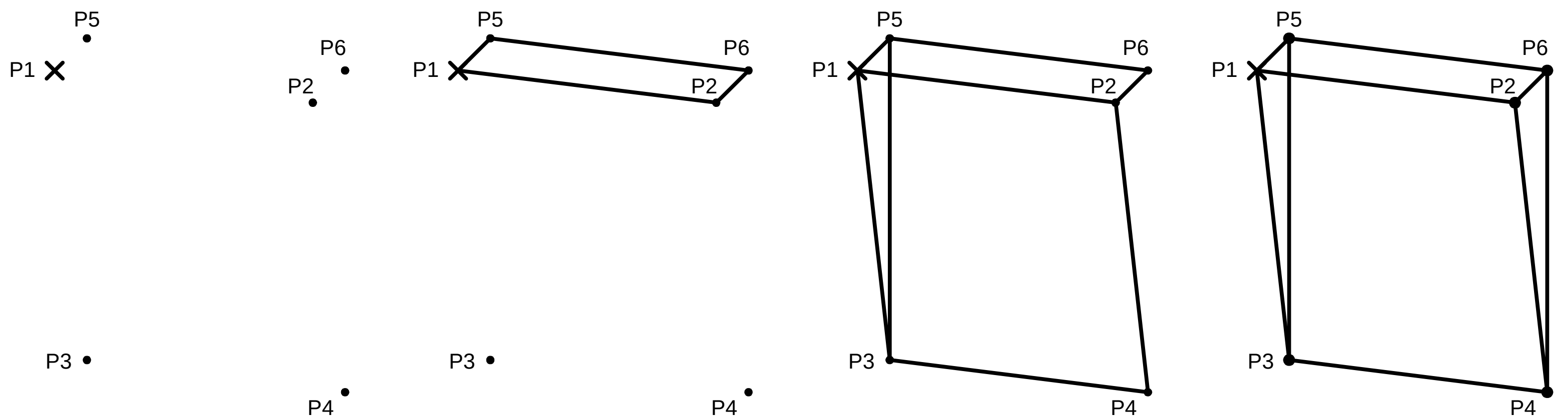


Self-Intersection Caused by Lifting



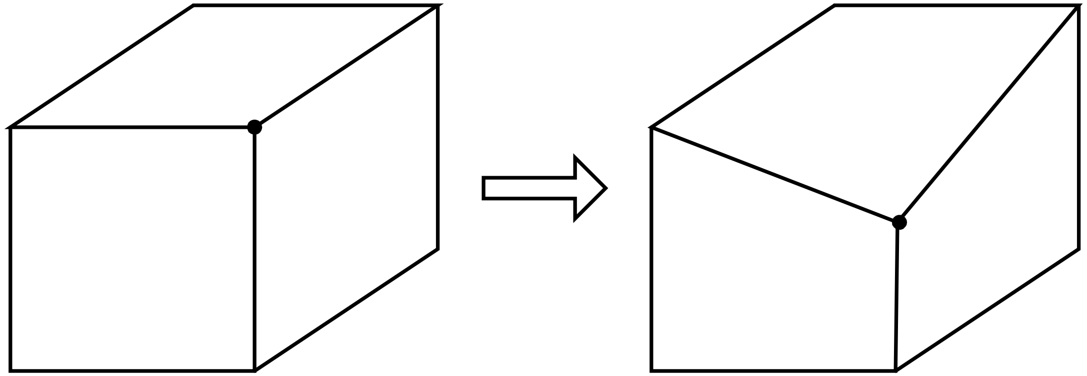
Boundary Modeling Functions

- Add, delete, or modify the lower entities of a solid, such as vertices, edges, and faces directly

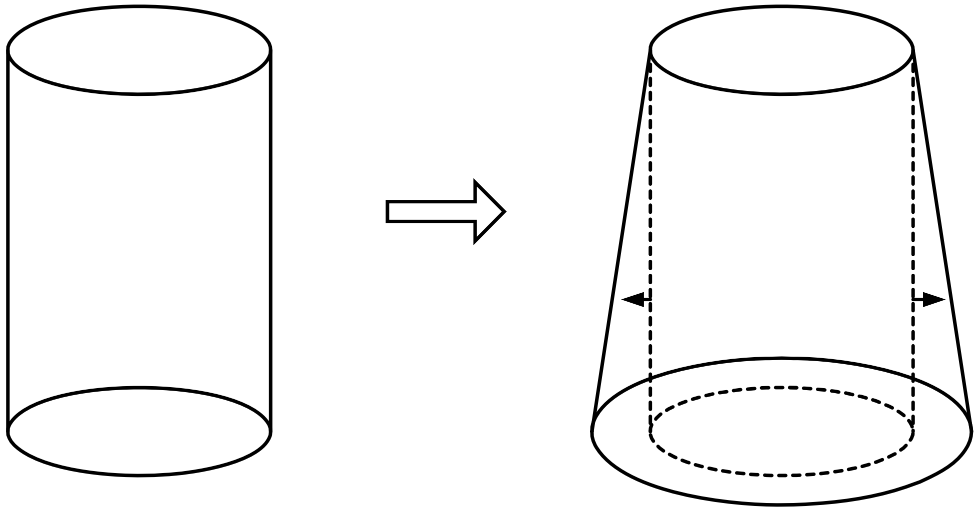


Tweaking

- Vertex Moving



- Surface Replacement

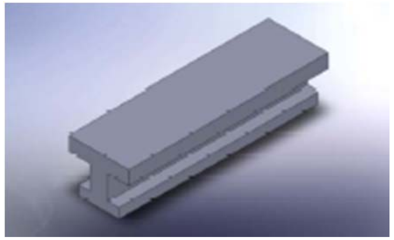


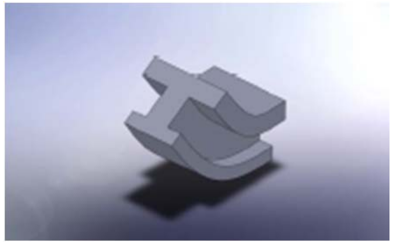
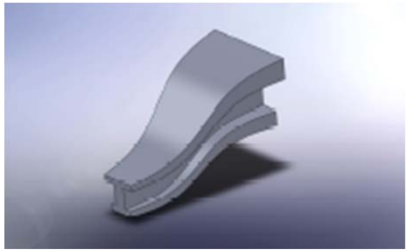
Feature-Based Modeling

- Features
  - Familiar shape units having engineering significance
  - Depend on application areas
  - Manufacturing features: Chamfer, Hole, Fillet, Slot, Pocket
  - Design features: Boss, Rib, etc.
- Feature-Based Modeling
  - Enables the designer to model solids by using features
  - "make a hole of a certain size at a certain place" instead of traditional solid modeling commands like "create a cylinder of a certain size at a certain place and subtract it from the base body"

Features

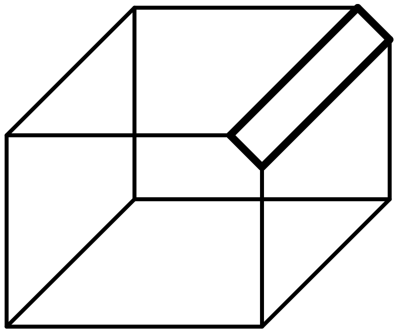
- Sketched: 2D geometry (sketch) swept along a 3D path
  - Extrude
  - Revolve
  - Sweep
  - Loft



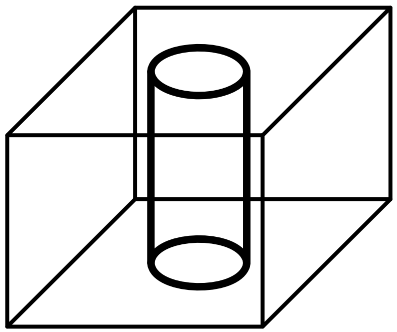



- Applied: Attached to existing geometry (eg. edges, faces)
  - Fillet
  - Pattern
  - Shell
  - Draft
  - Rib

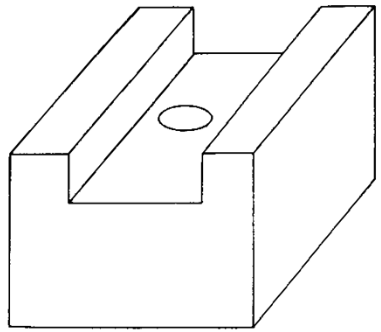
Manufacturing Features



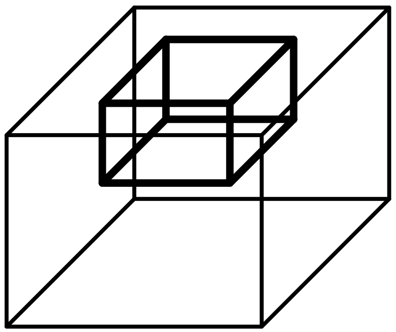
Chamfer



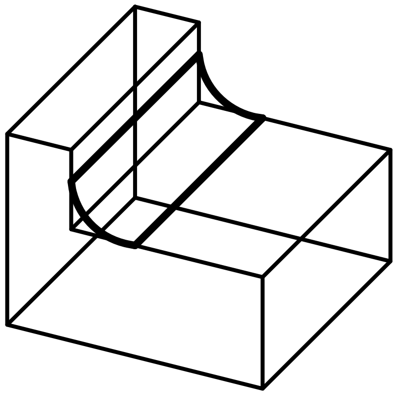
Hole



Slot



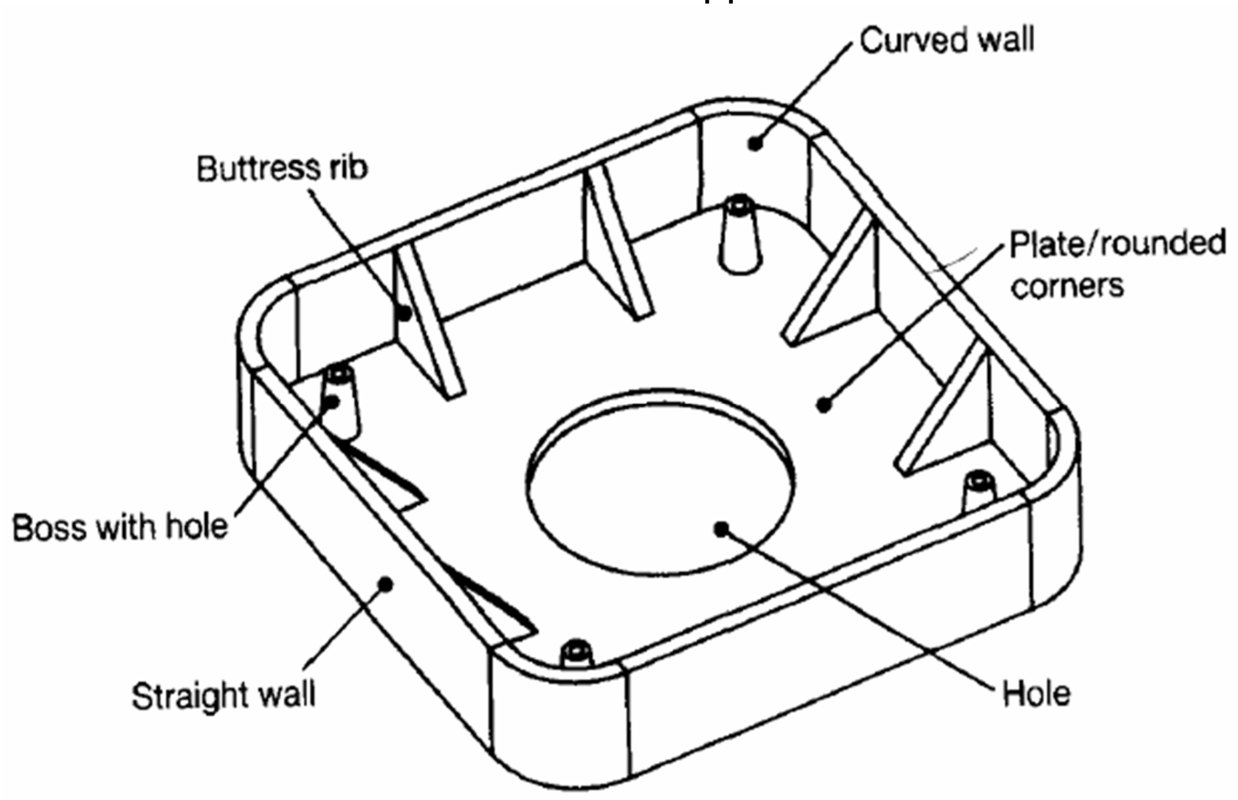
Pocket



Fillet

Example

A “feature-based” approach



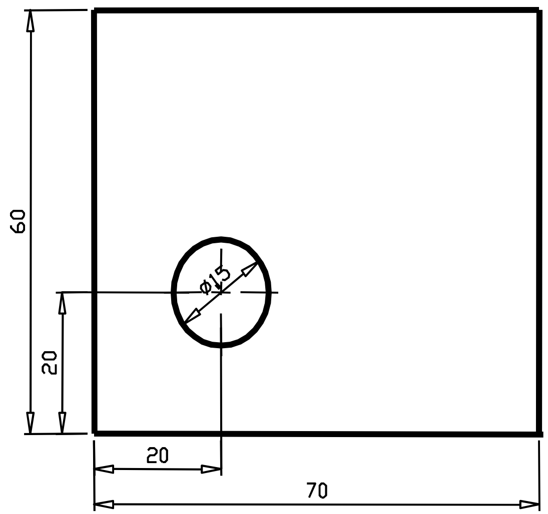
This is much easier to relate to common manufacturing operations and avoids creating parts that are impossible to manufacture using conventional tooling.

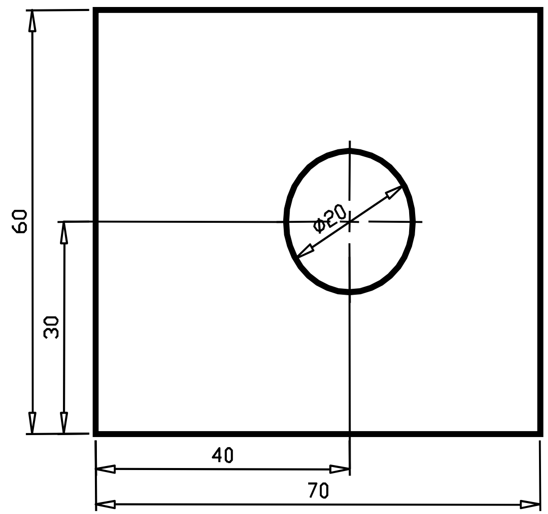
Parametric Modeling

- Allow the designer model a shape by using geometric constraints and dimension data on its elements
- Geometric Constraints
  - Describe the relation between the elements
  - Two faces are parallel, Two edges lie in a plane, etc
- Dimension Data
  - Include not only the dimensions but also the relations between the dimensions in the form of mathematical equations.
- Parametric Modeling
  - Construct the required shape by solving the equations that express the geometric constraints, those derived from the dimensions, and those obtained from the dimensional relations

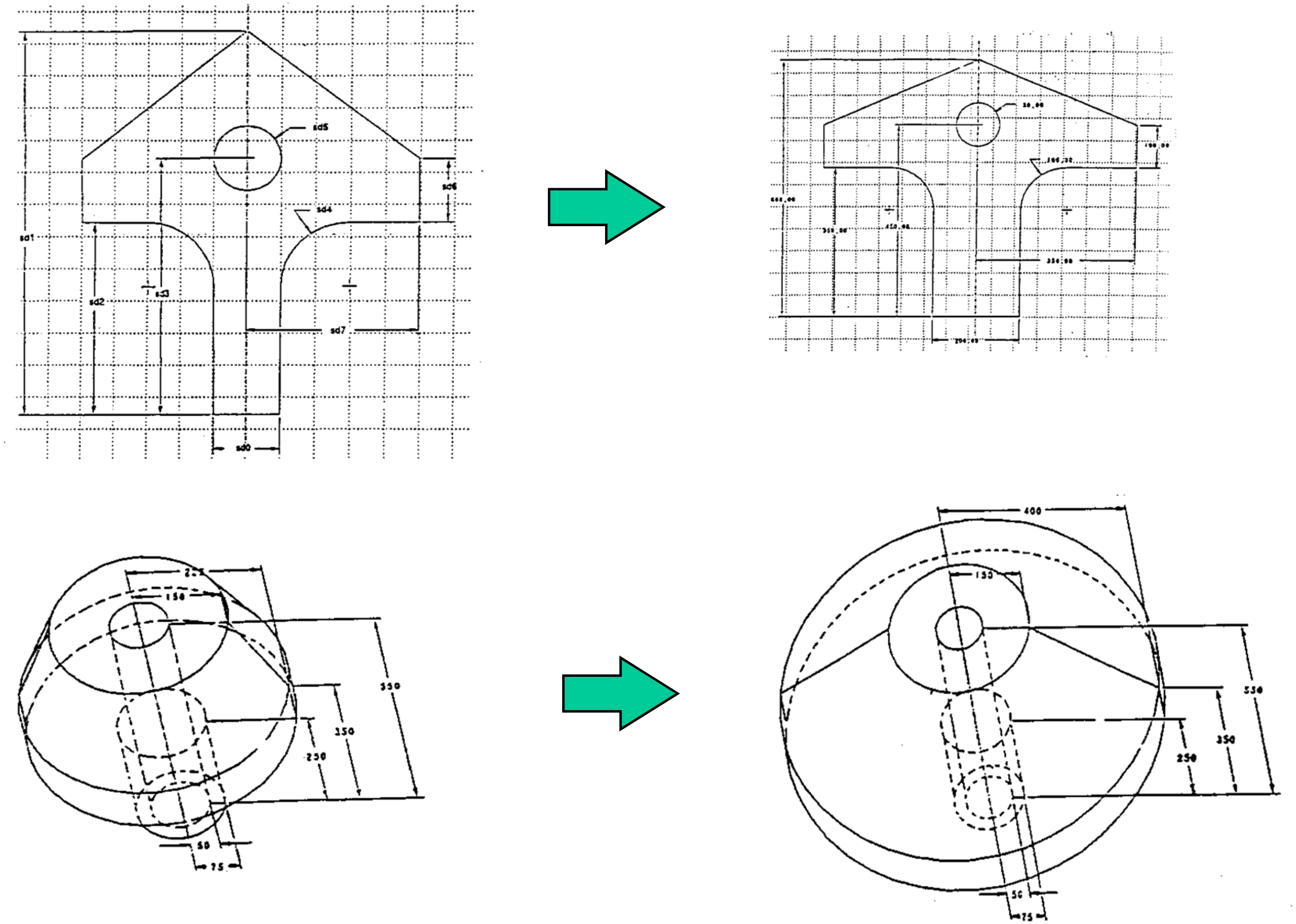
Parametric Modeling Sequence

- Input a 2-D shape as a rough sketch
- Input geometric constraints and dimension data
- Reconstruct the 2-D shape for the input of step2
- Repeat steps 2 and 3 until the desired model is obtained
- Create a 3-D shape by sweeping or swing the 2-D shape

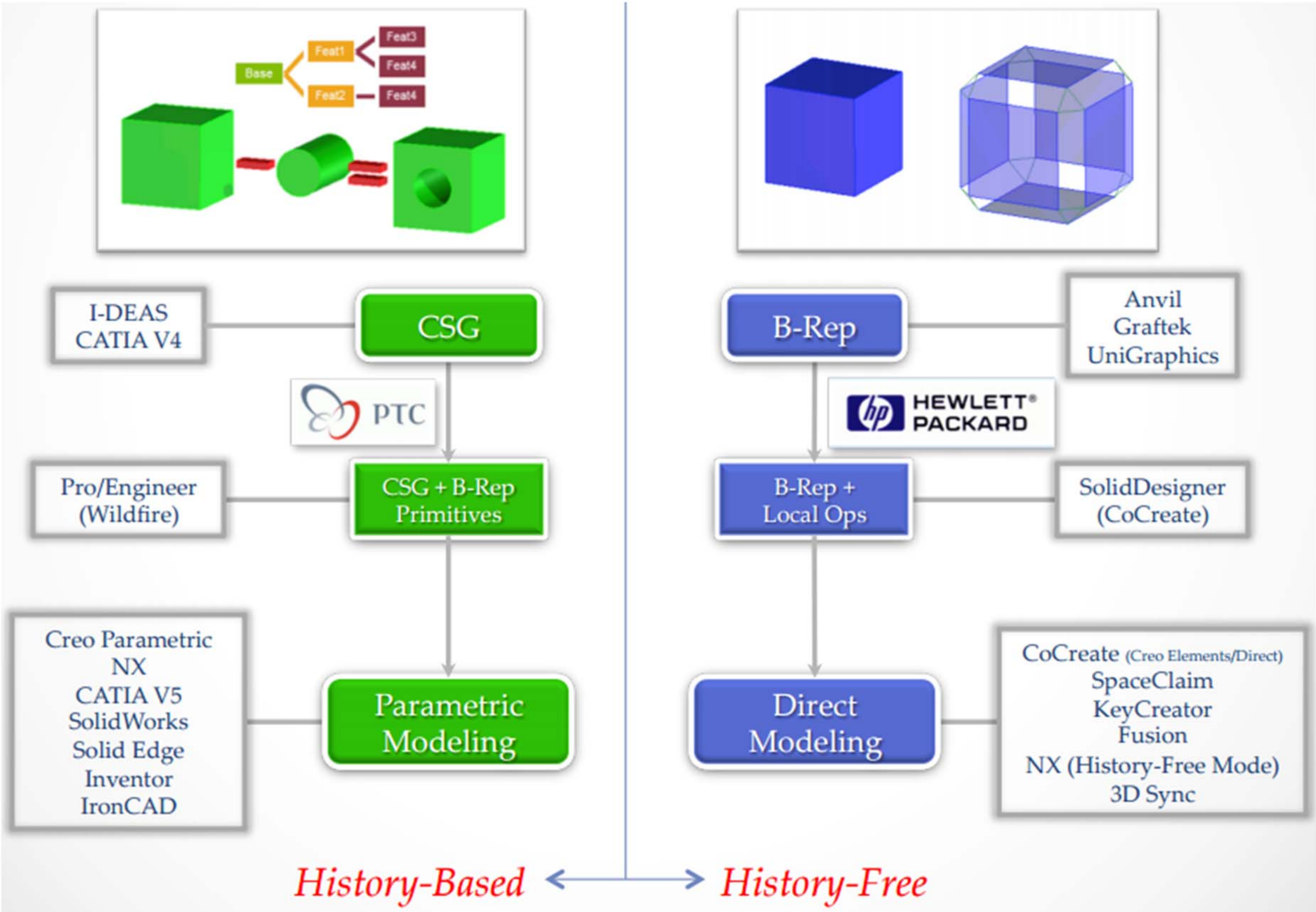




Examples of Parametric Modeling



Solid Modeling



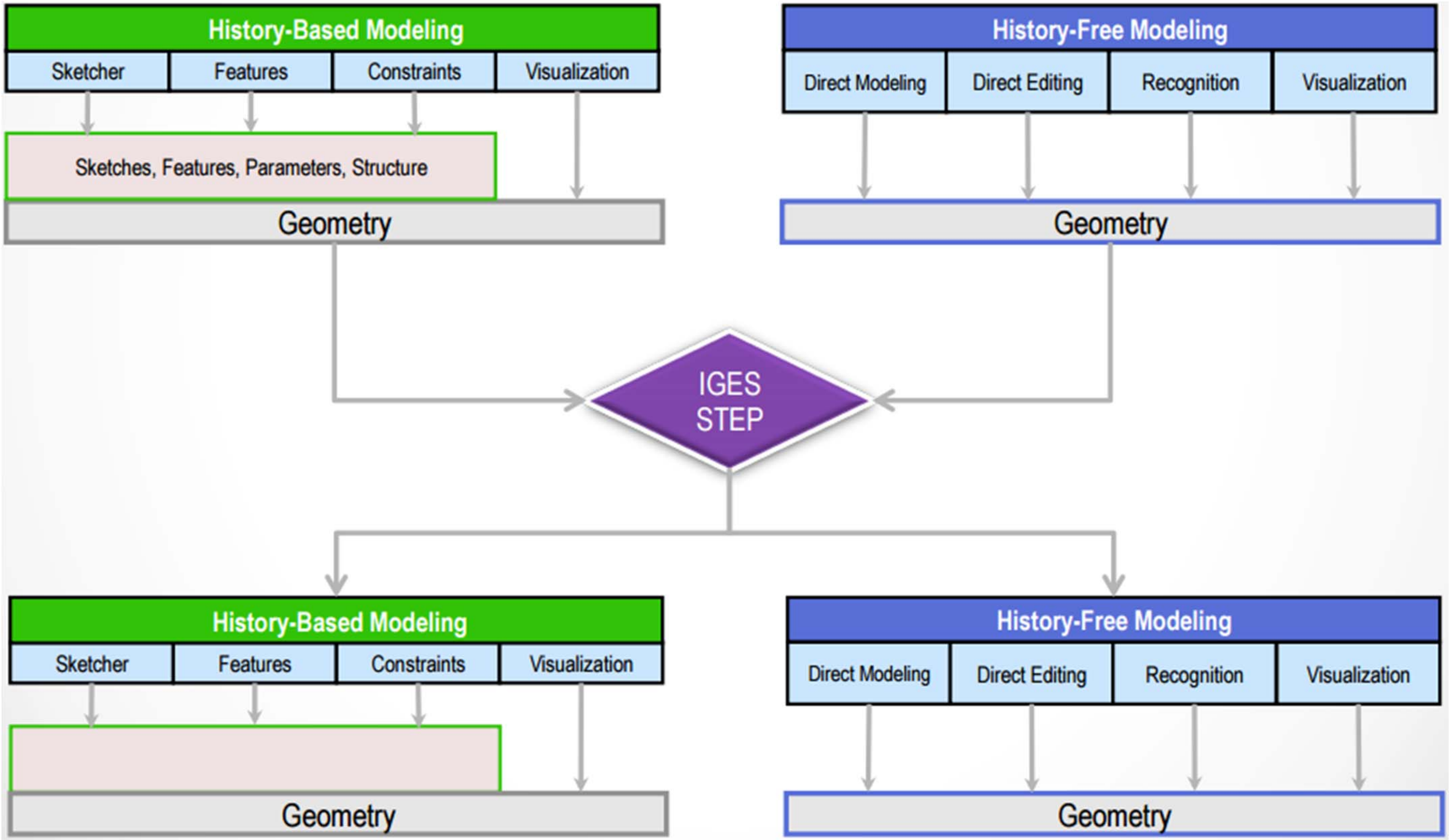
Technology Differences

- Parametric (history-based) approach
  - Structured modeling process
  - The history tree is the master
  - Constrained sketching
  - Inherent parent/child relationships
  - Part/assembly modes
  - Edits are typically indirect
  - Linear parameters
  - Direct edits are ordered in tree
  - Design intent defined via modeling process
- Direct (history-free) approach
  - Flexible modeling process
  - The geometry is the master
  - Flexible sketching
  - No parent/child relationship
  - No part/assembly mode
  - Edits are typically direct
  - Synchronous parameters
  - Direct and indirect edits just change geo
  - Design intent defined as needed

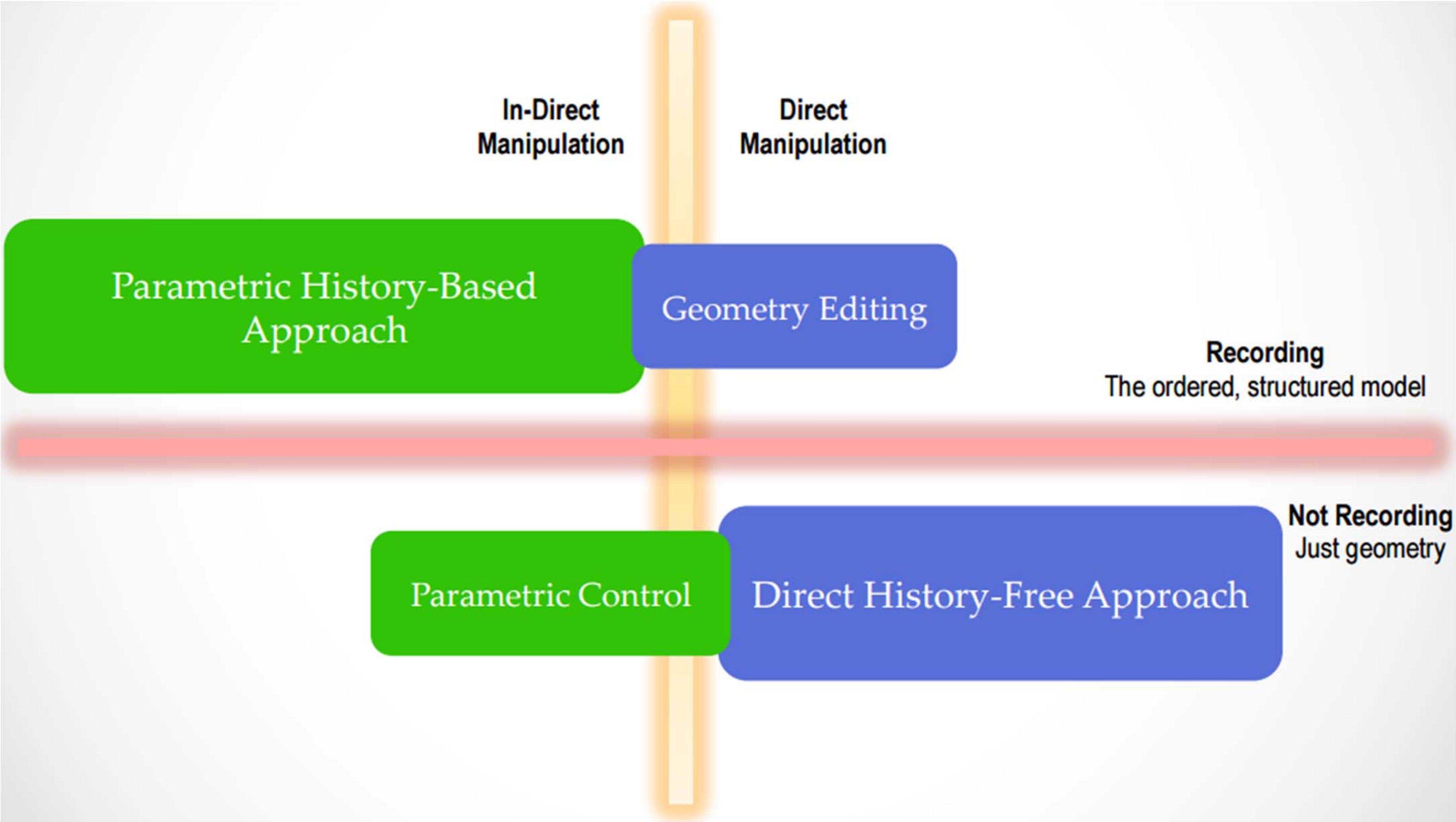
Key Value Opportunities

- History-Based Modeling
  - Use the parent/child relationship to:
    - Enable design automation
    - Create platforms and families
  - This is a powerful and rich approach
    - Where the product strategy is family-based or platform-driven
- History-Free Modeling
  - Quickly and easily create 3D designs
  - Create and modify the model through direct interaction
  - Make radical changes at any time
  - Edit geometry from any CAD source
  - This is a lightweight/flexible approach
    - Where the value of front loading a design with engineering constraints and relationships does not carry forward

Data Exchange

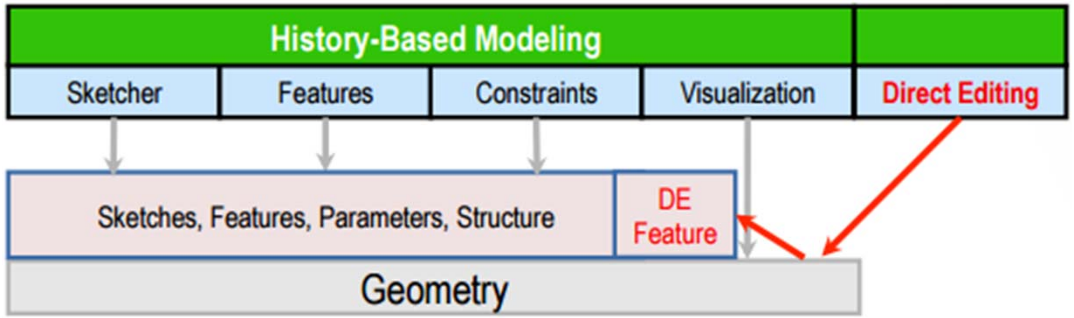


Merging the Technologies: Best of Both (1)

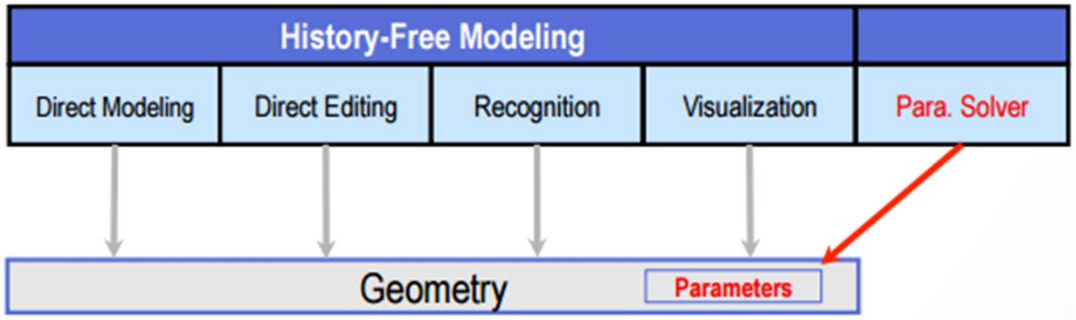


Merging the Technologies: Best of Both (2)

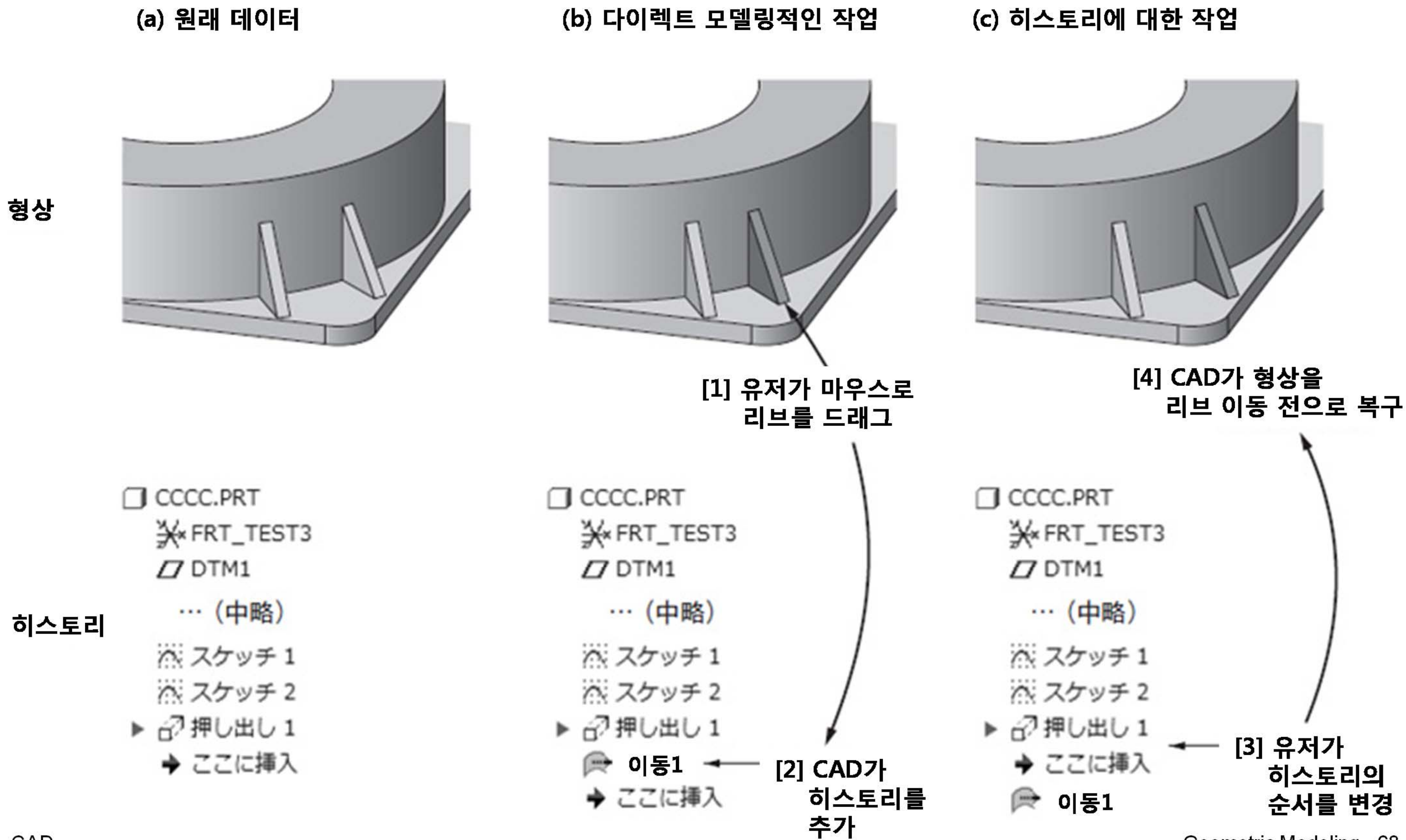
- Adding Direct Editing to History-Based Modeling



- Adding Parametric Control to History-Free Geometry



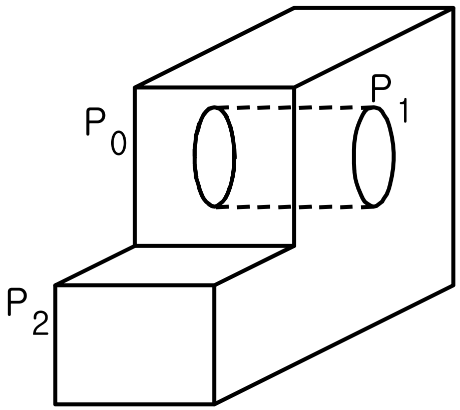
Example

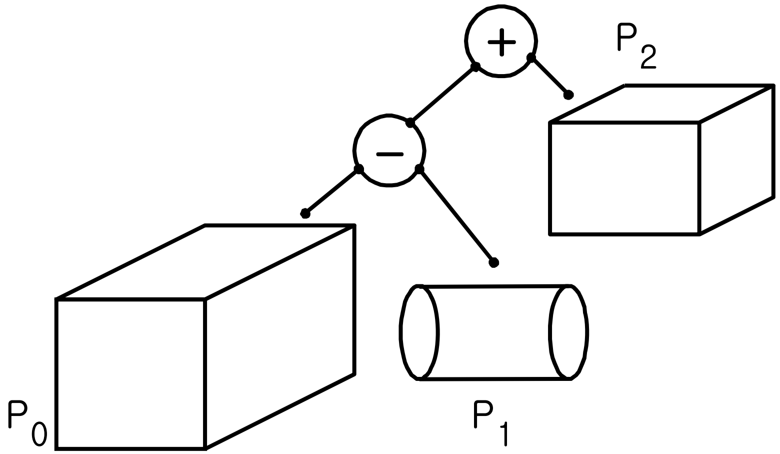

(a) 원래 데이터
(b) 다이렉트 모델링적인 작업
(c) 히스토리에 대한 작업
형상
히스토리
CCCC.PRT
FRT_TEST3
DTM1
… (中略)
スケッチ 1
スケッチ 2
押し出し 1
ここに挿入
[1] 유저가 마우스로 리브를 드래그
이동1
[2] CAD가 히스토리를 추가
[3] 유저가 히스토리의 순서를 변경
[4] CAD가 형상을 리브 이동 전으로 복구

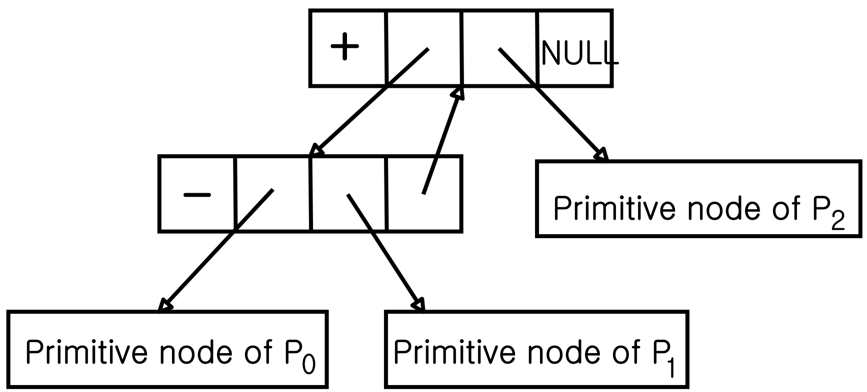
Data Structure for Solid Models

- CSG (Constructive Solid Geometry) Tree Structure
- B-Rep (Boundary Representation) Data Structure
  - Half-edge data structure
  - Winged-edge data structure
- Decomposition model structure
  - Voxel representation
  - Octree representation
  - Cell representation

CSG Tree Structure

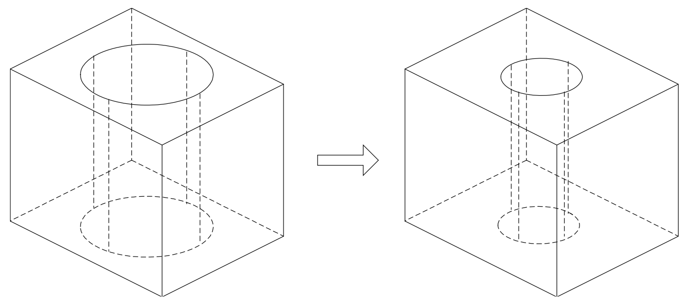






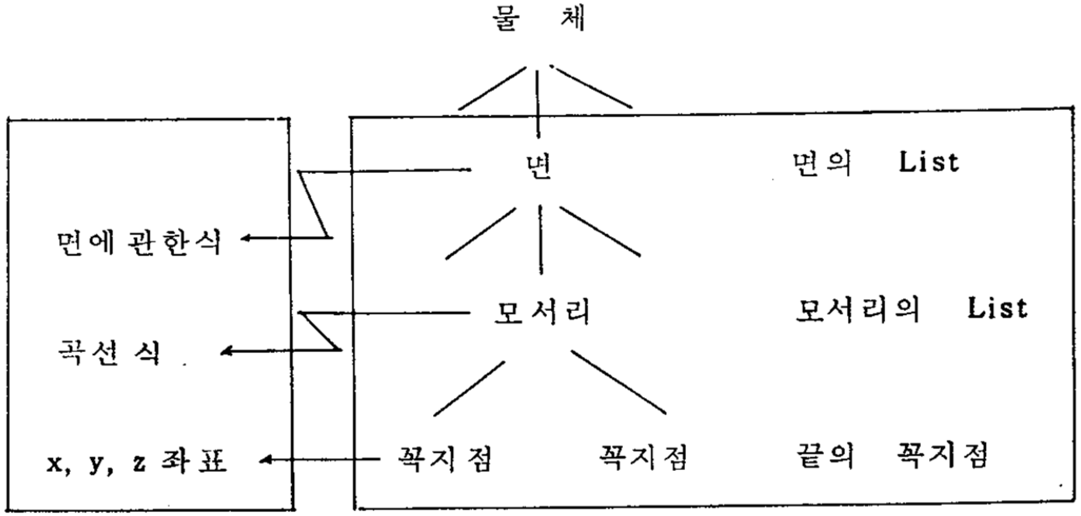
CSG Tree Data Structure

- Advantages:
  - Simple and compact, Easy to manage
  - The solid stored in a CSG tree is always a valid solid
  - Can always be converted to the B-rep
  - Easy to realize parametric modeling
- Disadvantages:
  - Only Boolean operations are allowed
  - Require a lot of computation to derive the B-rep information

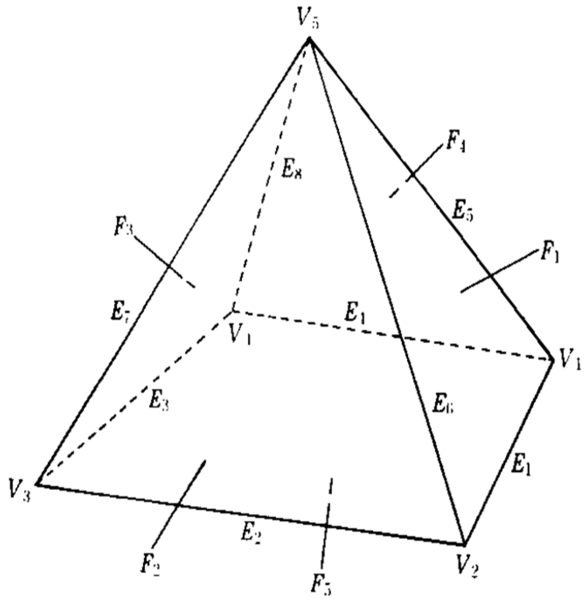


B-Rep Data Structure

- The basic elements composing the boundary of a solid would be the vertices, the edges, and the faces
- B-Rep data structure is a structure storing these entities with the information on how they are interconnected



Simple B-Rep Data Structure



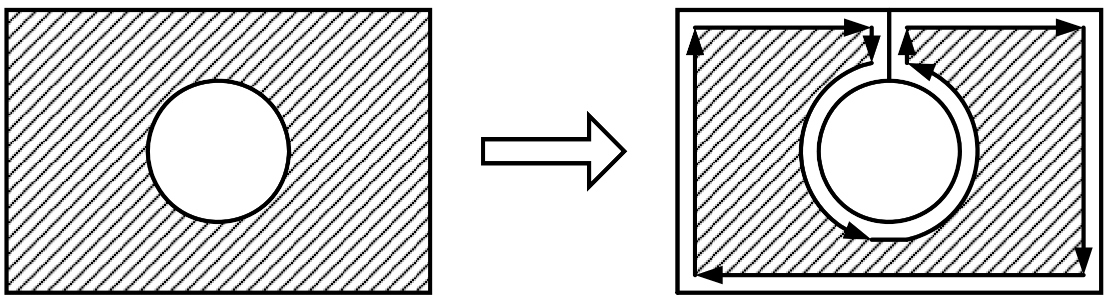
Face table	
Face	Edges
F_1	E_1, E_5, E_6
F_2	E_2, E_6, E_7
F_3	E_3, E_7, E_8
F_4	E_4, E_8, E_5
F_5	E_1, E_2, E_3, E_4

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

Problems of Simple B-Rep (1)

- Designed for only Planar Polyhedra
  - If a solid having the curved faces and the curve edges is to be stored, each row of the face table and the edge table should be modified to include the surface equation and the curve equation respectively
- Not Support a Face with Multiple Boundaries
  - It cannot be stored in the face table because it requires multiple list of edges instead of a single list
  - “bridge edge” Method: Add an edge connecting the external and the internal boundaries to merge two list of edges into one list

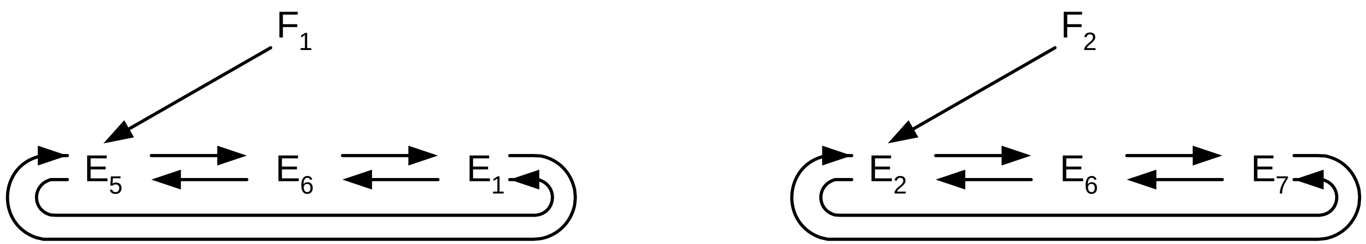


Problems of Simple B-Rep (2)

- Different number of columns for each row of the face table
  - the number of edges for each face is different and varies as the modeling operation proceeds
- Inefficient Search for Connectivity Information
  - searching two faces sharing an edge
  - searching all the edges sharing a vertex

Half Edge Data Structure (1)

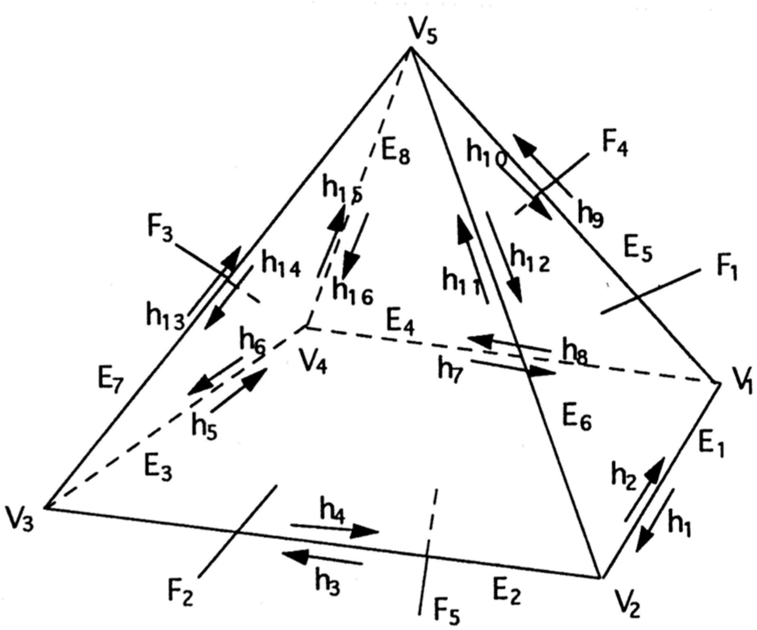
- Introduction of Doubly Linked List
  - As a remedy for the variable size of the face table
  - storing a list of edges for each face in a doubly linked list

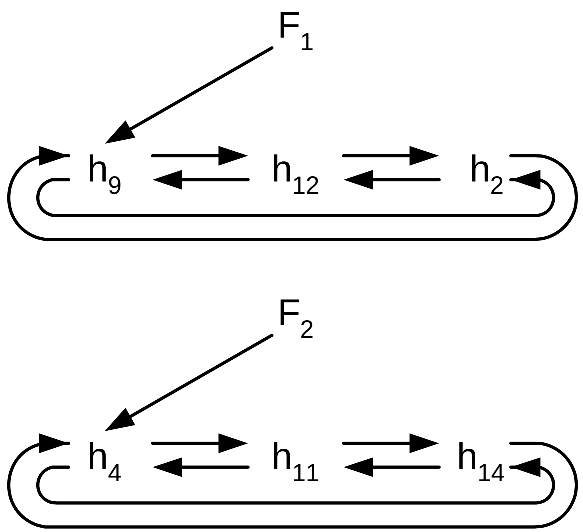


- New Problem:
  - Inconsistency of the previous and the next edge pointers of E_6

Half Edge Data Structure (2)

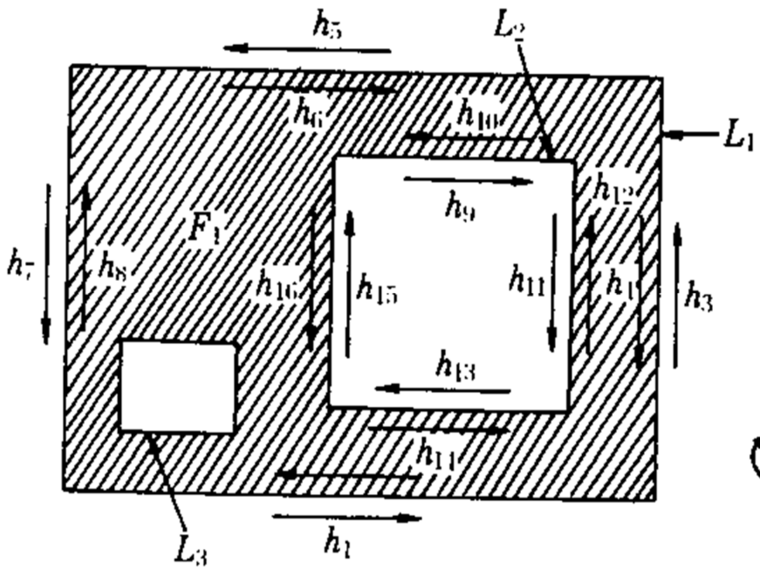
- Introduction of Half Edges
  - splitting each edge into halves and using these halves separately for the two faces sharing the original edge

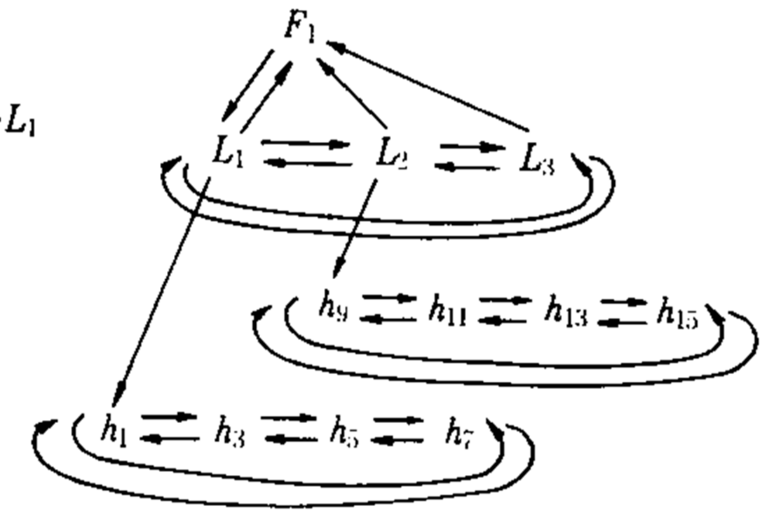




Half Edge Data Structure (3)

- Introduction of Loops
  - To represent the faces with inner holes without adding redundant bridge edges
  - A loop is a list of edges forming a closed circuit and thus any face is bounded by one peripheral loop (CCW) and several hole loops (CW)



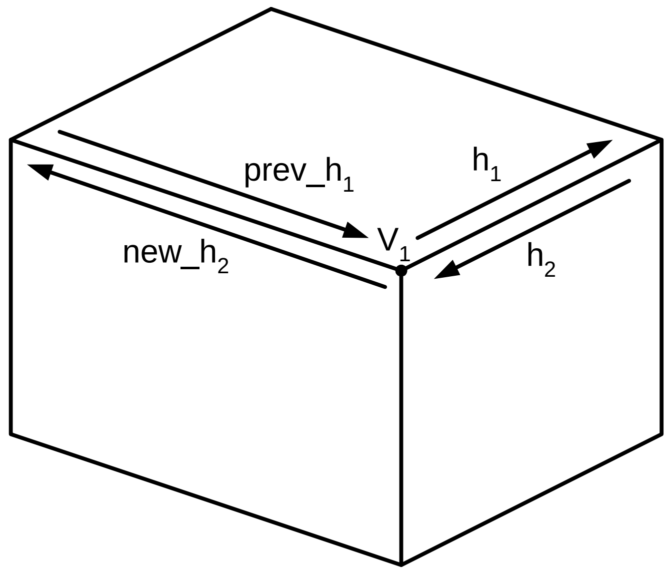


Half Edge Data Structure (4)

```
struct solid
{
	Id	solidno;	/* solid identifier */
	Face	*sfaces;	/* pointer to list of faces */
	Edge	*sedges;	/* pointer to list of edges */
	Vertex	*sverts;	/* pointer to list of vertices */
	Solid	*nexts;	/* pointer to next sold */
	Solid	*prevs;	/* pointer to previous solid */
};
struct face
{
	Id	faceno;	/* face identifier */
	Solid	*fsolid;	/* back pointer to solid */
	Loop	*flout;	/* pointer to outer loop */
	Loop	*floops;	/* pointer to list of loops */
	vector	feq;	/* face equation */
	Face	*nextf;	/* pointer to next face */
	Face	*prevf;	/* pointer to previous face */
};
struct loop
{
	HalfEdge	*ledg;	/* prt to ring of halfedges */
	Face	*lface;	/* back pointer to face */
	Loop	*nextl;	/* pointer to next loop */
	Loop	*prevl;	/* pointer to previous loop */
};
struct edge
{
	HalfEdge	*he1;	/* pointer to right halfedge */
	HalfEdge	*he2;	/* pointer to left halfedge */
	Edge	*nexte;	/* pointer to next edge */
	Edge	*preve;	/* pointer to previous edge */
};
struct halfedge
{
	Edge	*edg;	/* pointer to parent edge */
	Vertex	*vtx;	/* pointer to starting vertex */
	Loop	*wloop;	/* back pointer to loop */
	HalfEdge	*nxt;	/* pointer to next halfedge */
	HalfEdge	*prv;	/* pointer to previous halfedge */
};
struct vertex
{
	Id	vertexno;	/* vertex identifier */
	HalfEdge	*vedge;	/* pointer to a halfedge */
	vector	*vcoord;	/* vertex coordinates */
	Vertex	*nextv;	/* pointer to next vertex */
	Vertex	*prevv;	/* pointer to previous vertex */
};
union nodes
{
	Solid	s;
	Face	f;
	Loop	l;
	HalfEdge	h;
	Vertex	v;
	Edge	e;
};
```

Half Edge Data Structure (5)

- Finding an adjacency information between edges and vertices
  - Search all the edges connected to a vertex, denoted by V_1

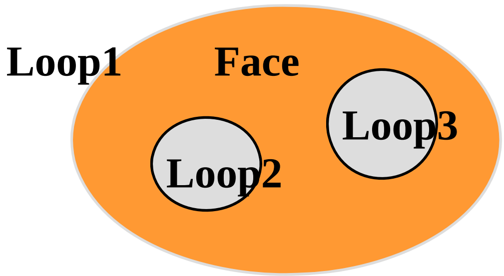


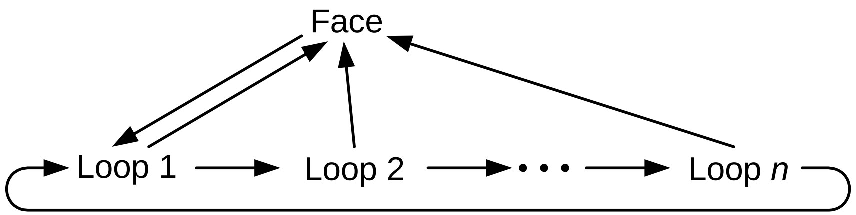
Winged Edge Data Structure (1)

- Introduced by Baumgart in 1974
- Extended by Braid to handle a solid with through holes by introducing the concept of a loop
- Although the face has the major role in describing a solid in Half Edge data structure, the edges play the major role in winged edge data structure
- Connections between vertices, edges, and faces in Winged Edge Structure
  - The list of edges for each face is not explicitly stored
  - The problem of varying number of edges for each face is solved without introducing the linked list, and consequently the half edges

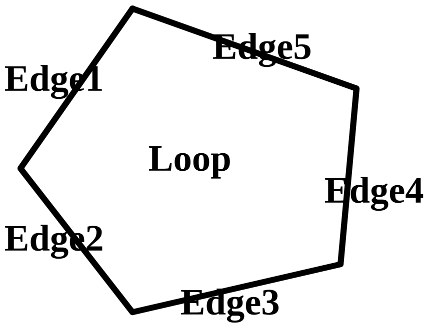
Winged Edge Data Structure (2)

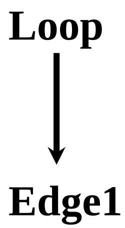
- Connection between a face and its loops





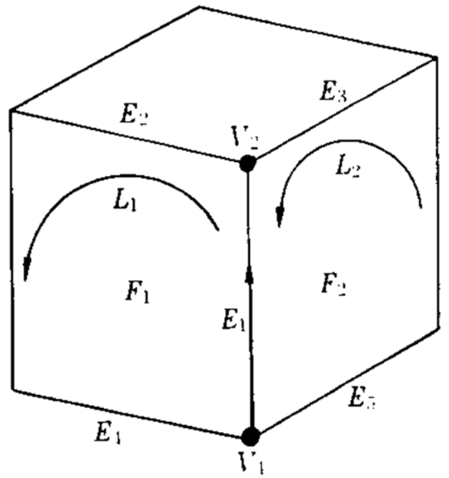
- Connection between a loop and its edges

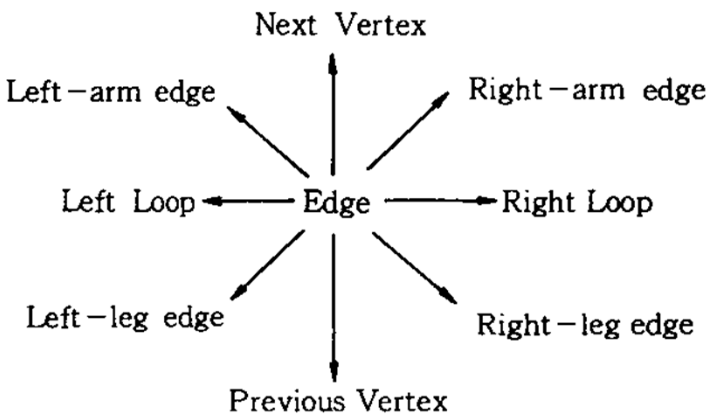




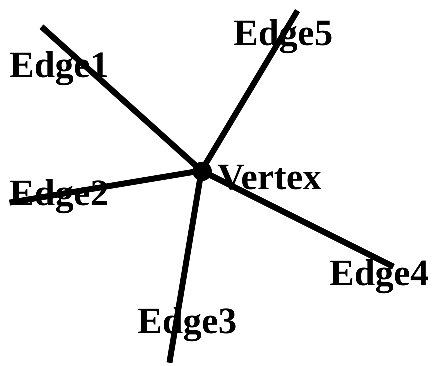
Winged Edge Data Structure (3)

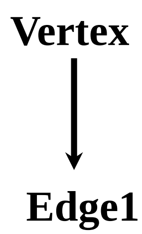
- Edge
  - 하나의 edge에 대해 다음과 같은 8개의 연결 요소를 저장한다



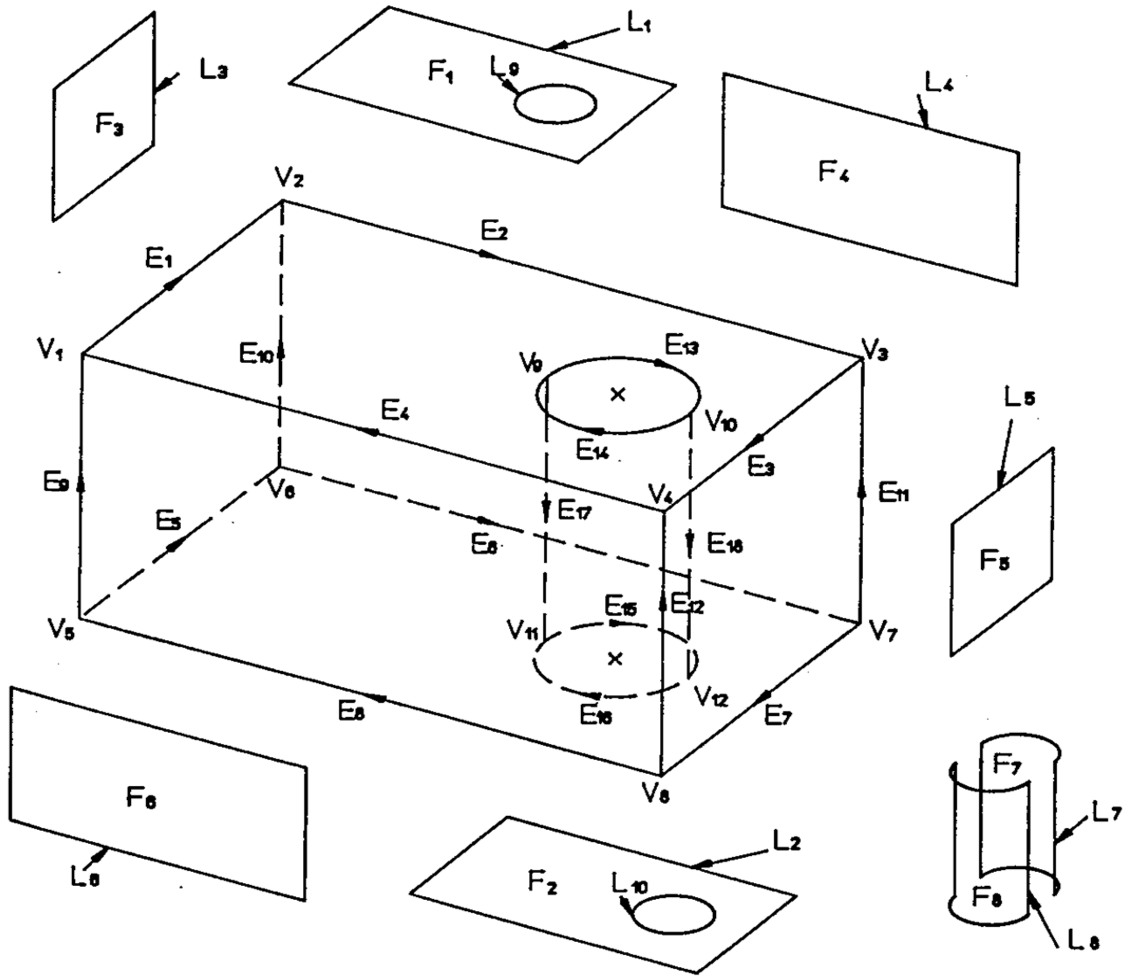


- Connection between a vertex and its edges





Example of Winged Edge Data Structure (1)



Example of Winged Edge Data Structure (2)

FACE

	Loop	*Type*	*Geometric Pointer*
F_1	1	1	2
F_2	2	1	1
F_3	3	1	3
F_4	4	1	4
F_5	5	1	5
F_6	6	1	6
F_7	7	2	1
F_8	8	2	1

LOOP

	Next Loop	*Face*	*Edge*	*Type*
L_1	9	1	1	0=peripheral
L_2	10	2	8	0
L_3	0	3	9	0
L_4	0	4	2	0
L_5	0	5	11	0
L_6	0	6	9	0
L_7	0	7	18	0
L_8	0	8	17	0
L_9	1	1	13	1=hole
L_{10}	2	2	15	1

EDGE

	Previous Vertex	*Next Vertex*	*Left Loop*	*Right Loop*	*Left arm*	*Left leg*	*Right leg*	*Right arm*	*Type*	*Geometric Pointer*
E_1	1	2	3	1	10	9	4	2	1	.
E_2	2	3	4	1	11	10	1	3	1	.
E_3	3	4	5	1	12	11	2	4	1	.
E_4	4	1	6	1	9	12	3	1	1	.
E_5	5	6	2	3	6	8	9	10	1	.
E_6	6	7	2	4	7	5	10	11	1	.
E_7	7	8	2	5	8	6	11	12	1	.
E_8	8	5	2	6	5	7	12	9	1	.
E_9	5	1	3	6	1	5	8	4	1	.
E_{10}	6	2	4	3	2	6	5	1	1	.
E_{11}	7	3	5	4	3	7	6	2	1	.
E_{12}	8	4	6	5	4	8	7	3	1	.
E_{13}	9	10	9	7	14	14	17	18	2	1
E_{14}	10	9	9	8	13	13	18	17	2	1
E_{15}	11	12	7	10	18	17	16	16	2	2
E_{16}	12	11	8	10	17	18	15	15	2	2
E_{17}	9	11	7	8	15	13	14	16	1	.
E_{18}	10	12	8	7	16	14	13	15	1	.

VERTEX

	Edge	*x,y,z*
V_1	1	1
V_2	2	2
V_3	3	3
V_4	4	4
V_5	5	5
V_6	6	6
V_7	7	7
V_8	8	8
V_9	13	9
V_{10}	14	10
V_{11}	15	11
V_{12}	16	12

B-Rep(Boundary Representation)

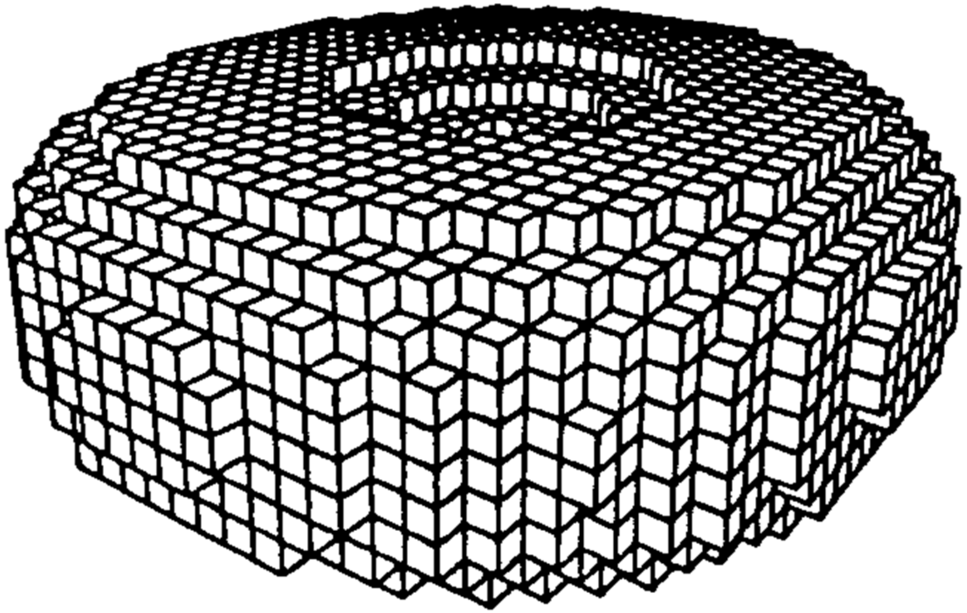
- Advantages:
  - Quick response (for display...) as the result of boundary evaluation is stored
  - Provide topology information immediately
  - Support various modeling functions including the Boolean operations
- Disadvantages:
  - Complex data structure and a lot of data storage requirement
  - Not easy to realize the parametric modeling technique
  - Invalid solid models may happen

Decomposition Model Structure

- Approximate description of a solid model as an aggregate of simple solids
- Many possible decomposition models depending upon the selection of the simple solid and the method of aggregation.
- Typical Decomposition Models
  - Voxel representation
  - Octree representation
  - Cell representation

Voxel Representation (1)

- A three dimensional extension of the raster representation



Voxel Representation (2)

- Advantages
  - A solid of arbitrary shape can be described always accurately or approximately at least (human bones and organs
  - Easy to calculate the mass properties of a solid, such as mass and moments of inertia
  - Easy to obtain the result of the Boolean operation between two solids
  - Able to represent the space excluding the solid
    - useful to calculate the trajectory of robots avoiding the obstacles

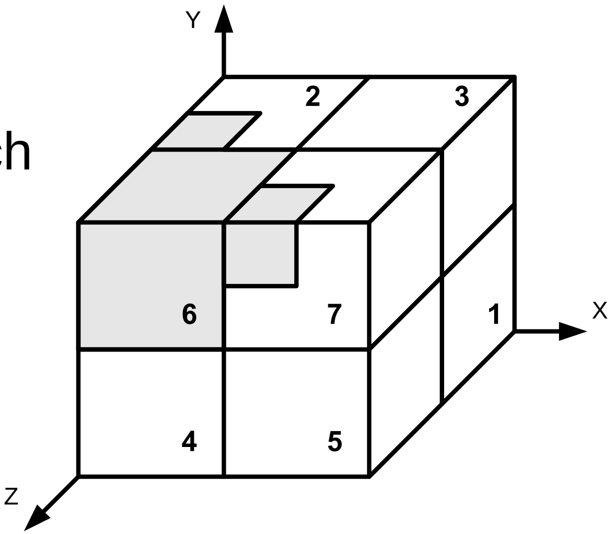
Voxel Representation (3)

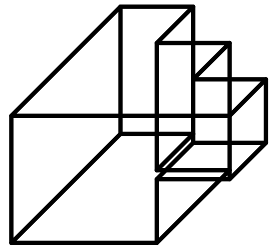
- Drawbacks
  – The memory space required to store the voxel representation increase dramatically as the size of the voxels decrease
  – The voxel representation is inherently an approximation of the original solid

- Usage of Voxel Representation
  – Not used as a mathematical representation of a solid
  – Often used as an auxiliary representation to increase the computational efficiency

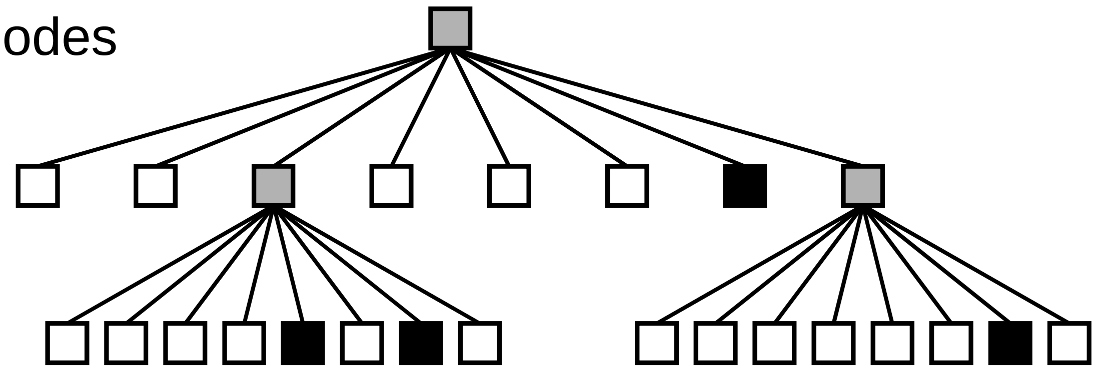
Octree Representation

- The original cube or is divided into eight cubes, which is called *Octants*, each time



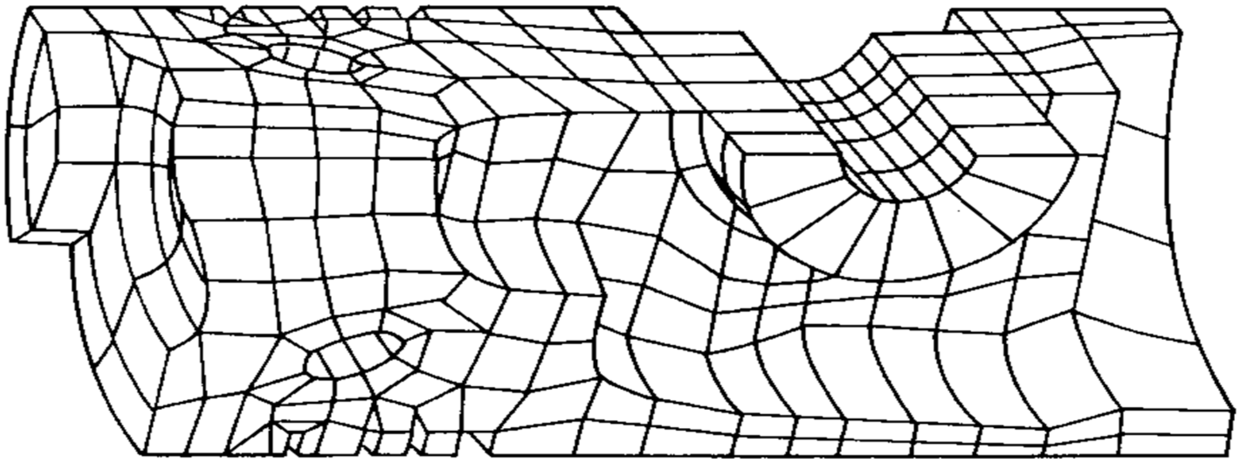


- *Octree*: a tree whose nodes represent the octants



Cell Decomposition Representation

- Describes a solid as an aggregate of simple cells with arbitrary shape
- Finite elements is a typical example



Topological Operators

- Topological operations manipulating the data in the B-Rep data structure are necessary
- The topology entities stored in B-Rep data structures are *shell, face, loop, edge*, and *vertex*.
- Two possible choices:
  1. The topological operators each of which manipulates these entities separately
     - e.g. an operator to make an edge, an operator to delete a vertex, etc
  2. The topological operators that manipulates these entities in a small group

Problems of The First Type of Topological Operators

- The Operators Manipulating Topology Entities Separately

1. Invalid Solid

  - There is an inherent contradiction of trying to handle independently each topology entity which is not independent of each other.
  - The relation called the Euler-Poincare formula exists among the numbers of the topology entities.

2. Inefficiency

  - An addition or a deletion of a topology entity accompanies the change in other topology entities in most cases.

Euler-Poincare formula

$$v - e + f - h = 2\,(s - p)$$

v : no. of vertices

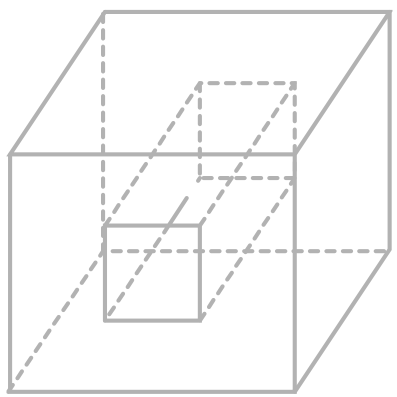
e : no. of edges

f : no. of faces or the peripheral loops

h : no. of hole loops (l – f)

p : no. of passages (through hole)

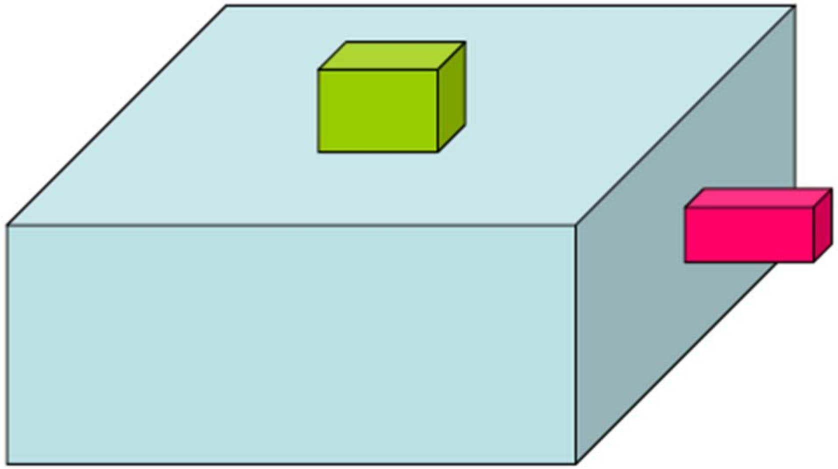
s : no. of shells (internal void of a solid ≥ 1)

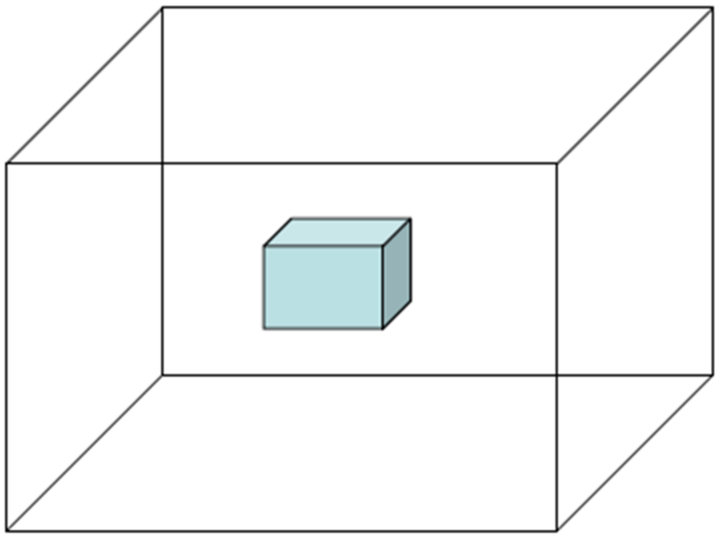


v=16, f=10, h=2, s=1, and p=1

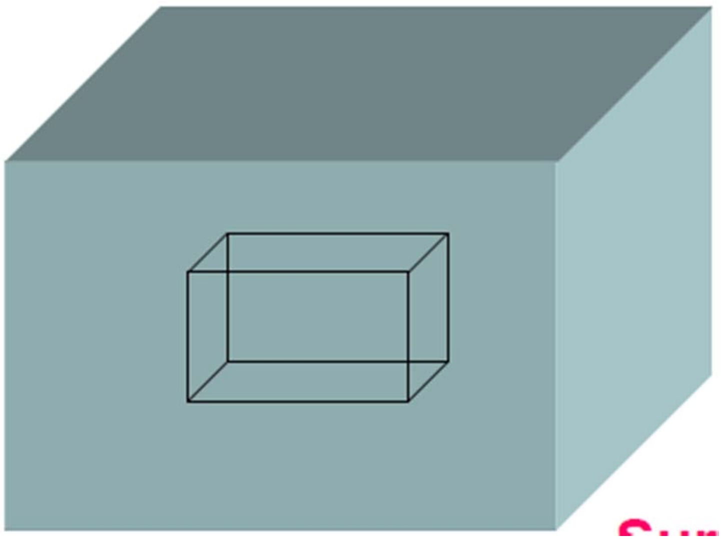
16 - 24 + 10 - 2 = 2 (1 - 1)

Example





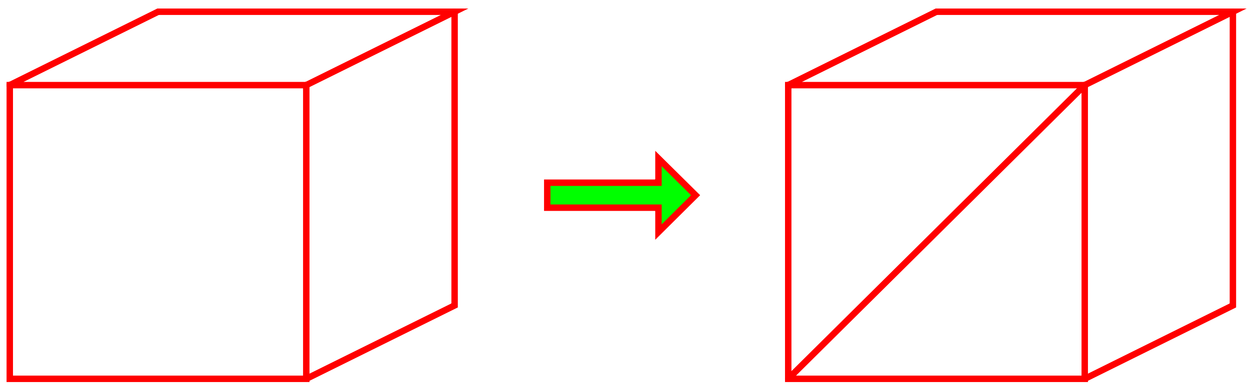
interior hole



surface hole

Dependency between Topology Entities

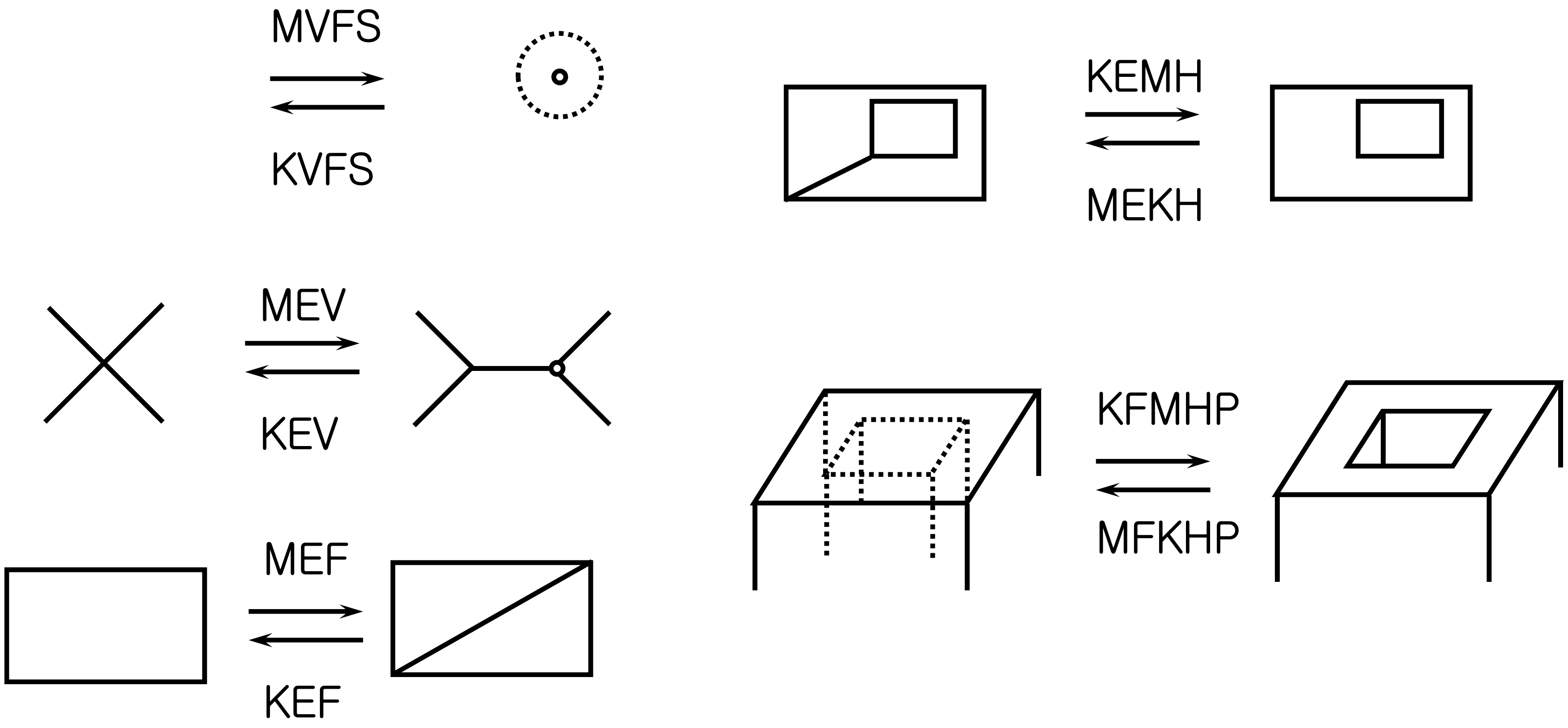
- An addition or a deletion of a topology entity accompanies the change in other topology entities in most cases.
- Example:
  - Change in a face caused by an addition of an edge



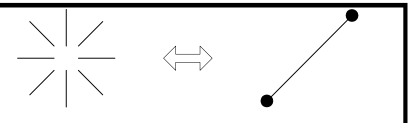
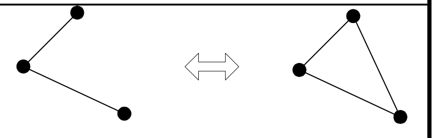
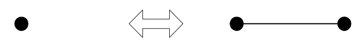
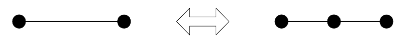
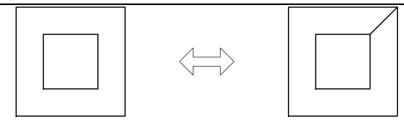
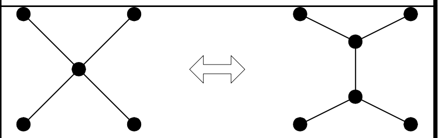
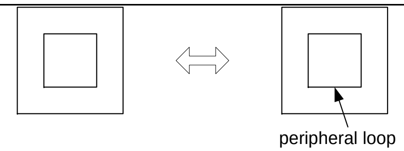
Euler Operators

- The operators that manipulate the topology entities in a small group with satisfying the Euler-Poincare formula.
- The valid solid can be guaranteed after the topology change
- More than five sets of the operators are required
  - As there are five independent topology entities in the Euler-Poincare formula
  - $v - e + f - h = 2(s - p)$
  - $v - e + f - (l - f) = 2(s - p)$
  - $v - e + 2f - l = 2(s - p)$

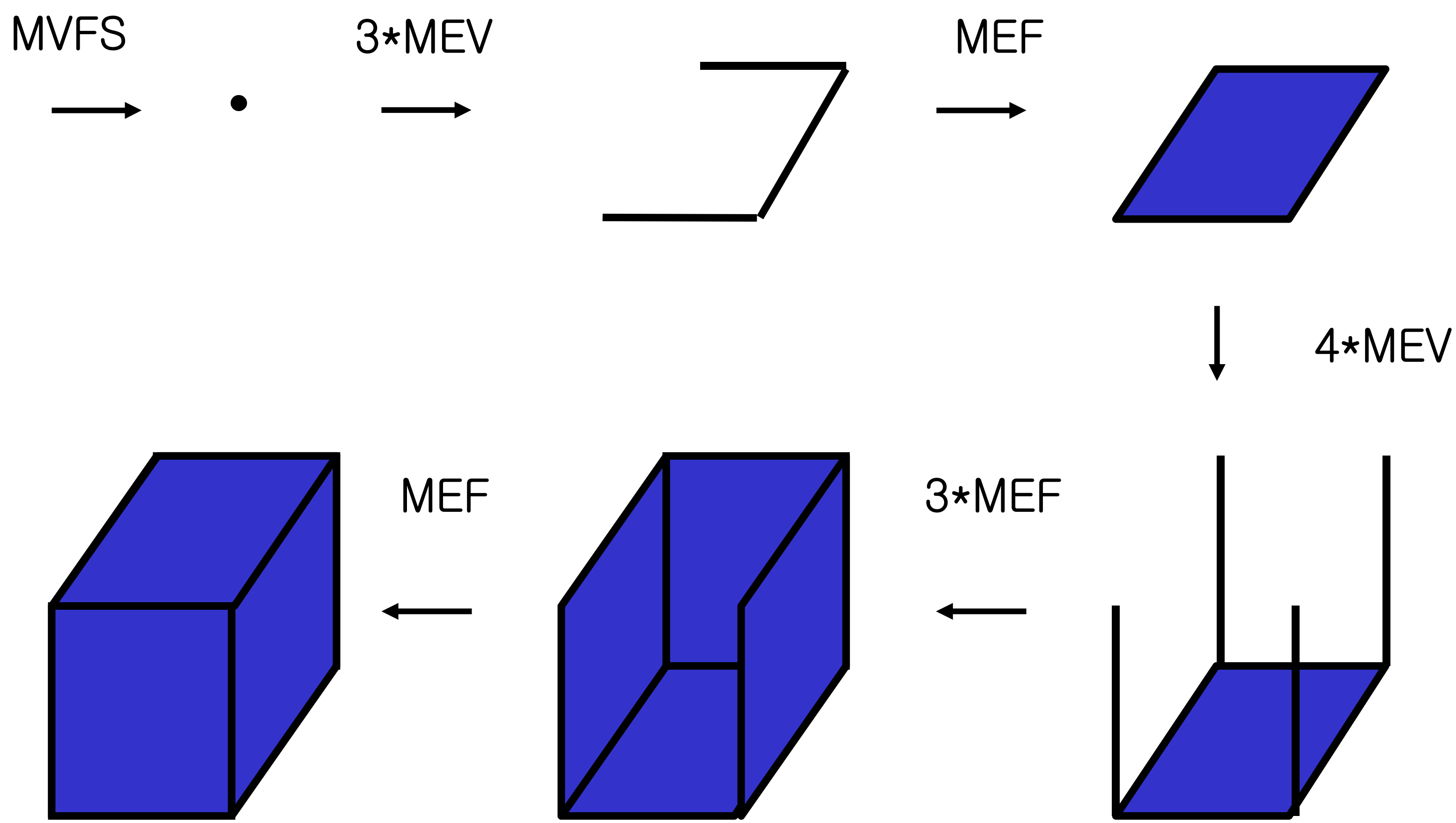
Euler Operations



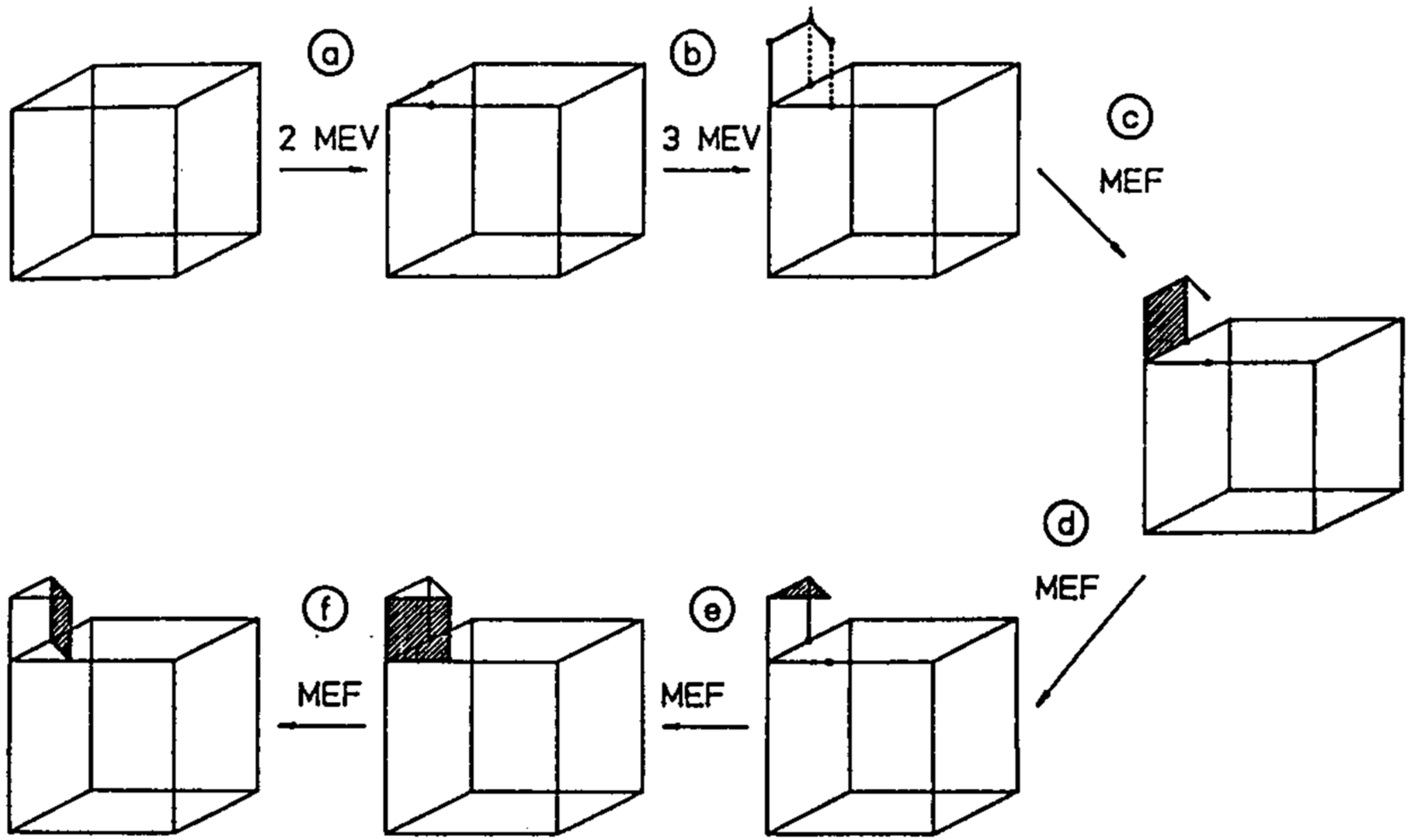
Euler Operations

MEVVLS (KEVVLS)	make(kill) edge, two vertices, loop, shell	
MEL (KEL)	make(kill) edge, loop	
MEV (KEV)	make(kill) edge, vertex	
MVE (KVE)	make(kill) vertex, edge	
MEKH (KEMH)	make(kill) edge, kill(make) hole	
MZEV (KZEV)	make(kill) zero length edge, vertex	
MPKH (KPMH)	make(kill) peripheral loop, kill(make) hole loop	

Example of Euler Operators

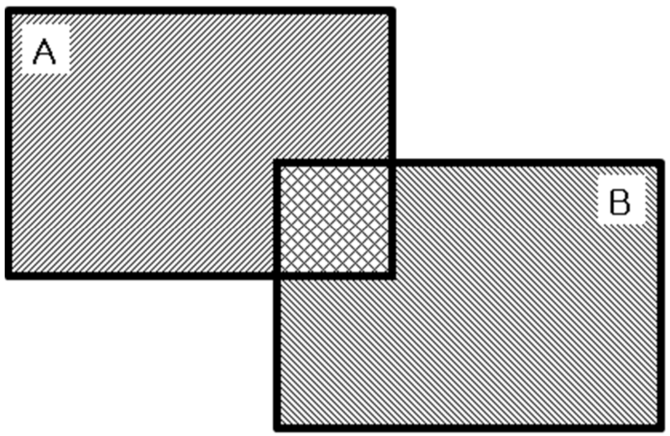


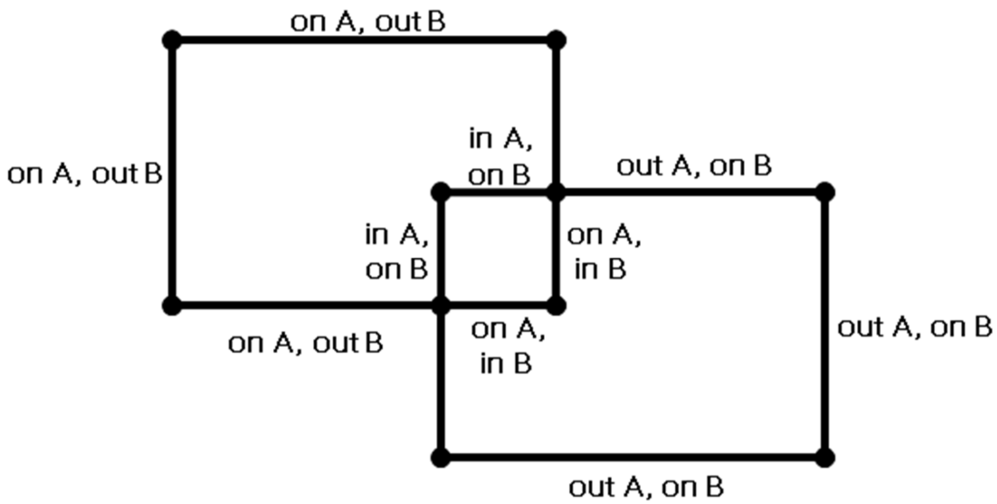
Example of Euler Operators



Boolean Operations

- Intersect faces and collect all the edges
- Classify edges: in/on/out of A and B
- Collect the proper edges
  - Based on their relative location and the specific Boolean operation
  - “A union B” = collect all edges except “inA” or “inB” edges
  - “A intersection B”= collect all edges except “outA” or “outB” edges





Formulae for Volume Properties

Volume: $V = \iiint_V dV$

Centroid:

$$x_c = \frac{1}{V}\iiint_V x\,dV$$

$$y_c = \frac{1}{V}\iiint_V y\,dV$$

$$z_c = \frac{1}{V}\iiint_V z\,dV$$

Moments of Inertia:

$$I_{xx} = \iiint_V (y^2 + z^2)\,dV$$

$$I_{yy} = \iiint_V (x^2 + z^2)\,dV$$

$$I_{zz} = \iiint_V (y^2 + x^2)\,dV$$

Products of Inertia:

$$I_{xy} = \iiint_V xy\,dV$$

$$I_{yz} = \iiint_V yz\,dV$$

$$I_{zx} = \iiint_V zx\,dV$$

- Voxel or octree representation: coordinates → sum
- CSG: adding or subtracting the volume integrals of primitives
- B-Rep: not so simple

Calculation of Volumetric Properties (1)

- Step 1: Convert the volume integral to the surface integral by Gauss's theorem

$$\psi = \iiint_V F(x,y,z)\,dV = \iiint_V \left(\nabla \bullet \Phi\right)\,dV = \iint_s \left(\Phi \bullet \mathbf{n}\right)\,ds = \iint_s G(x,y,z)\,ds$$

- Step 2: Surface integral is obtained by summing the surface integral over each face because $S = \Sigma S_i$

$$\iint_S G(x,y,z)\,ds = \sum_{i=1}^{n_f} \psi_i \quad \text{where} \begin{cases} \psi_i = \iint_{S_i} G(x,y,z)\,ds \\ n_f : \text{ number of faces} \end{cases}$$

Calculation of Volumetric Properties (2)

- Step 3: Convert each surface integral to a double integral over the domain of the parameters defining S_i

$$\text{face } S_i \rightarrow \mathbf{P}(u,v) = x(u,v)\mathbf{i} + y(u,v)\mathbf{j} + z(u,v)\mathbf{k}$$

$$\psi_i = \iint_{R_i} G[x(u,v), y(u,v), z(u,v)] \left|\mathbf{J}\right| dudv = \iint_{R_i} H\left(u,v\right) dudv$$

$$\text{where } \left|\mathbf{J}\right| = \left| \frac{\partial \mathbf{P}(u,v)}{\partial u} \times \frac{\partial \mathbf{P}(u,v)}{\partial v} \right|$$

If the domain R_i is a square represented by $0 \le u \le 1$ and $0 \le v \le 1$,

$$\psi_i = \iint_{R_i} H\left(u,v\right) dudv = \sum_{i=1}^{m} \sum_{j=1}^{n} w_i w_j H\left(u_i, v_j\right)$$

Gaussian Quadrature

- Sample parameter values and corresponding weights for Gaussian quadrature

i, j	wi	ui, vj
	n , m = 2	
1	0.500 000 000	0.211 324 865
2	0.500 000 000	0.788 675 135
	n , m = 3	
1	0.277 777 778	0.112 701 665
2	0.444 444 444	0.500 000 000
3	0.277 777 778	0.887 298 335
	n , m = 4	
1	0.173 927 423	0.069 431 844
2	0.326 072 577	0.330 009 478
3	0.326 072 577	0.669 990 522
4	0.173 927 423	0.930 568 156
	n, m = 5	
1	0.118 463 443	0.046 910 077
2	0.239 314 335	0.230 765 345
3	0.284 444 444	0.500 000 000
4	0.239 314 335	0.769 234 655
5	0.118 463 443	0.953 089 923
	n, m = 6	
1	0.085 662 246	0.033 765 243
2	0.180 380 787	0.169 395 307
3	0.233 956 967	0.380 690 407
4	0.233 956 967	0.619 309 593
5	0.180 380 787	0.830 604 693
6	0.085 662 246	0.966 234 757

Calculation of Volumetric Properties (3)

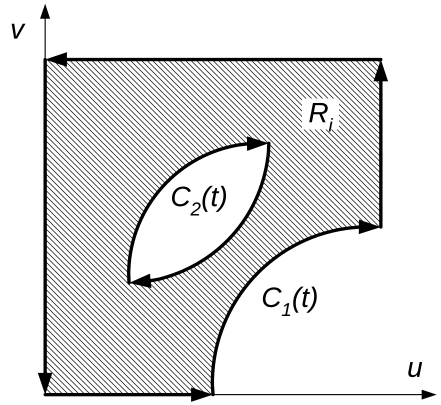
- Step 4: (not mapped to a square in its parametric domain) Convert each double integral to the line integral by Green's theorem
  - As double integrals over irregular domains cannot be evaluated by Gaussian quadrature

$$\left[\iint_{R_i} H(u,v)\,du\,dv = \right] \iint_{R_i} \left[\frac{\partial \alpha(u,v)}{\partial u} - \frac{\partial \beta(u,v)}{\partial v} \right] du\,dv = \oint \left[\beta(u,v)du + \alpha(u,v)dv \right]$$

$$H(u,v) = \frac{\partial \alpha(u,v)}{\partial u} - \frac{\partial \beta(u,v)}{\partial v} \xrightarrow{\beta(u,v)=0} H(u,v) = \frac{\partial \alpha(u,v)}{\partial u}$$

$$H(u,v) = \sum_{i=0}^{M} \sum_{j=0}^{M} a_{ij} u^i v^j \rightarrow \alpha(u,v) = \sum_{i=0}^{M} \sum_{j=0}^{M} \frac{1}{i+1} a_{ij} u^{i+1} v^j$$

$$\psi_i = \iint_{R_i} H(u,v)\,du\,dv = \oint \alpha(u,v)\,dv$$



Calculation of Volumetric Properties (4)

- Step 5: Line integral along the closed boundary is evaluated by summing the line integral along each curve segments

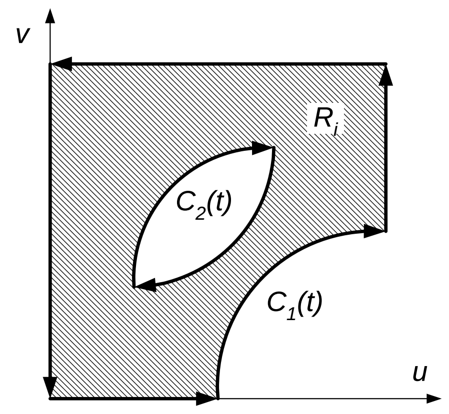
$$\psi_i = \iint_{R_i} H(u,v)\,du\,dv = \oint \alpha(u,v)\,dv = \sum_{l=1}^{L} \int_l \alpha(u,v)\,dv$$

summation for all the curve segments on the boundary

any curve segment along the boundary in the uv domain:

$$u = u(t),\ v = v(t),\ 0 \le t \le 1$$

$$\psi_i = \sum_{l=1}^{L} \int_0^l \alpha\left[u(t), v(t)\right] \frac{dv}{dt}\,dt$$

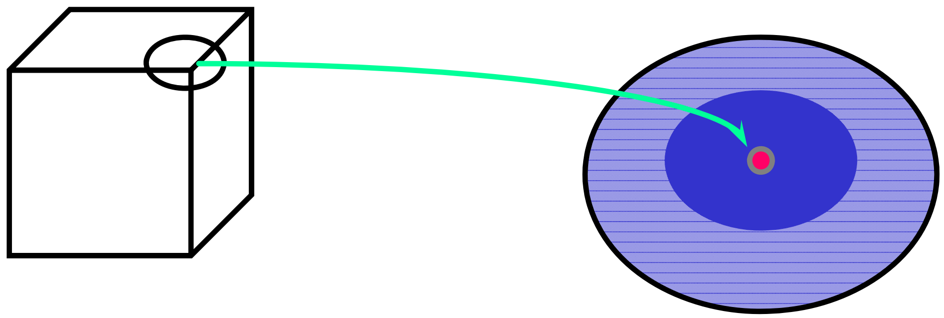


Nonmanifold Modeling Systems

- Allow the unified representation of wireframe, surface, solid and cellular models simultaneously in the same modeling environment
  - Uncompleted geometric models at intermediate design stages
  - Abstract models of mixed dimensions for FEM analysis
  - Cellular structure for FEM solid meshes
  - Isolated vertices, Dangling edges, Laminar faces, Screen faces

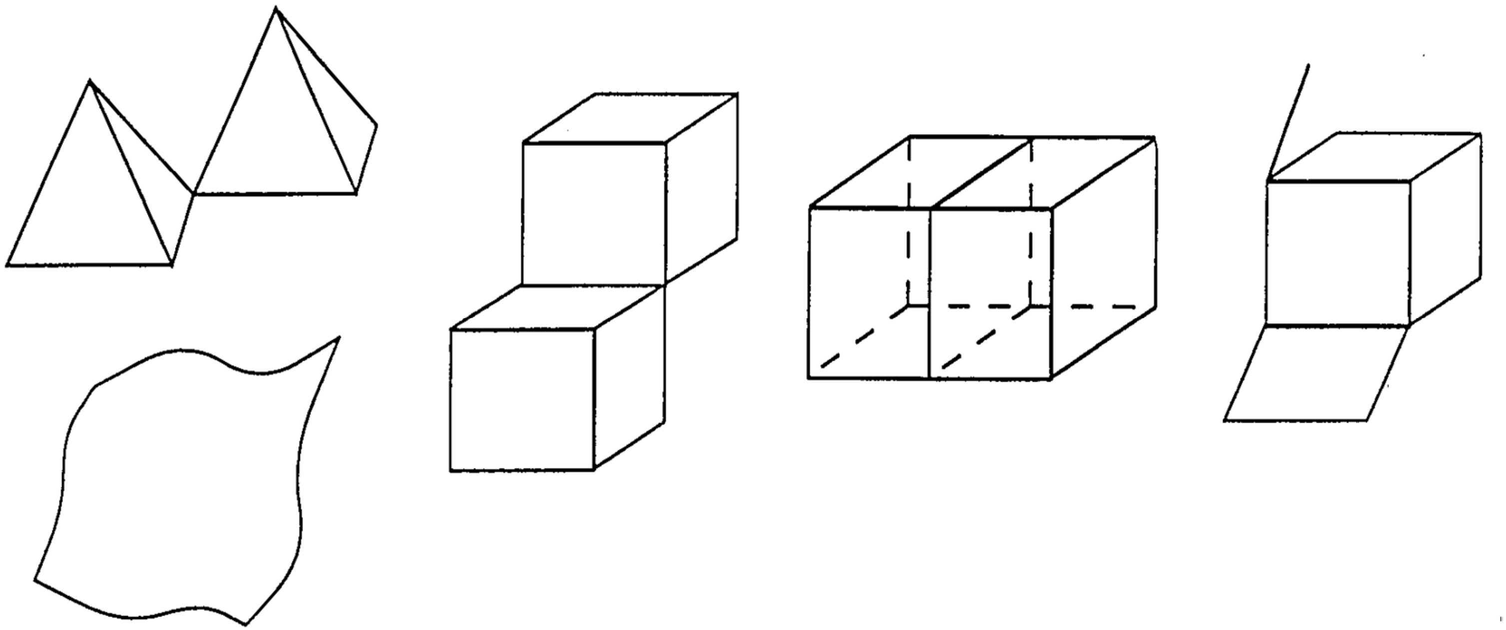
(Two-)Manifold Models

- Manifold == Two-Manifold
- Two Manifold = Every point has a neighborhood that homeomorphic to a two-dimensional disk
- Used in B-Rep-based solid modeling systems



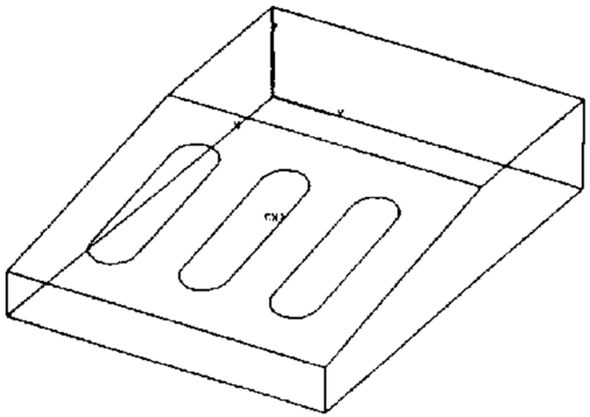
Nonmanifold Models

- Nonmanifold == Non Two-Manifold

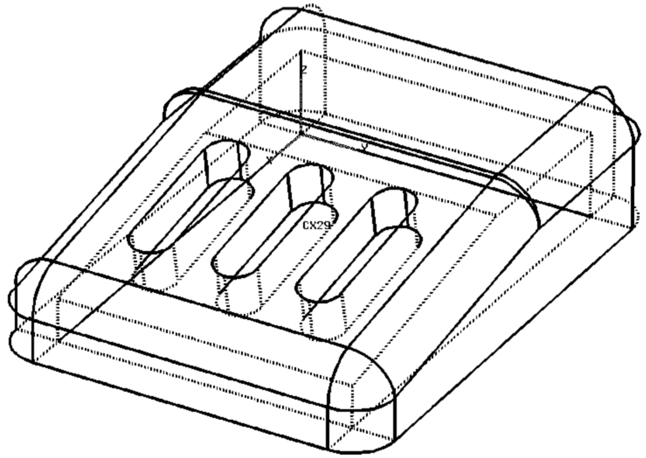


Conversion of Sheet into Solid

Sheet Model

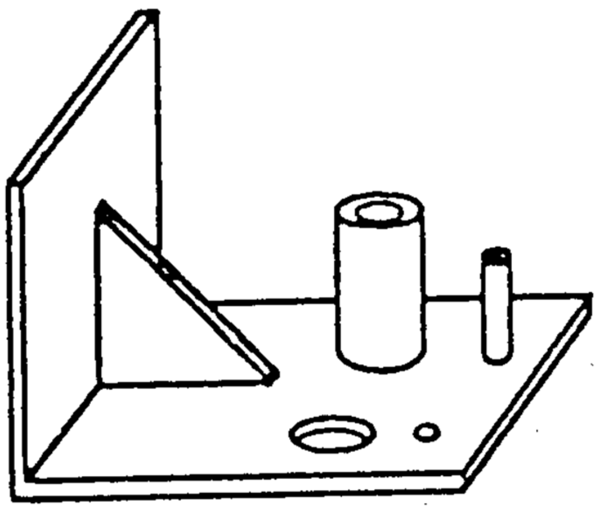


Solid Model

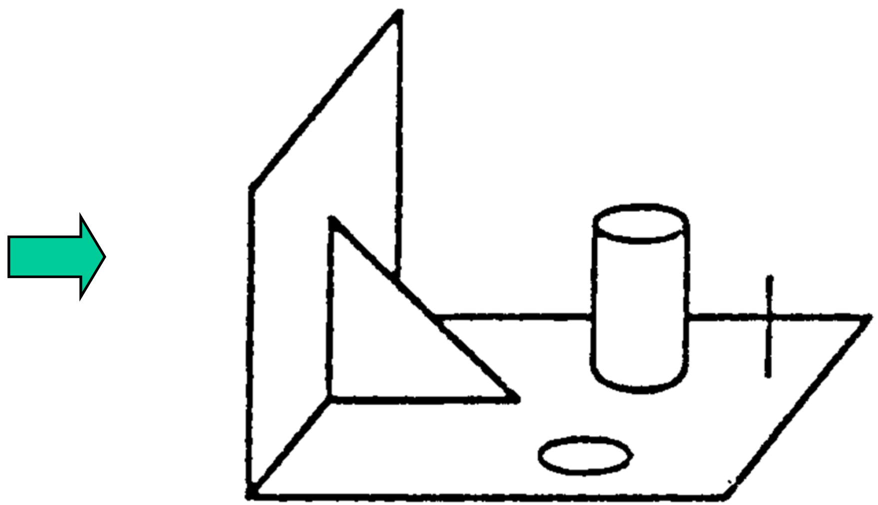


Abstract Models for Analysis

Design Model



Analysis Model



Nonmanifold Modeling Systems

- Advantages:
  - Wide range of representation with a single data structure
    - Wireframes, Surfaces, Solids, Cellular Models, and their Mixtures
    - Support most of the geometric models for design, analysis and manufacturing
- Disadvantages:
  - Data structure is complicated and difficult to manipulate
  - Large amount of data storage
  - More calculation in Boolean operations than solid modelers
- Products:
  - ACIS, I-DEAS Master Series, SHAPES, AnyShape (KMU)