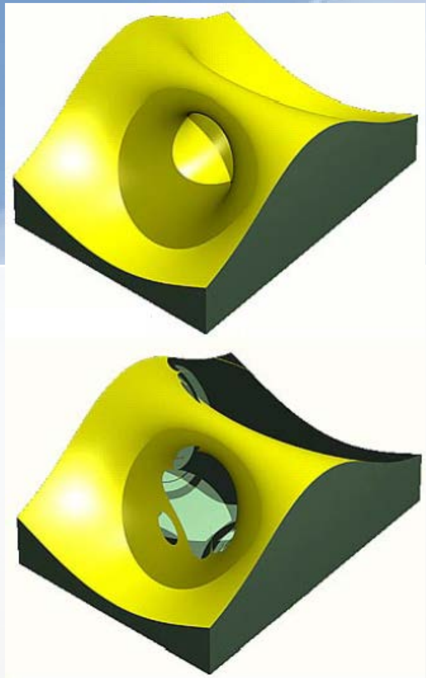
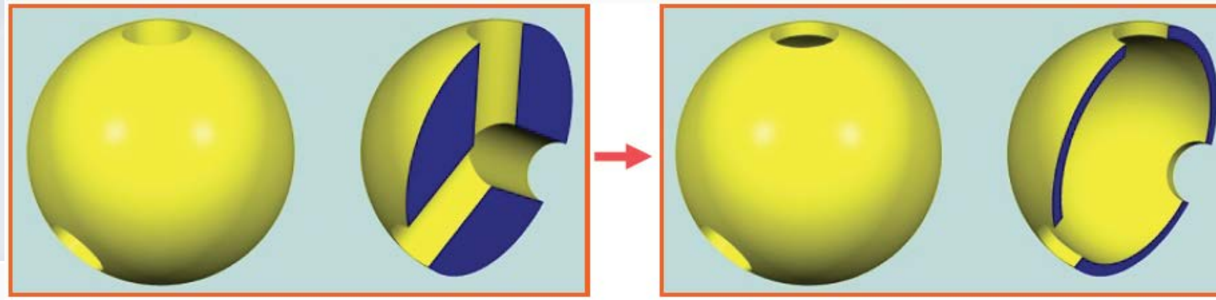
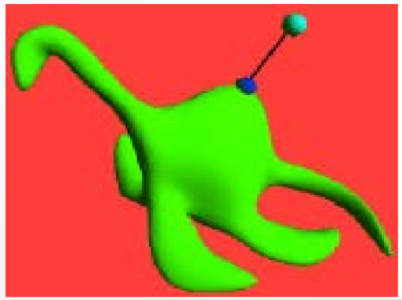
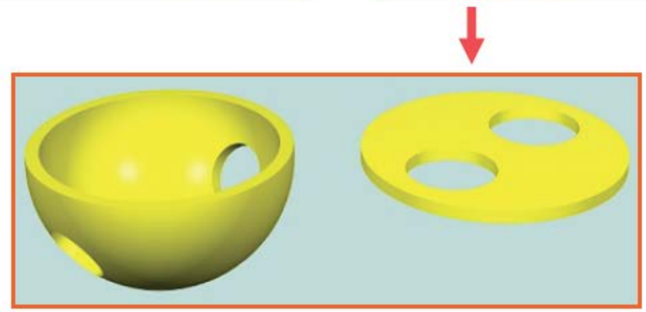
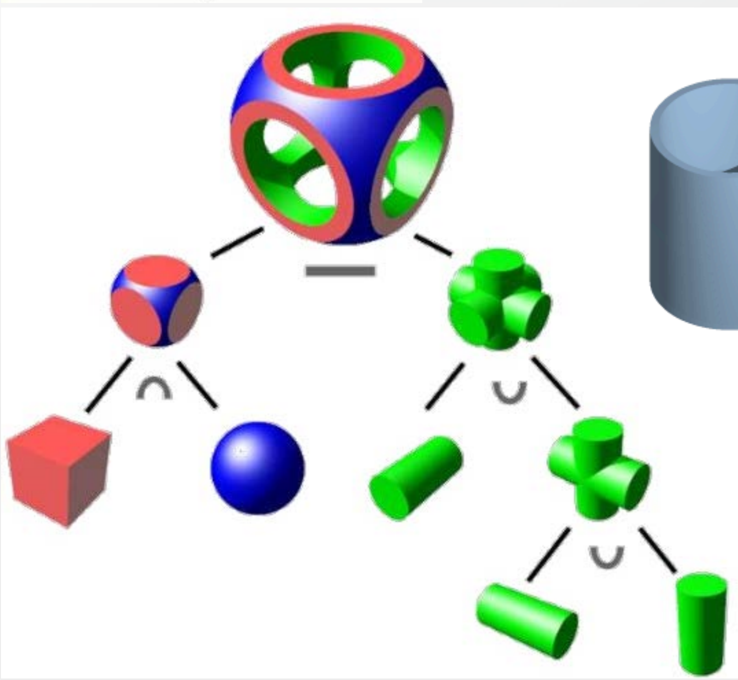





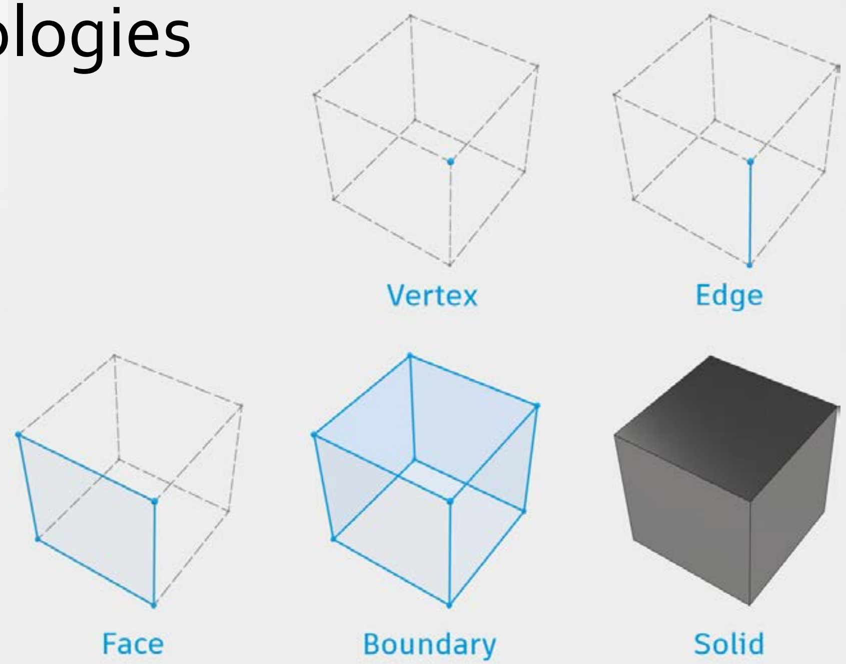
Advanced CAD
Solids

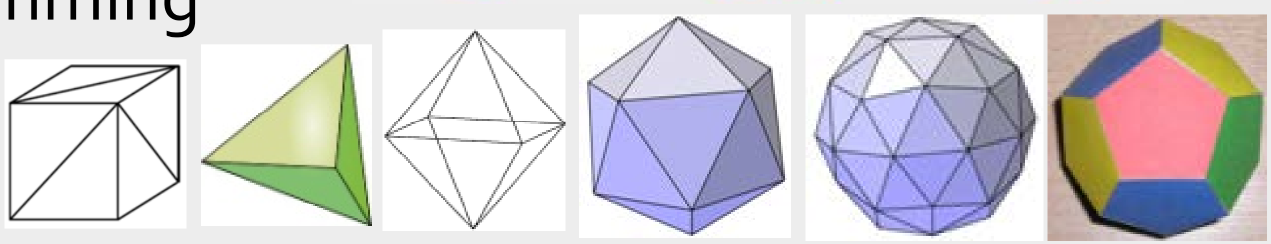


Hikmet Kocabas
Prof. PhD.
Istanbul Technical University

Lectures, Outline of the course

1 Advanced CAD Technologies
2 Geometric Modeling
3 Transformations
4 Parametric Curves
5 Splines, NURBS
6 Parametric Surfaces
7 Solid Modeling
8 API programming





Textbooks

- Computer Aided Engineering Design, Anupam Saxena, Birendra Sahay, Springer, 2005
- CAD/CAM Theory and Practice , Ibrahim Zeid, McGraw Hill , 1991, Mastering CAD/CAM, ed. 2004
- The NURBS Book, Les Piegl, Springer-Verlag, 1997
- Solid Modeling with DESIGNBASE, Hiroaki Chiyokura, Addison-Wesley Pub., 1988
- 3D CAD Principles and Applications, H Toriya, Springer-Verlag, 1991

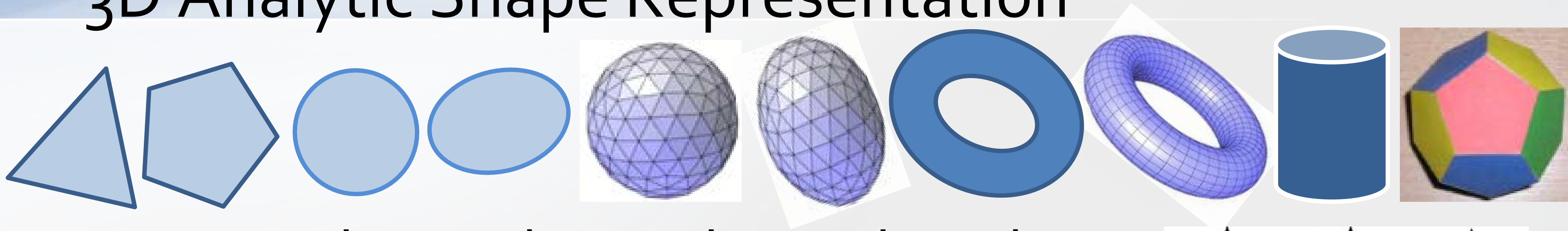
References

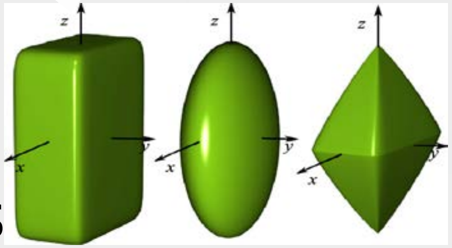
- *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, Fourth Edition*, Gerald Farin, September 1996

- *Geometric Modeling*, Second Edition, Michael E. Mortenson, John Wiley & Sons, January 1997

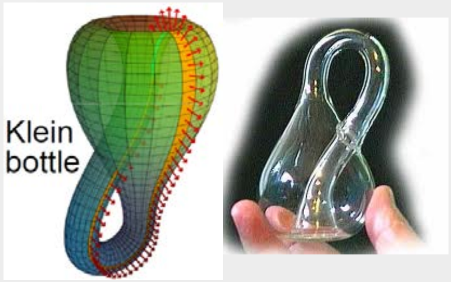
- *Mathematical Elements for Computer Graphics*, Rogers, D.F., Adams, J.A., McGraw Hill, 1990.

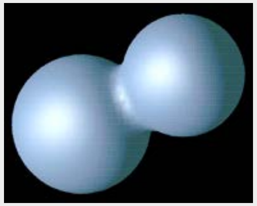
3D Analytic Shape Representation

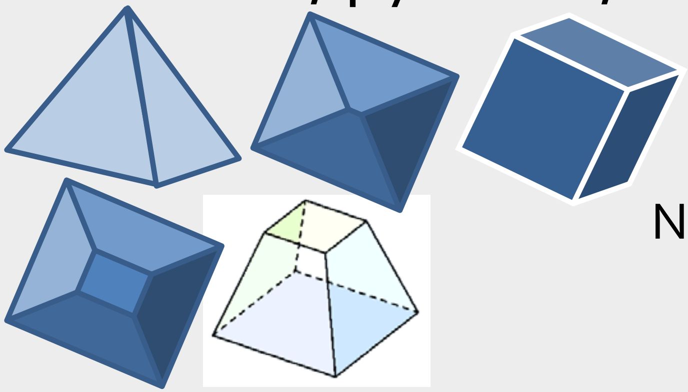


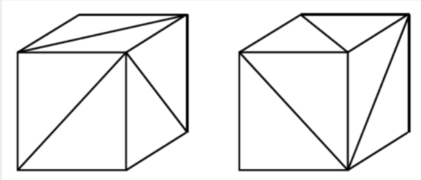


- Triangular meshes, polygonal meshes
- Analytic (commonly-used) shape
- Quadric surfaces, sphere, ellipsoid, torus
- Superquadric surfaces, superellipse, superellipsoid
- Blobby models, tetrahedron, pyramid, hexahedron



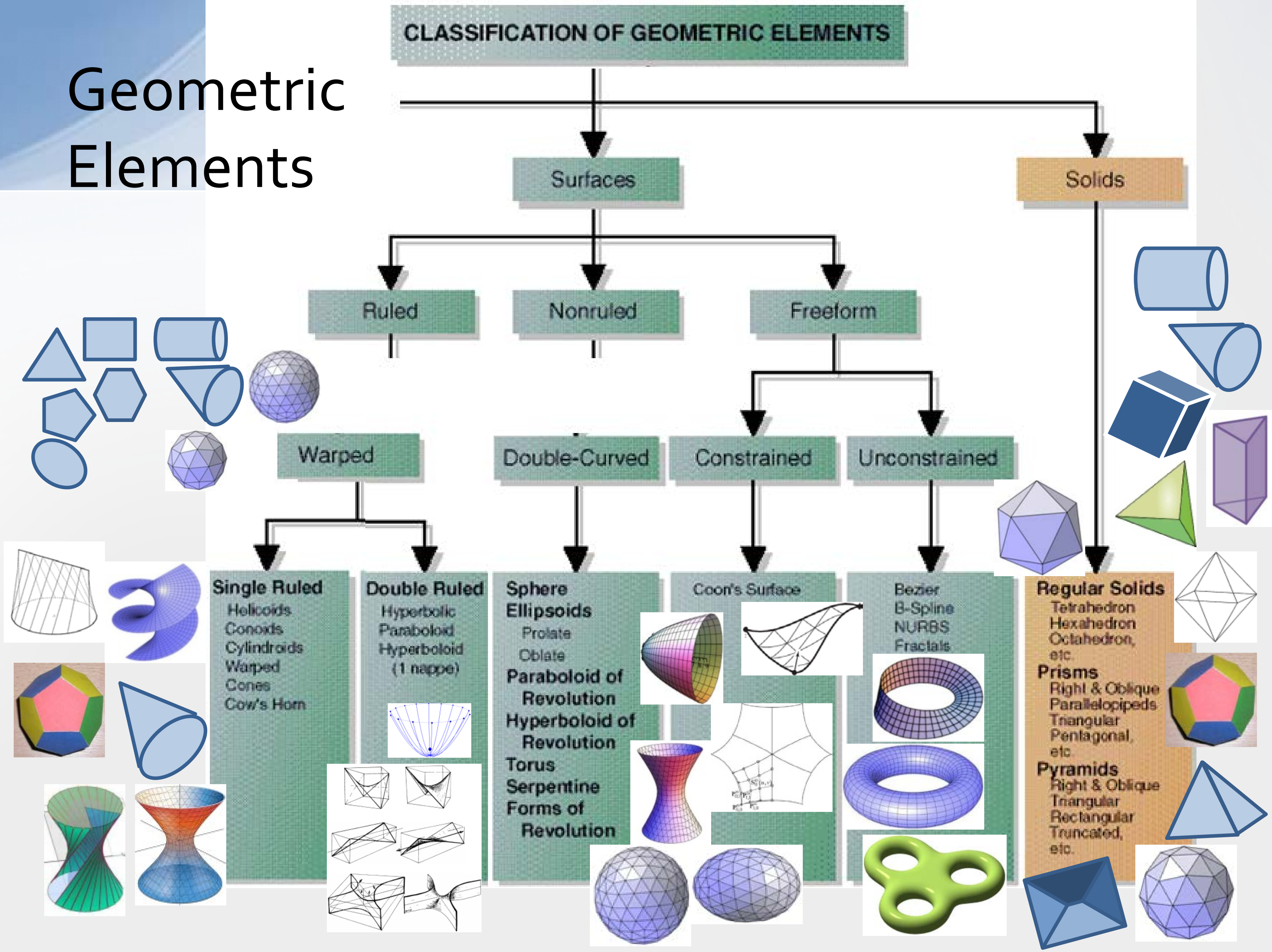






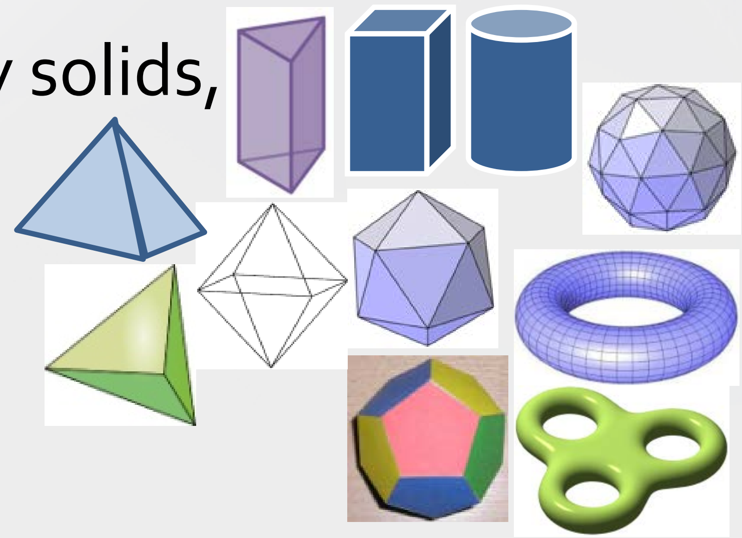
Non-unique model

Geometric Elements



Geometric Modeling

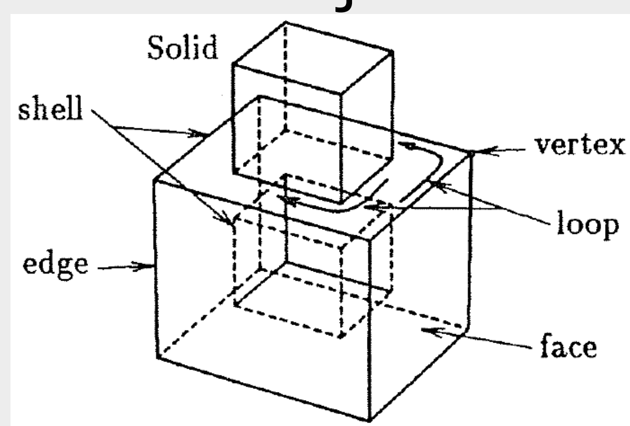
A typical solid model is defined by solids, surfaces, curves, and points.



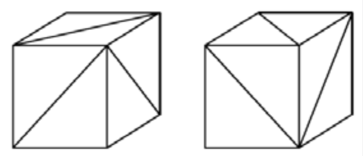
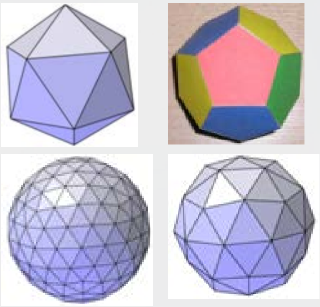
Solids are bounded by surfaces.
They represent solid objects.
Analytic shape

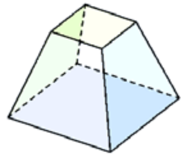
Surfaces are bounded by lines. They represent surfaces of solid objects, or planar or shell objects.

Curves are bounded by points.
They represent edges of objects.



Geometric Modeling



There is a built-in hierarchy among solid model entities.

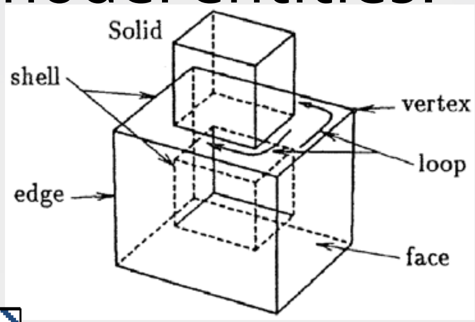
Points are the foundation entities.

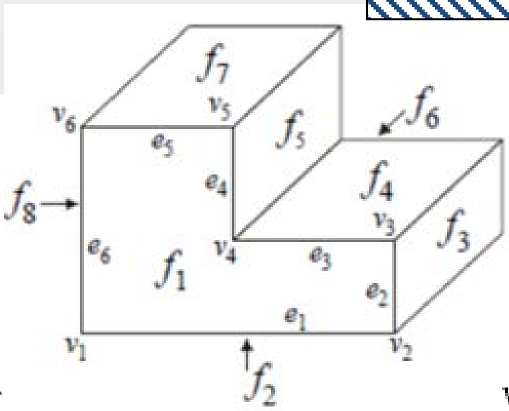
Curves are built from the points,

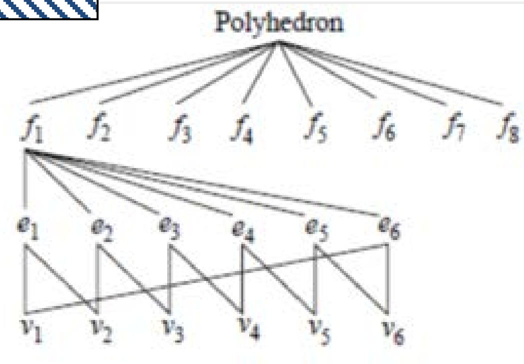
Surfaces from curves,

Solids from surfaces.

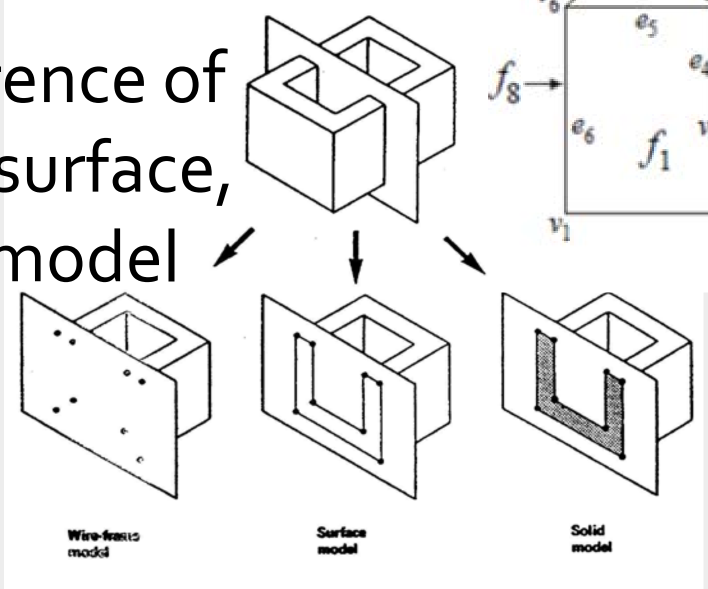
Solids
Surfaces
Curves
Points

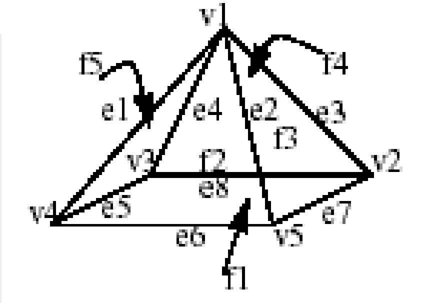


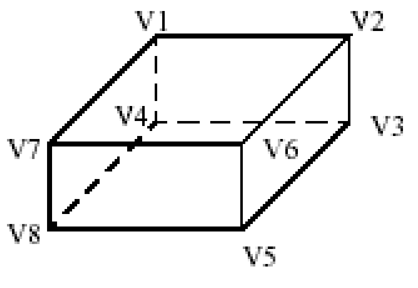




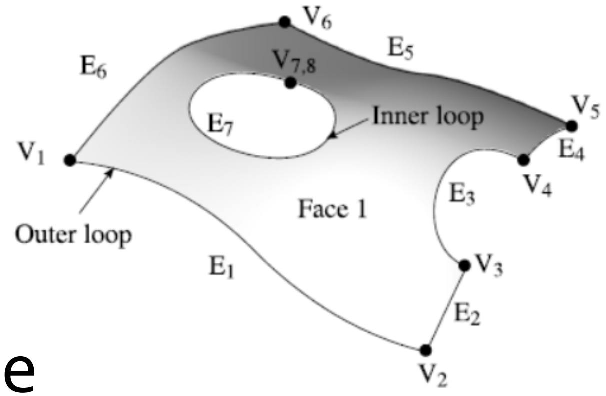
Difference of wire, surface, solid model



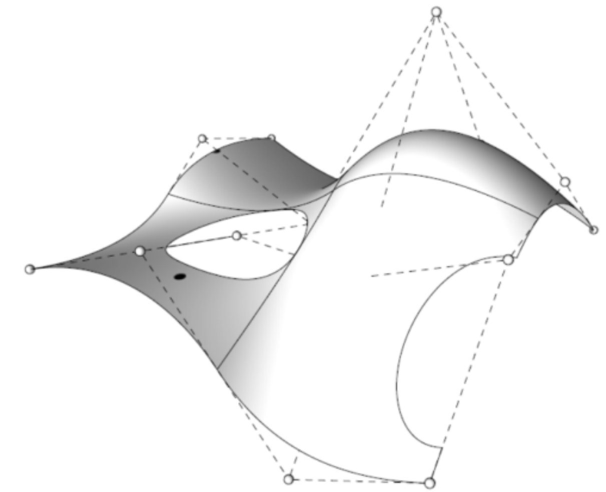




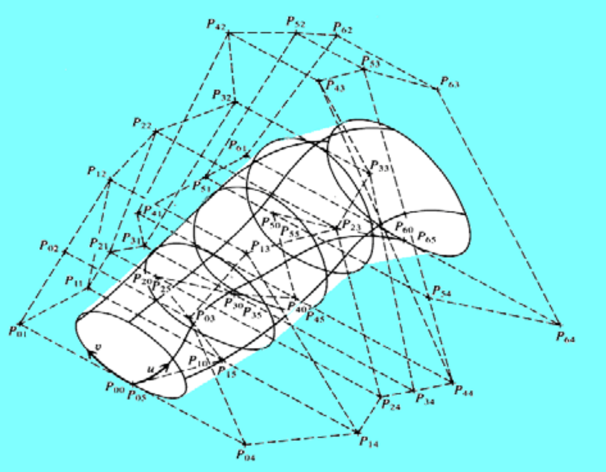
Surface Modeling

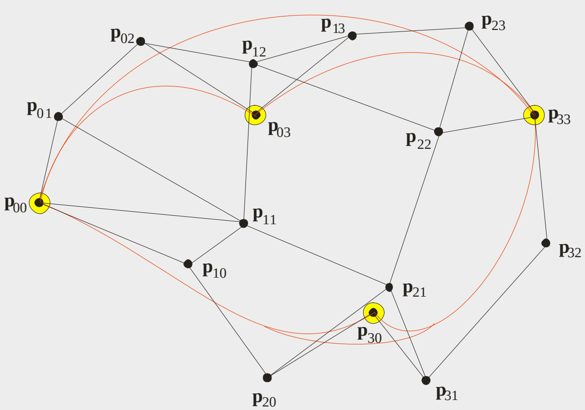


Bezier, B–spline and NURBS surface
is a tensor product surface
and is the product of two curves.



Surfaces are defined by grid
and have two sets of parameters,
two sets of knots, control points
and so on.

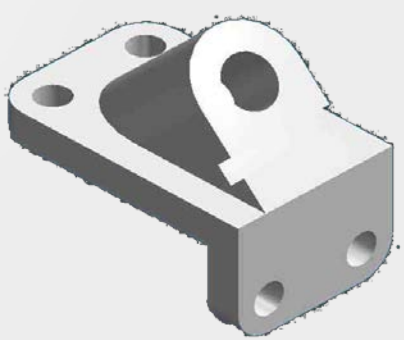
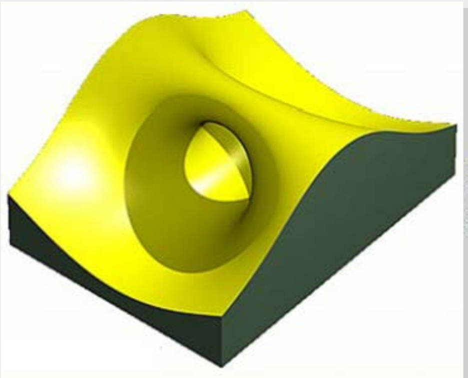
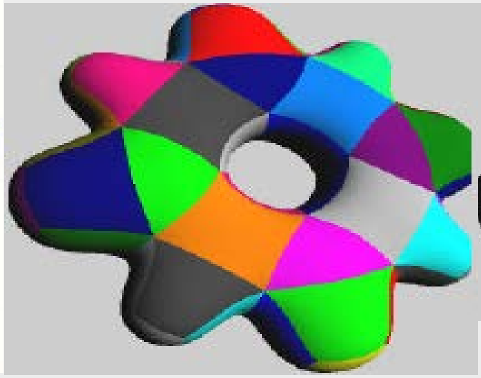
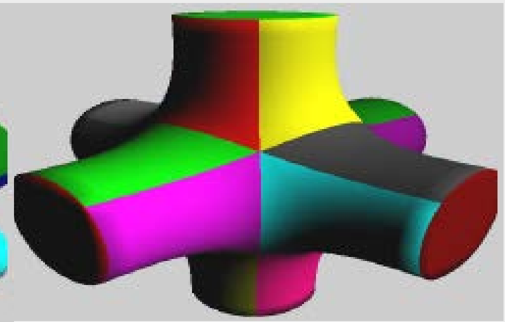




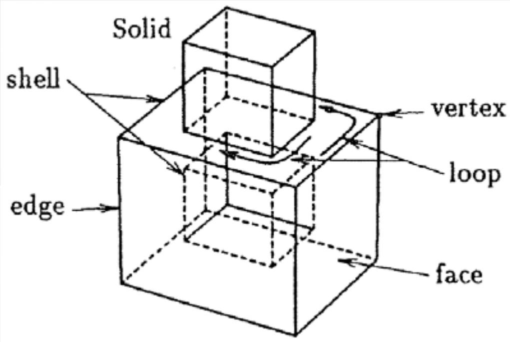
Solid Modeling

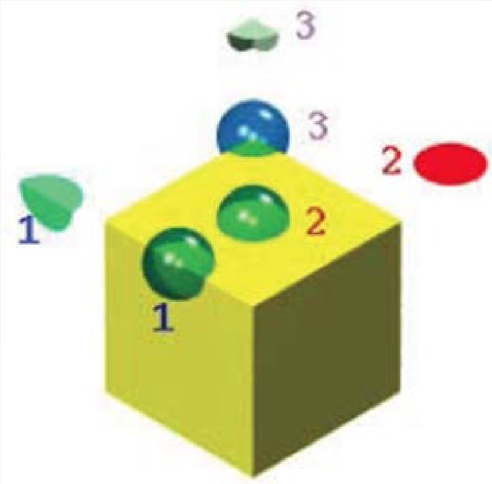
Solid Models are complete, valid and unambiguous.
Models have interior, volume, and mass properties.

While no representation can describe all possible solids,
a representation should be able to represent
a useful set of geometric objects.

Solid Modeling



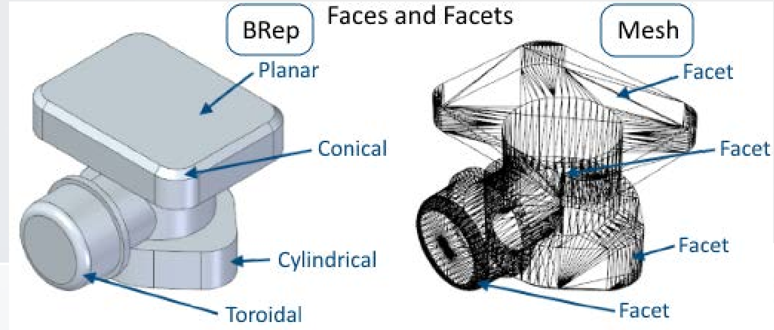


A solid object is defined by the interior volume space contained within the defined boundary of the object.

A closed boundary is needed to define a solid object,

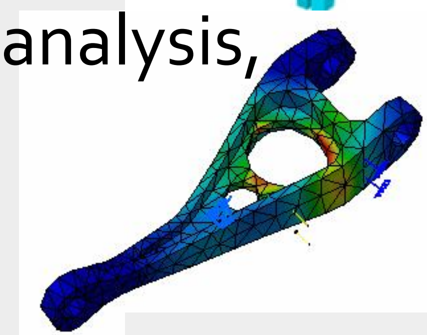
- informationally complete, compact, valid representation
- points in space to be classified relative to the object, if it is inside, outside, or on the object
- store both **geometric and topological information**, can verify whether two objects occupy the same space
- improves the quality of design, improves visualization, and has potential for functional automation and integration.

Solid Modeling



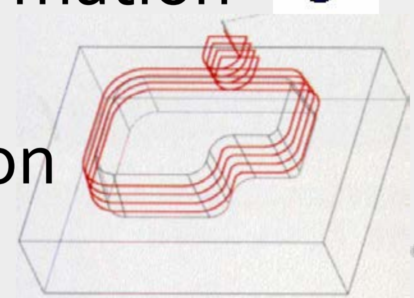
Support using volume information

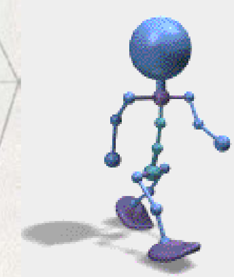
- weight or volume calculation,
  centroids, moments of inertia calculation,
- stress analysis (finite elements analysis),
  heat conduction calculations, dynamic analysis,
- system dynamics analysis



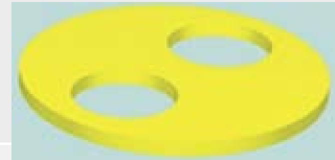
Using volume and boundary information

- generation of CNC codes,
  robotic and assembly simulation





Solid Modeling



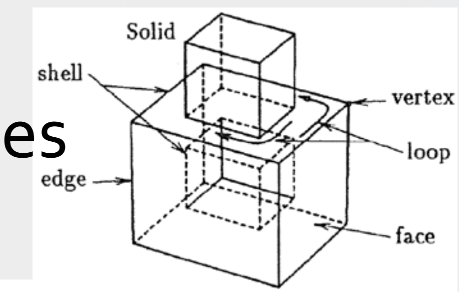
Solids models must satisfy the following criteria:

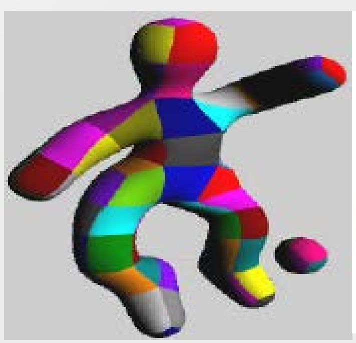
Rigidity: Shape of object remains fixed when manipulated.

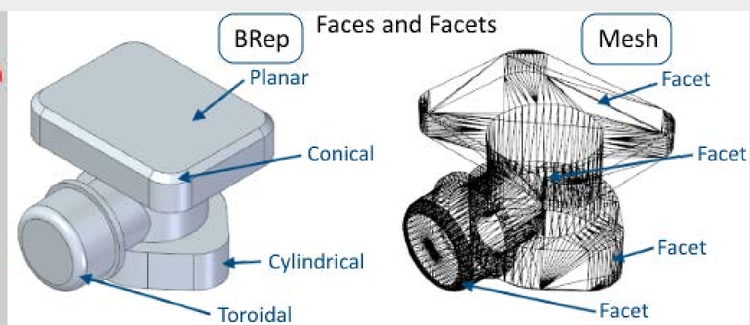
Homogeneity: All boundaries remain in contact.

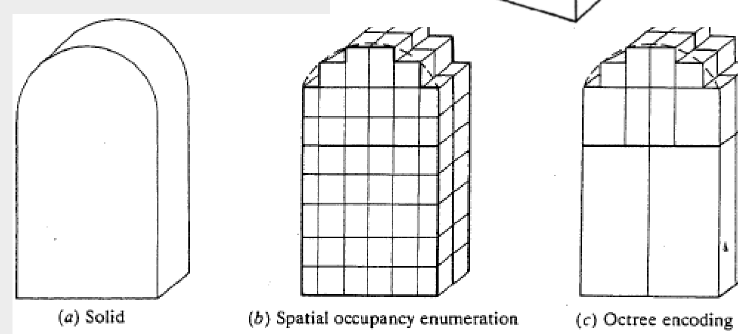
Finiteness: No dimension can be infinite.

Divisibility: Model yields valid sub-volumes when divided.









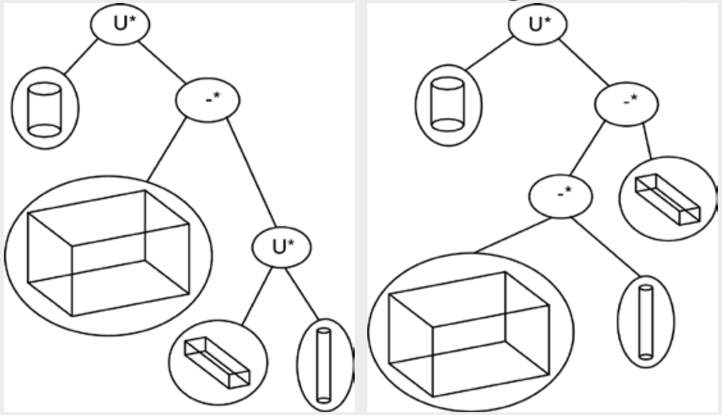
Requirements for Solid Representation

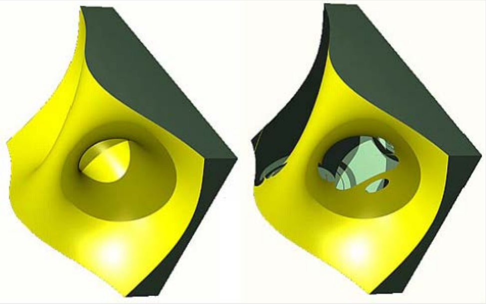
Uniqueness

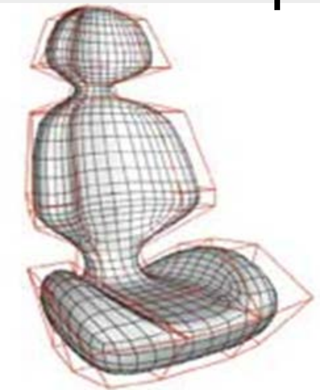
That is, there is only one way to represent a particular solid. If a representation is unique, then it is easy to determine if two solids are identical since one can just compare their representations.

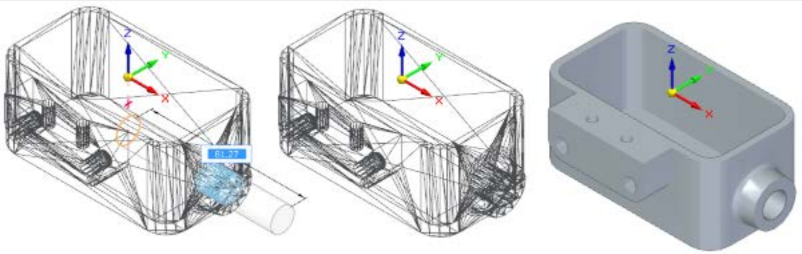
Accuracy

A representation is said **accurate** if no approximation is required.







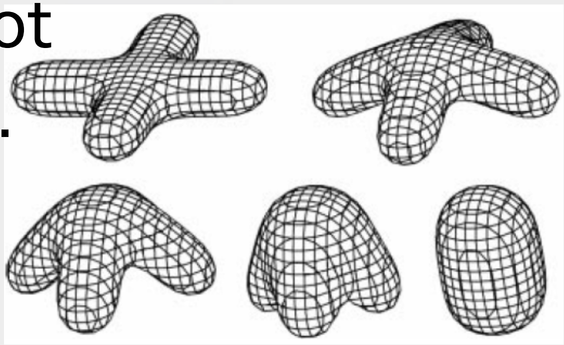


Parametric Hole in Mesh Geometry

Requirements for Solid Representation

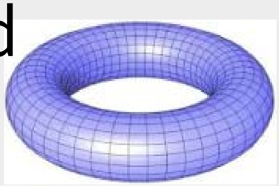
Validness

This means a representation should not create any invalid or impossible solids.

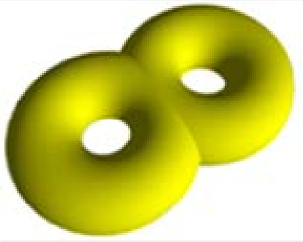
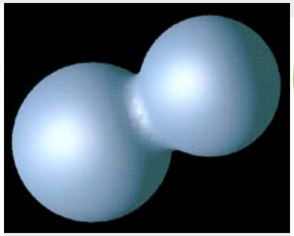
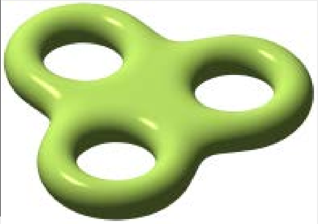


Closure

Solids will be transformed and used with other operations such as union and intersection. "**Closure**" means that transforming a valid solid always yields a valid solid



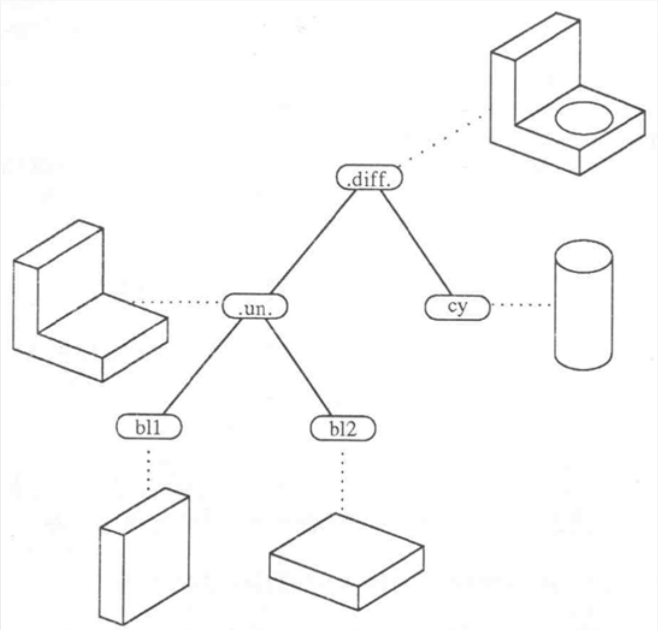


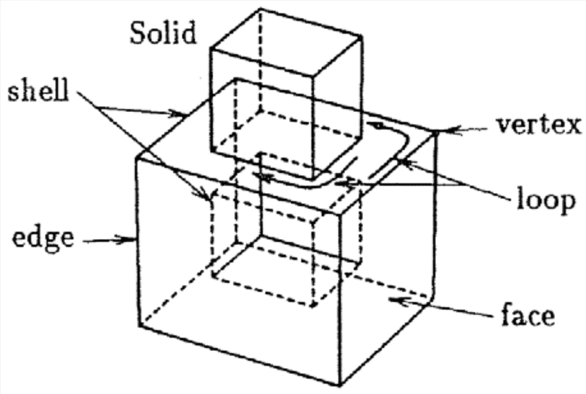





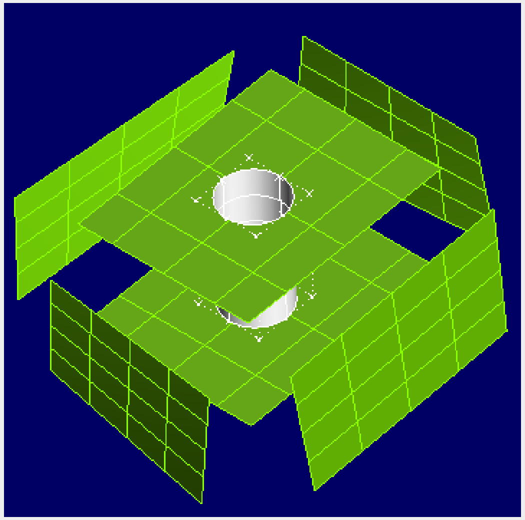
Requirements for Solid Representation

Compactness and Efficiency

A good representation should be compact enough for saving space and allow for efficient algorithms to determine desired physical characteristics



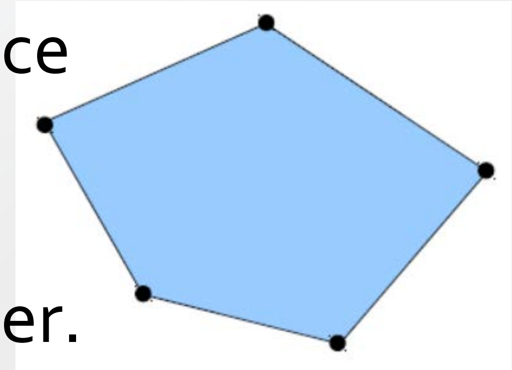


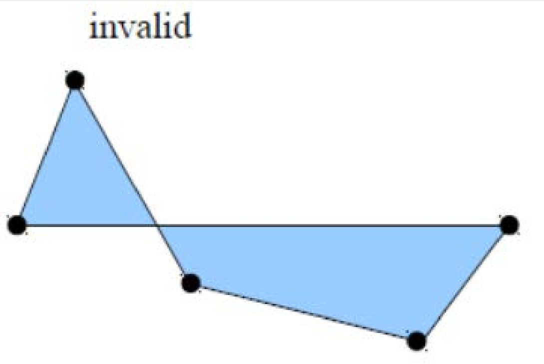


Requirements for Solid Representation

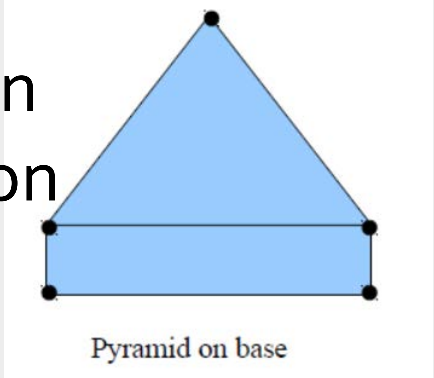
Validity of the B-Rep (Boundary representation) Solid model

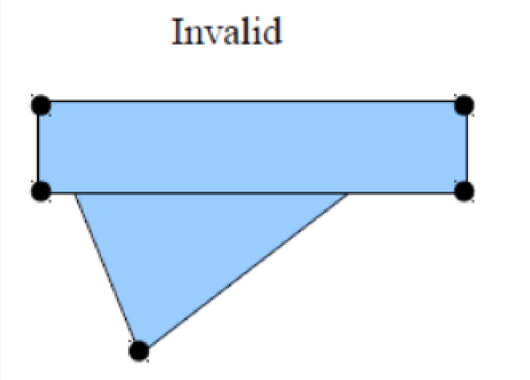
The boundary of a face is made up of edges that are not allowed to intersect each other.



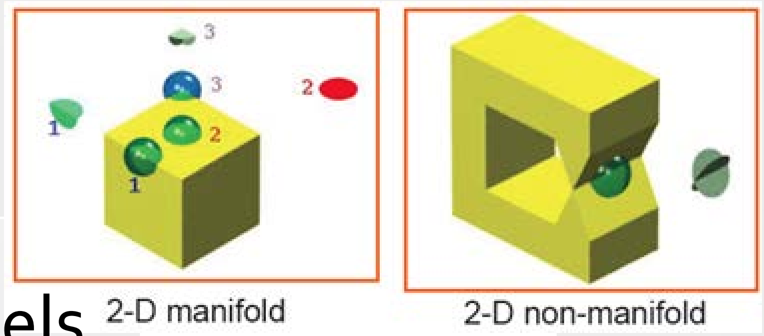


The faces of a model can only intersect in common edges or vertices.

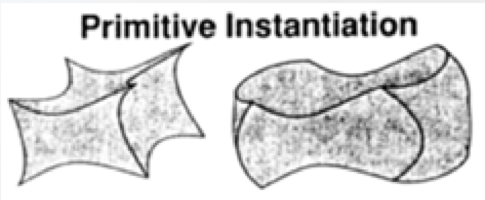


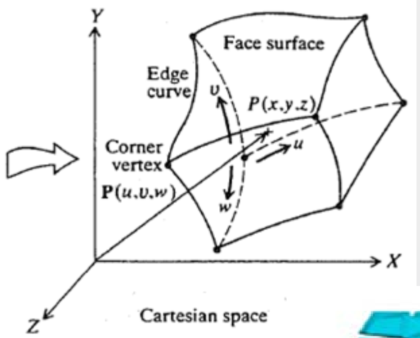


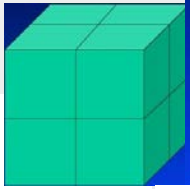
Solid modeling techniques

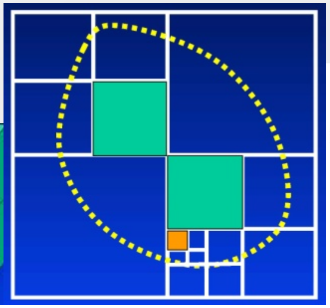


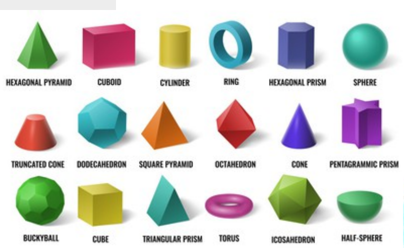
Surface Based 2-Manifolds Models,
3D Parametric Solid,
Primitive Instancing,
Space Subdivision,
Cell Decompositions,
Octree Model,

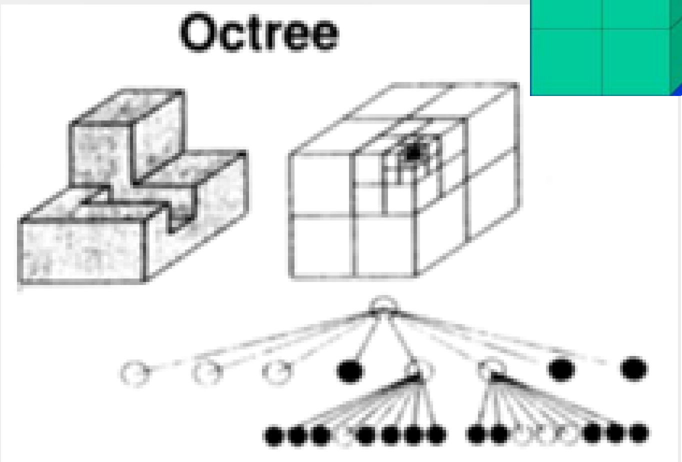


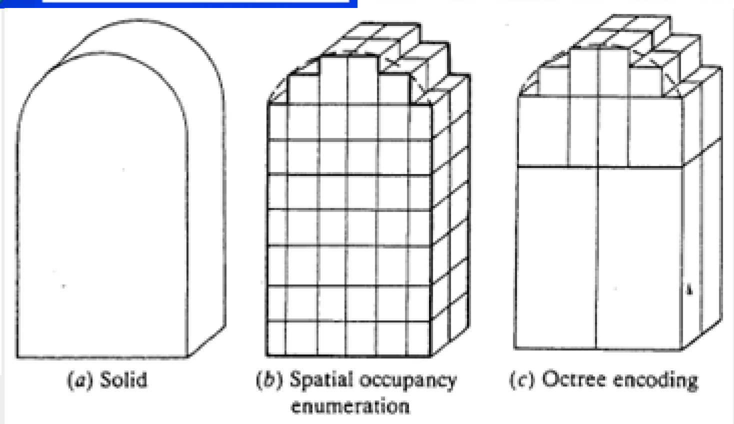






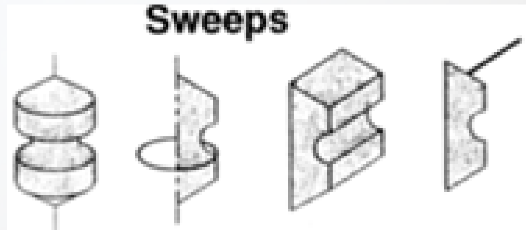


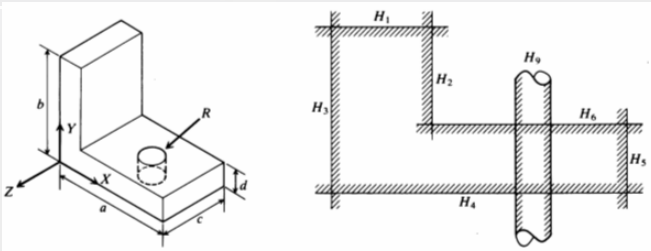


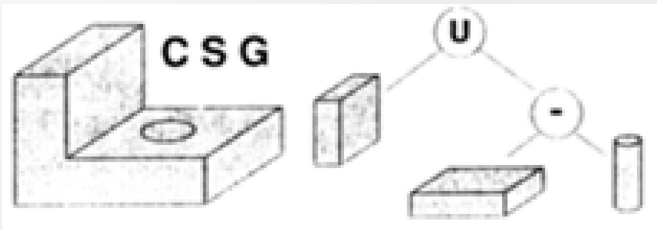


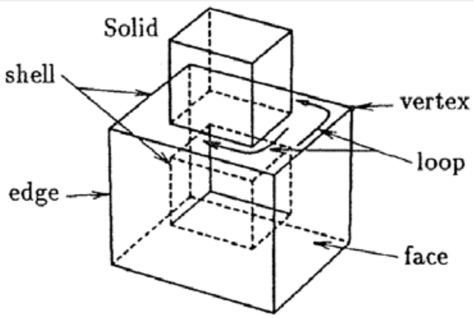
Solid modeling techniques

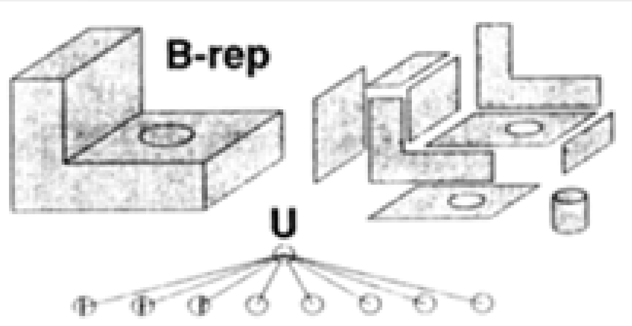
Sweeping,
Half Spaces,
CSG,
B-rep

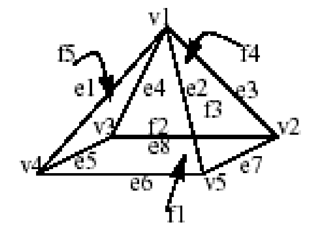


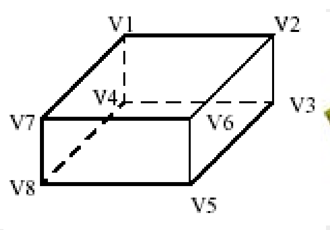


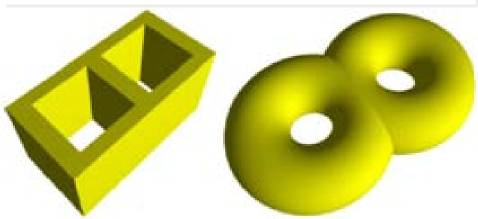


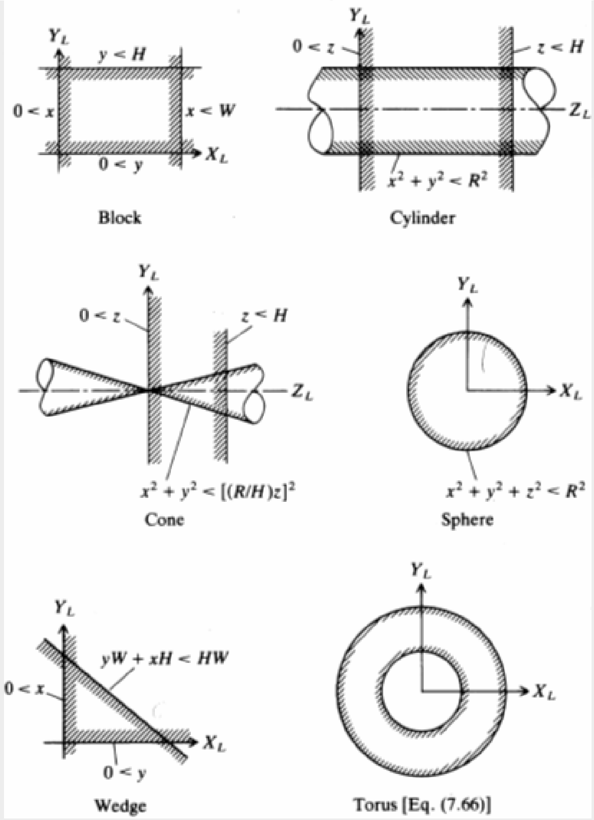






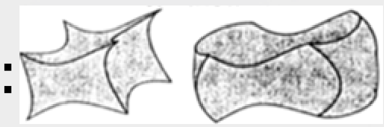


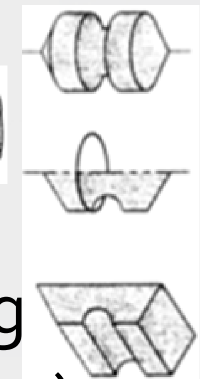




Solid Modeling (cont.)

The most common solid-modeling techniques used by CAD systems are:



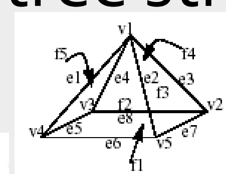


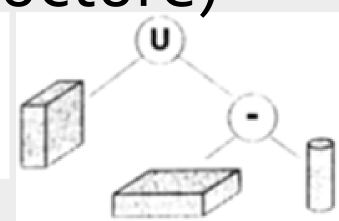
- Pre-defined geometric Primitive instancing,
- Sweeping in the form of extrusion and revolving
- Constructive Solid Geometry (CSG tree structure)
- Boundary representation (B-rep)
- Feature Based Modeling

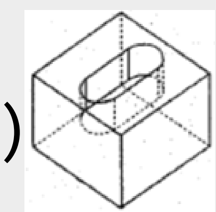
(uses feature-based primitives)

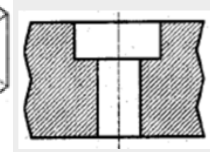
- Parametric Modeling

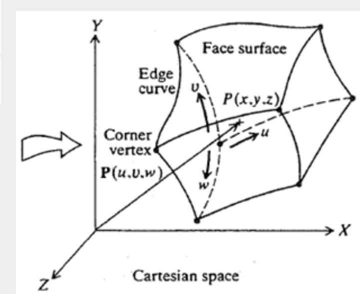
(ASM, uses 3D parametric solid)



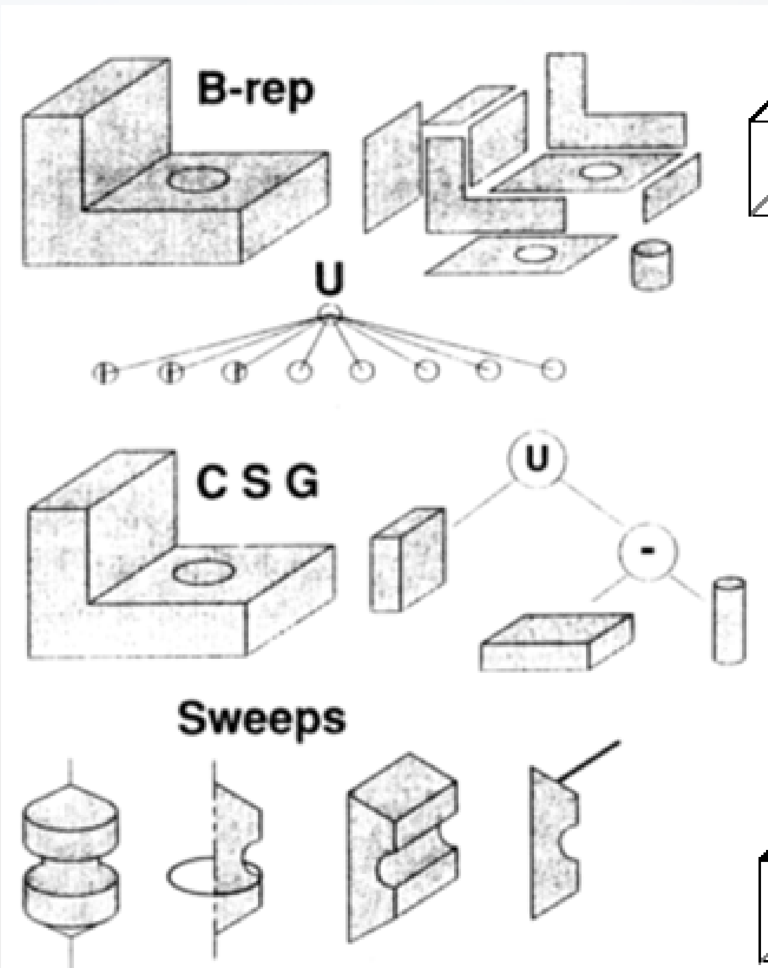




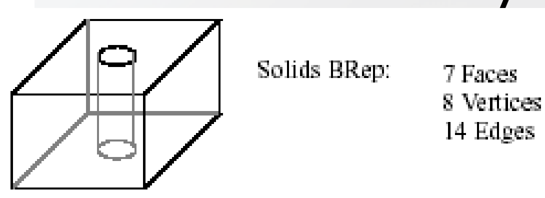




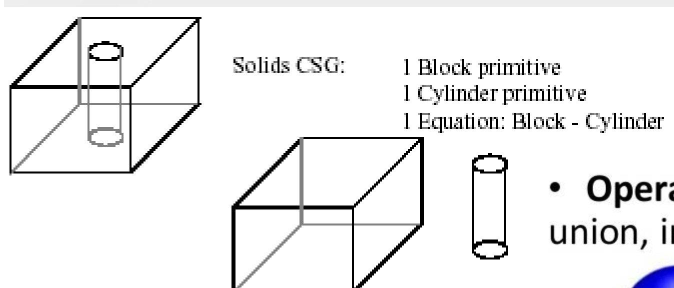
Solid modeling approaches



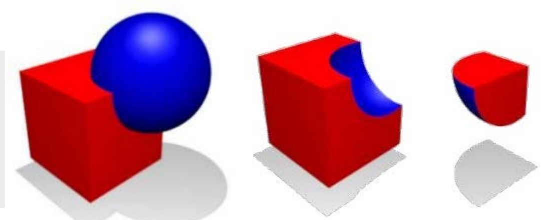
Boundary Representation (B-rep)



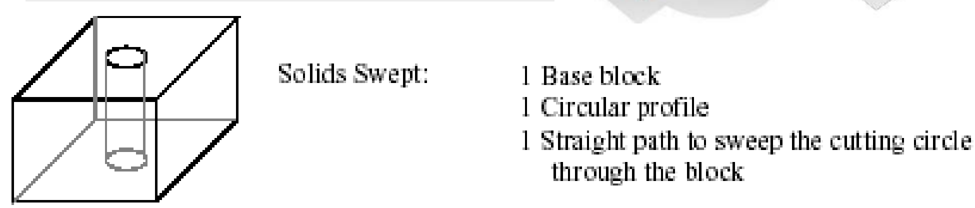
Constructive Solid Geometry (CSG)



- **Operations**: union, intersection and difference.

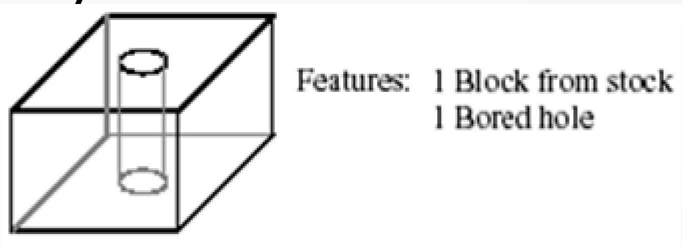


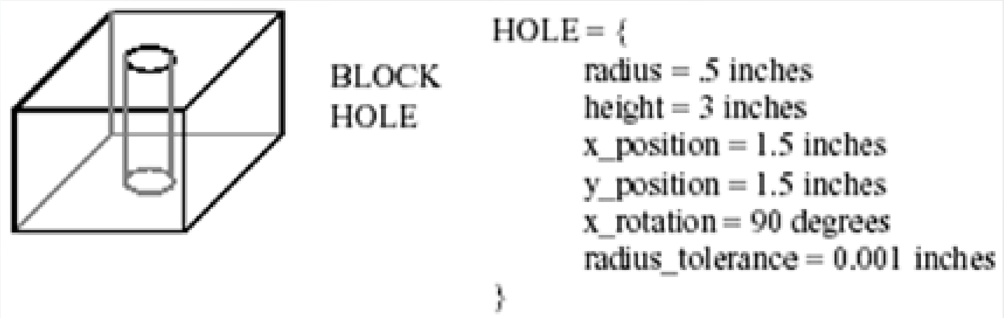
Sweep



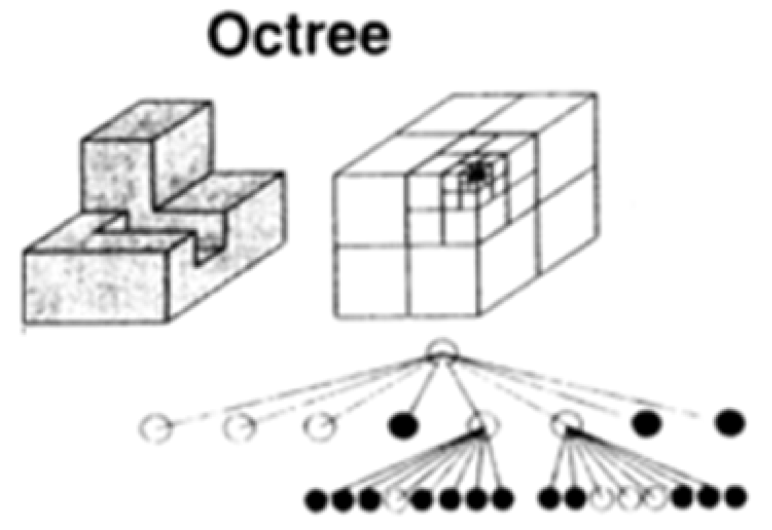
Solid modeling approaches

Hybrid (Feature based modelers)



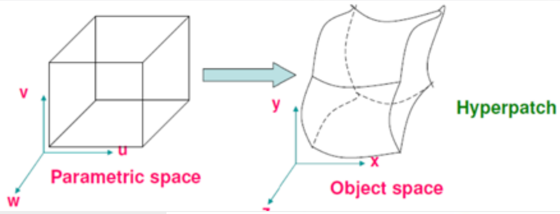


Octree Modeling

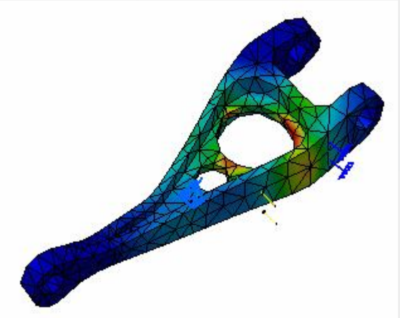


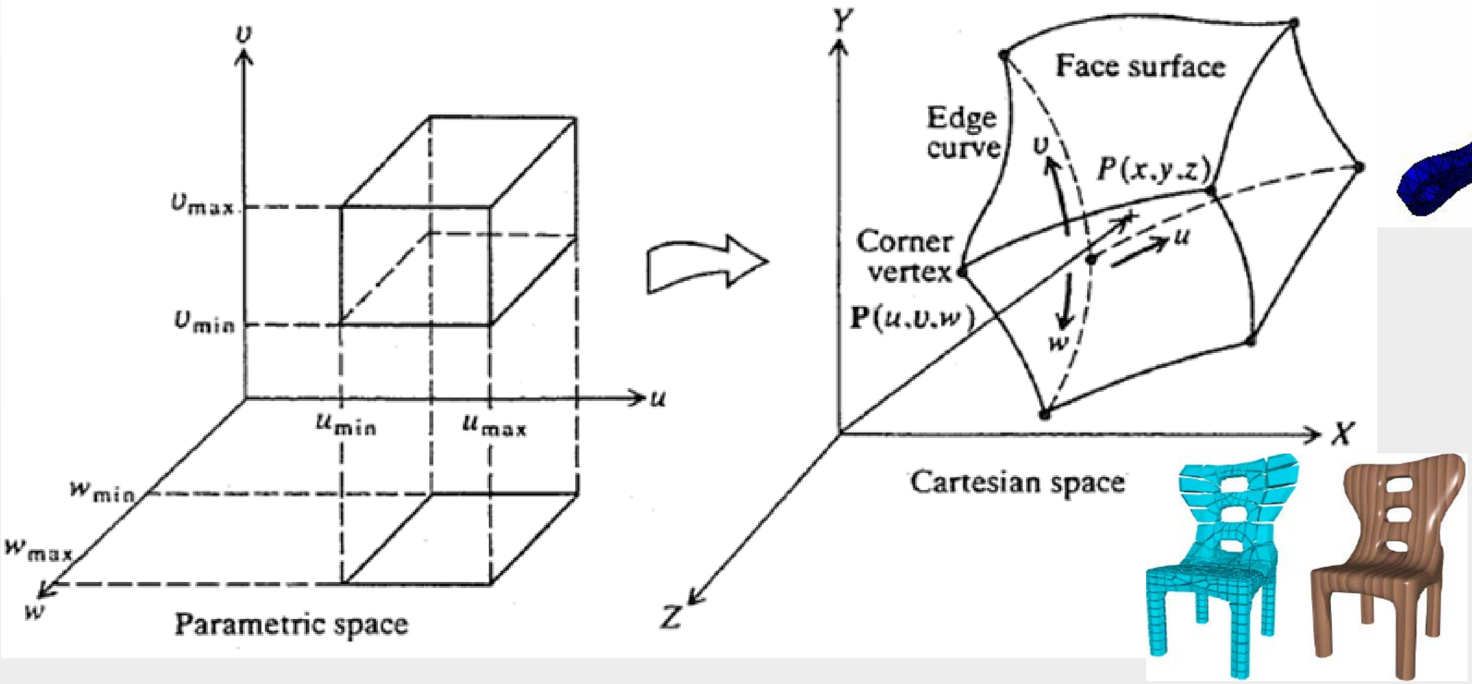
Parametric Solid
Analytical Solid Modeling (ASM, FEM)

$$P(u,v,w)=\sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{l}p_{i,j,k}B_{i,4}(u)B_{j,4}(v)B_{k,4}(w)$$

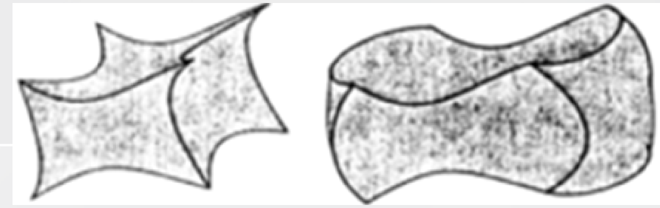


$$x = x\ (u,v,w) \quad y = y\ (u,v,w) \quad \text{and} \quad z = z\ (u,v,w)$$

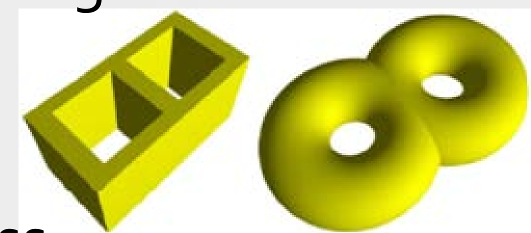




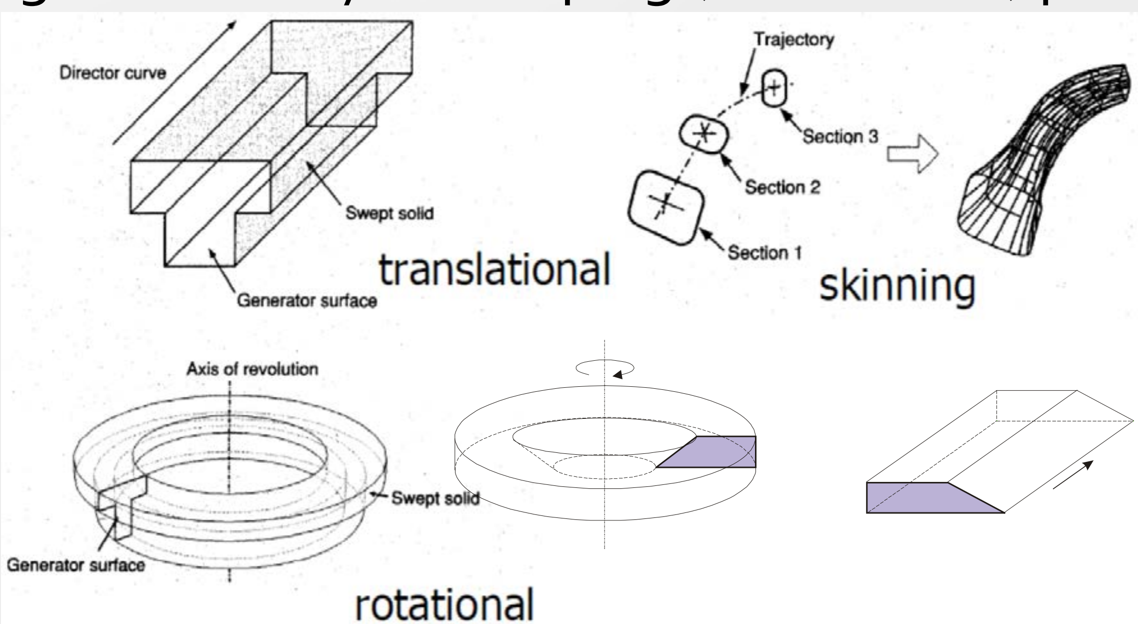
Primitive Instancing and Sweeping

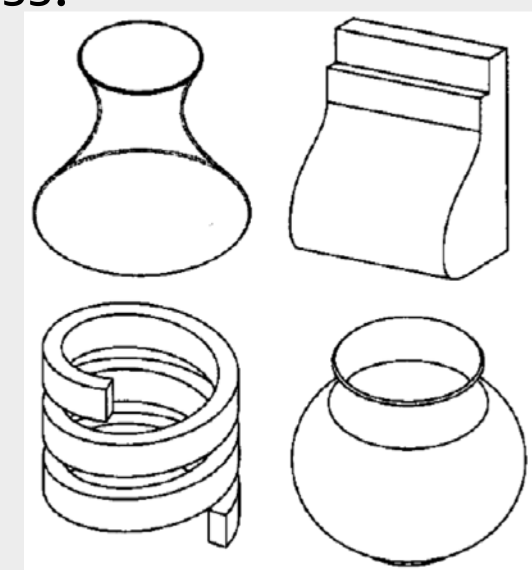


Primitive instancing (Feature) refers to the scaling of simple geometrical models (primitives) by manipulating one or more of their descriptive parameters.



Most simple geometric primitives can be generated by a sweeping (“extrusion”) process.

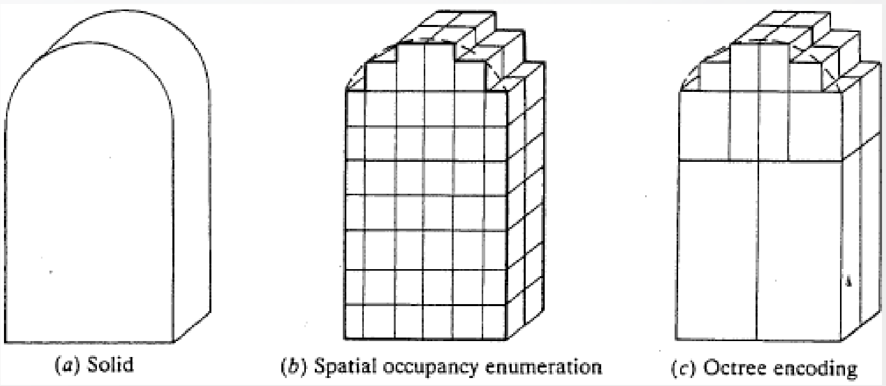


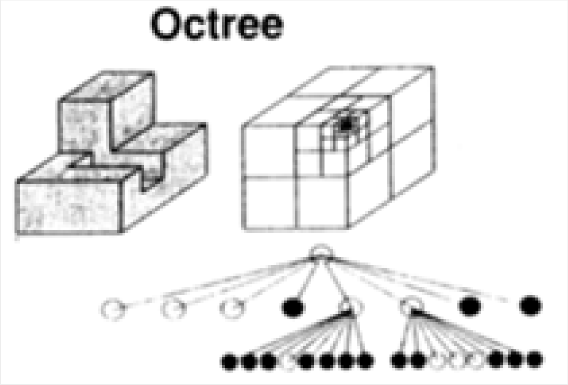


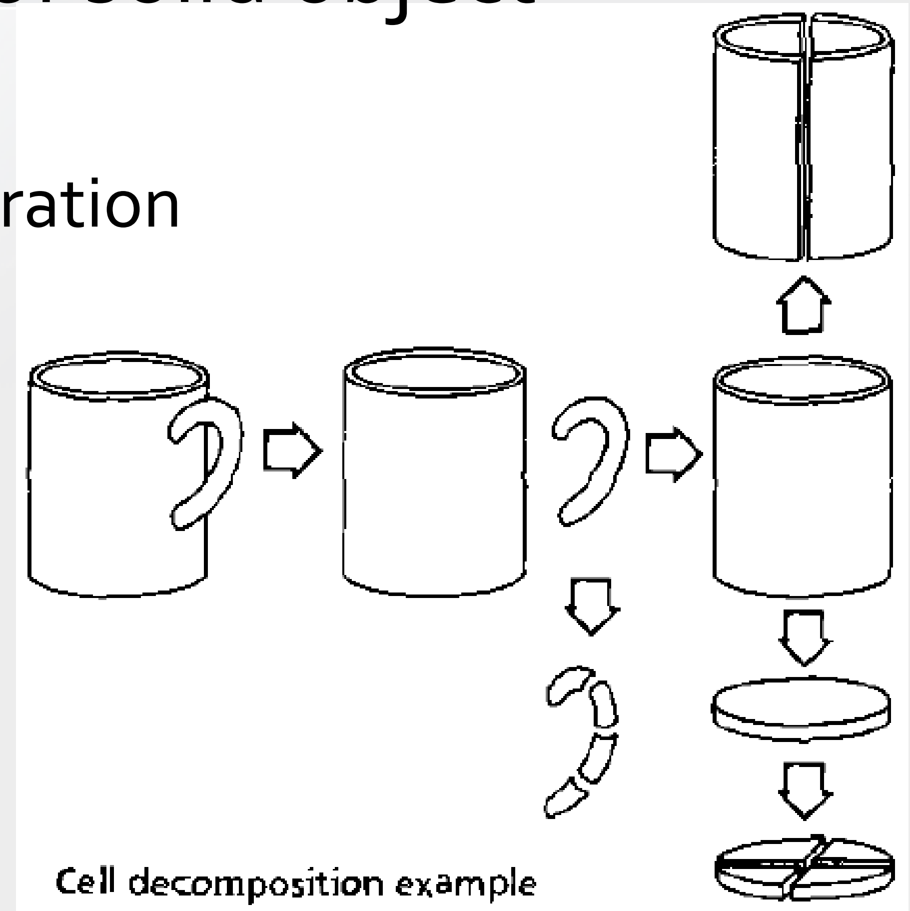
Cell decomposition of solid object

Space partitioning model

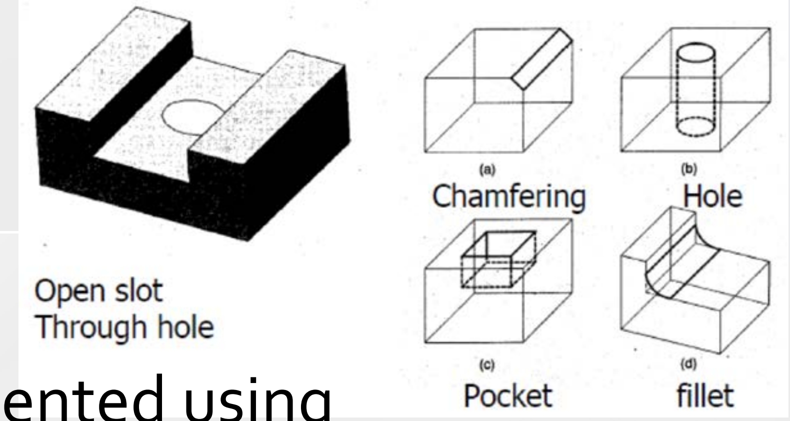
Spatial-occupancy enumeration







Feature-based modeling



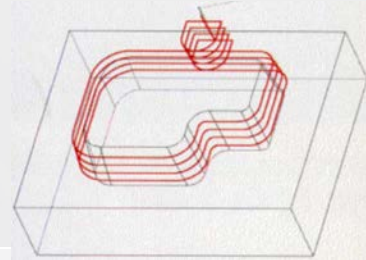
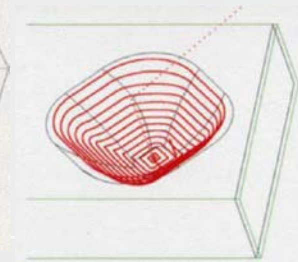
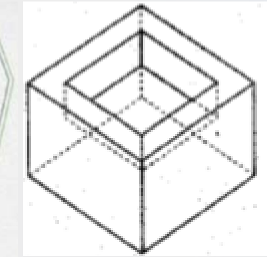
Past Approach

The graphical information is represented using low level graphical elements such as points, lines, arcs, etc. The textual information is represented as texts, notes and symbols attached to a drawing.

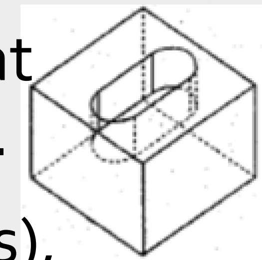
Ideal/Present Approach – feature-based modeling

To represent part geometry using high-level feature primitives such as holes, slots, pockets, etc. (consistent to the engineering practice), and to represent dimensions, tolerances, surface finishes, etc. as meaningful design entities.

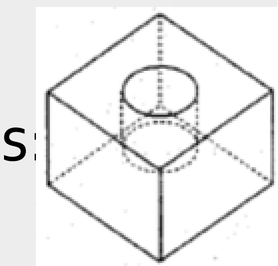
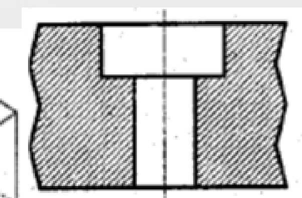
Feature-Based Design

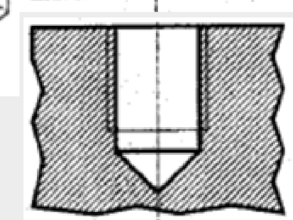
Features are specific **geometrical shapes** on a part that can be **associated with certain fabrication processes**.



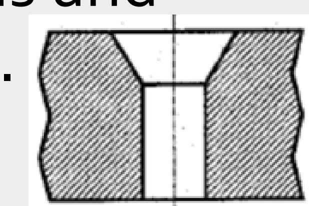
Features can be classified as form (geometric elements), material, precision (tolerancing data), and technological (performance characteristics).

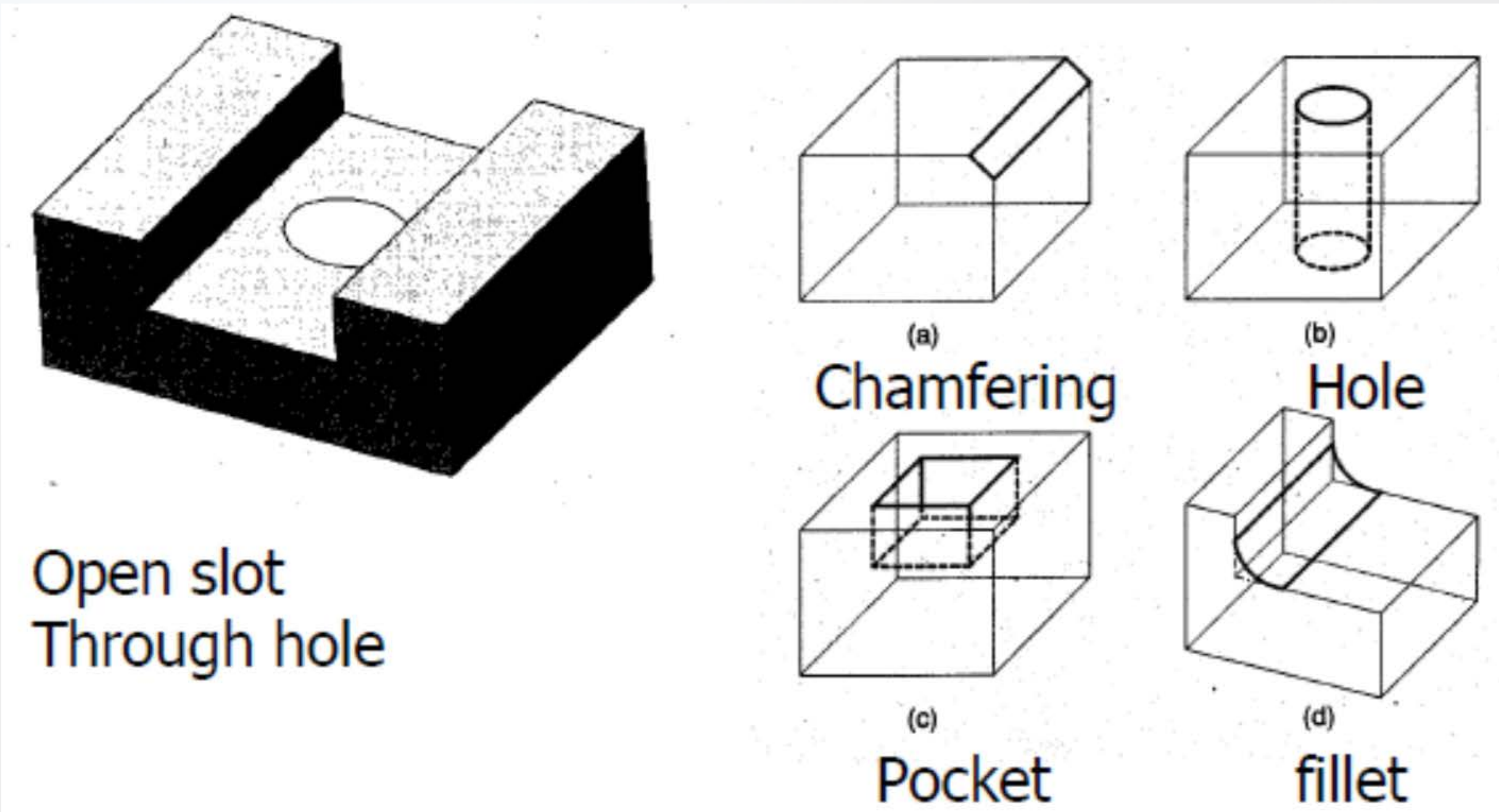
The primary **objectives** of design by features:



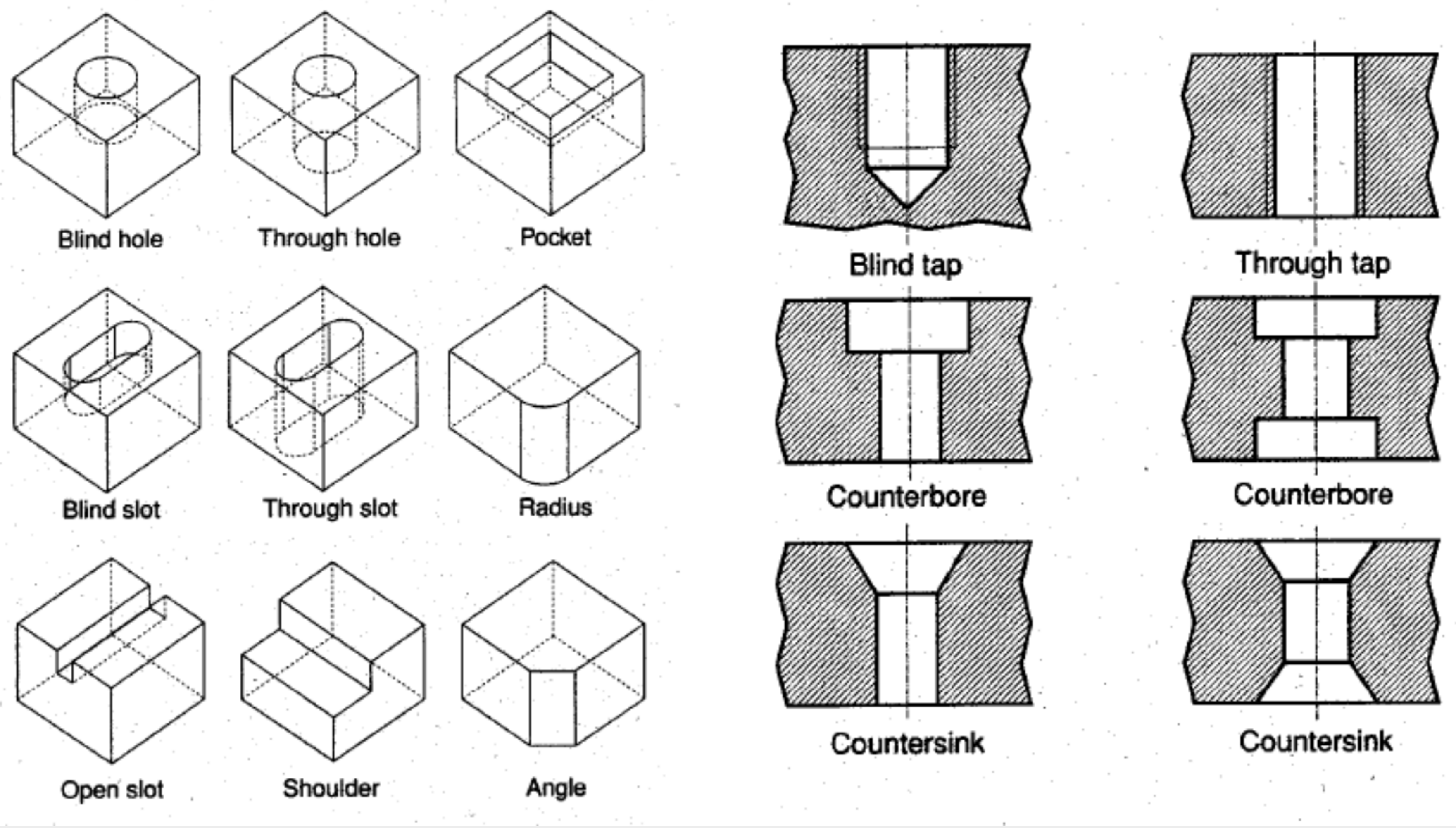
- **Increase the efficiency** of the designer during the geometric-modeling phase, and
- **Provide a bridge** (mapping) to engineering-analysis and **process-planning phases** of product development.



Feature-Based Design



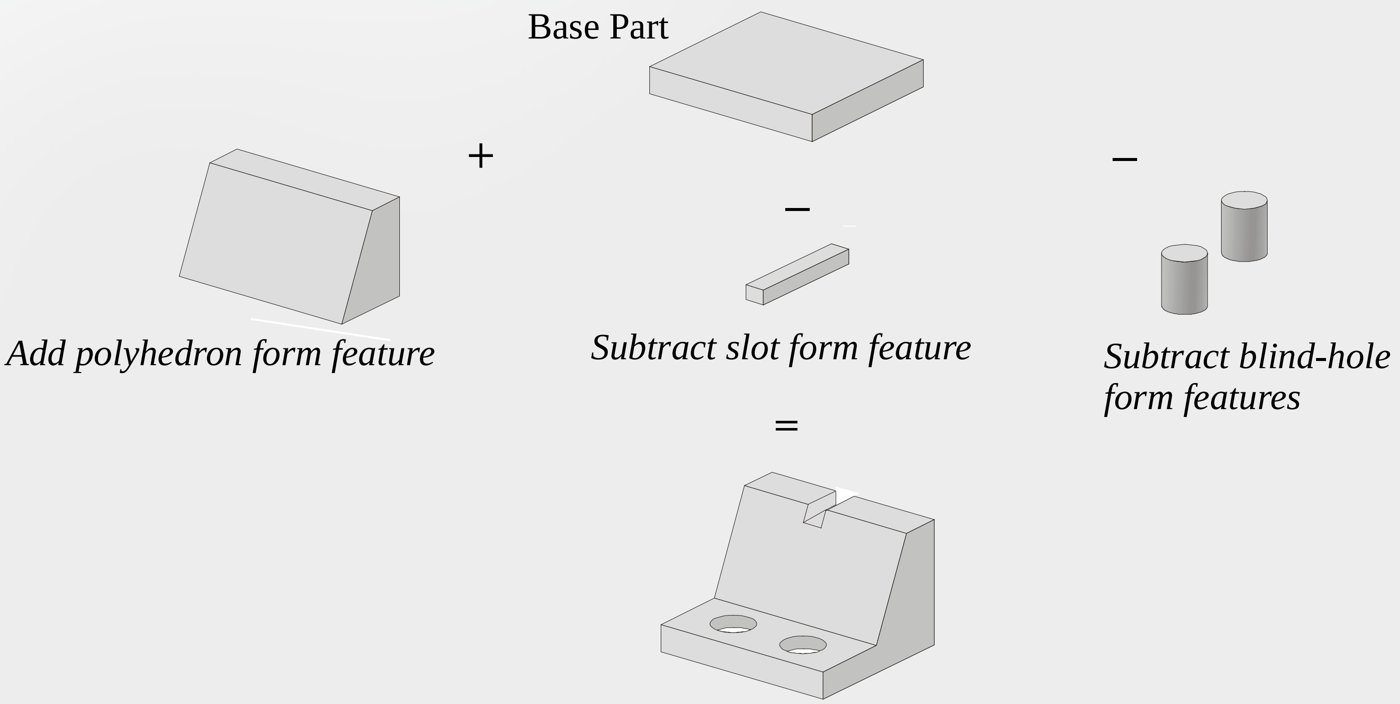
Machining Features



Design by Features

A solid model is configured through a sequence of form-feature attachments to the primary representation of the part.

Features could be chosen from a library of pre-defined features or could be extracted from the solid models of earlier designs.



Feature Recognition

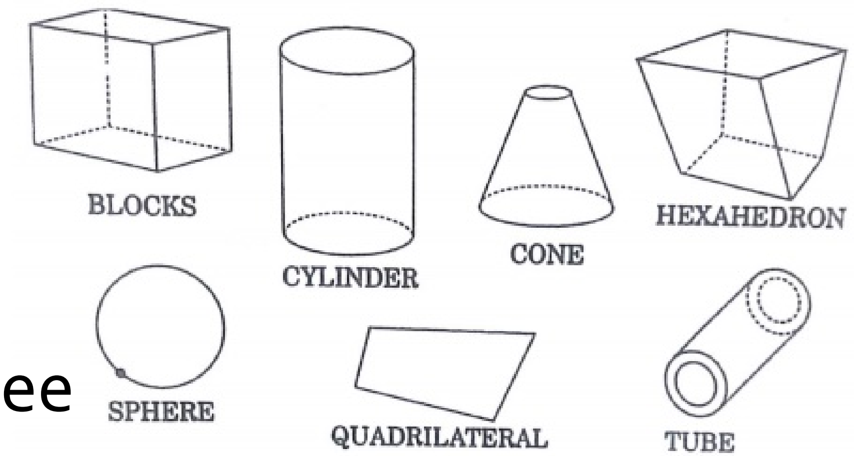
Currently, feature recognition refers to examination of parts' solid models for the **identification of predefined features** and for their **extraction**.

In the future, extraction methods will examine a part's solid model for the existence of geometric features that have not been predefined and extract them:

- Such features would, then, be classified and coded for possible future use in a **Group technology GT**-based CAD system - Namely, these features would be extractable based on a user-initiated search for the most-similar feature in the database via a GT-code.

Constructive Solid Geometry (CSG)

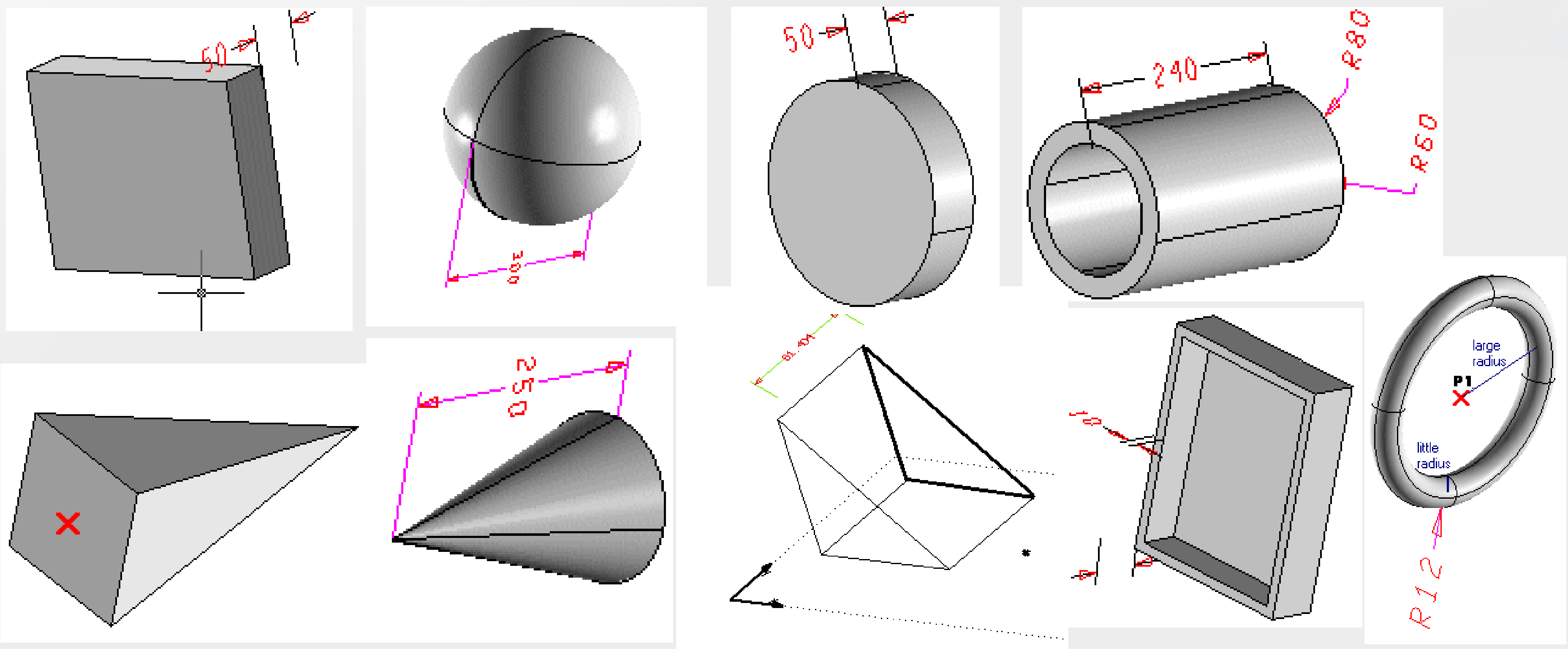
- Based on simple geometric primitives
  - cube, parallelepiped, prism, pyramid, cone, sphere, torus, cylinder, solid by points etc.
- Primitives are positioned and combined using boolean operations
  - **union** (addition)
  - **difference** (subtraction)
  - **Intersection**
- Represented as a boolean tree



Solid primitives.

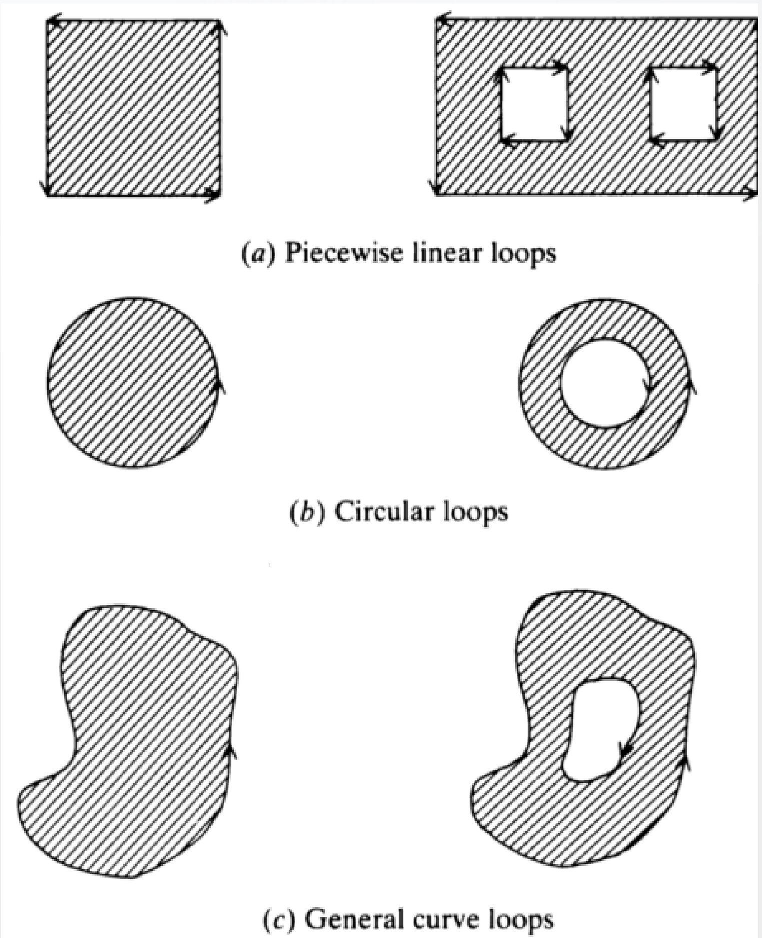
CSG Primitives

Based on simple geometric primitives:
cube, parallelepiped, prism, pyramid, cone, sphere, torus, cylinder, solid by points etc.



Half Spaces used in CSG modeling

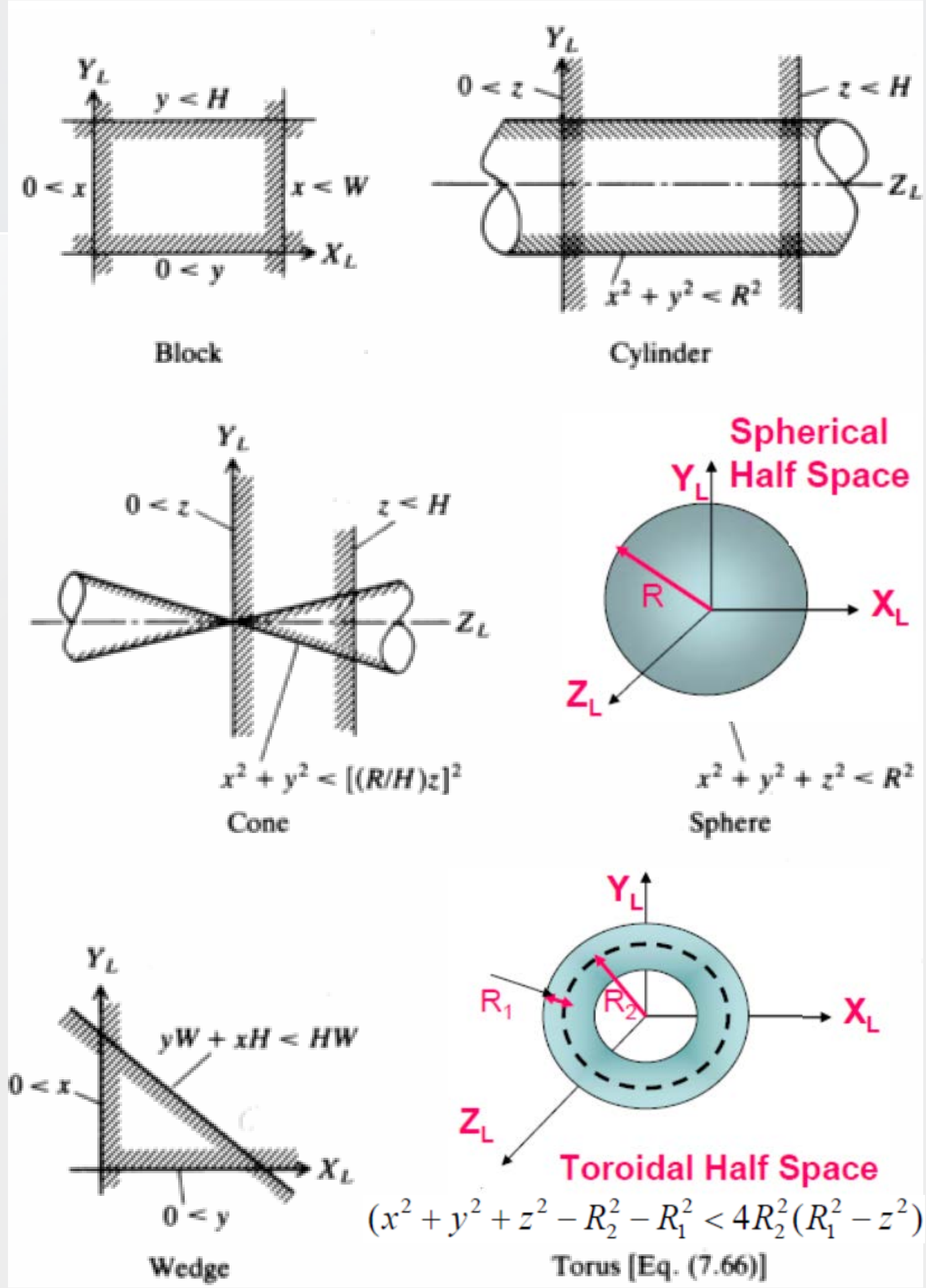
Surface descriptions



(a) Piecewise linear loops

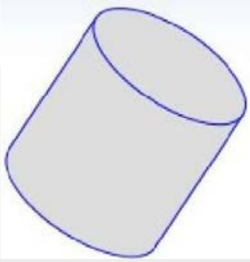
(b) Circular loops

(c) General curve loops

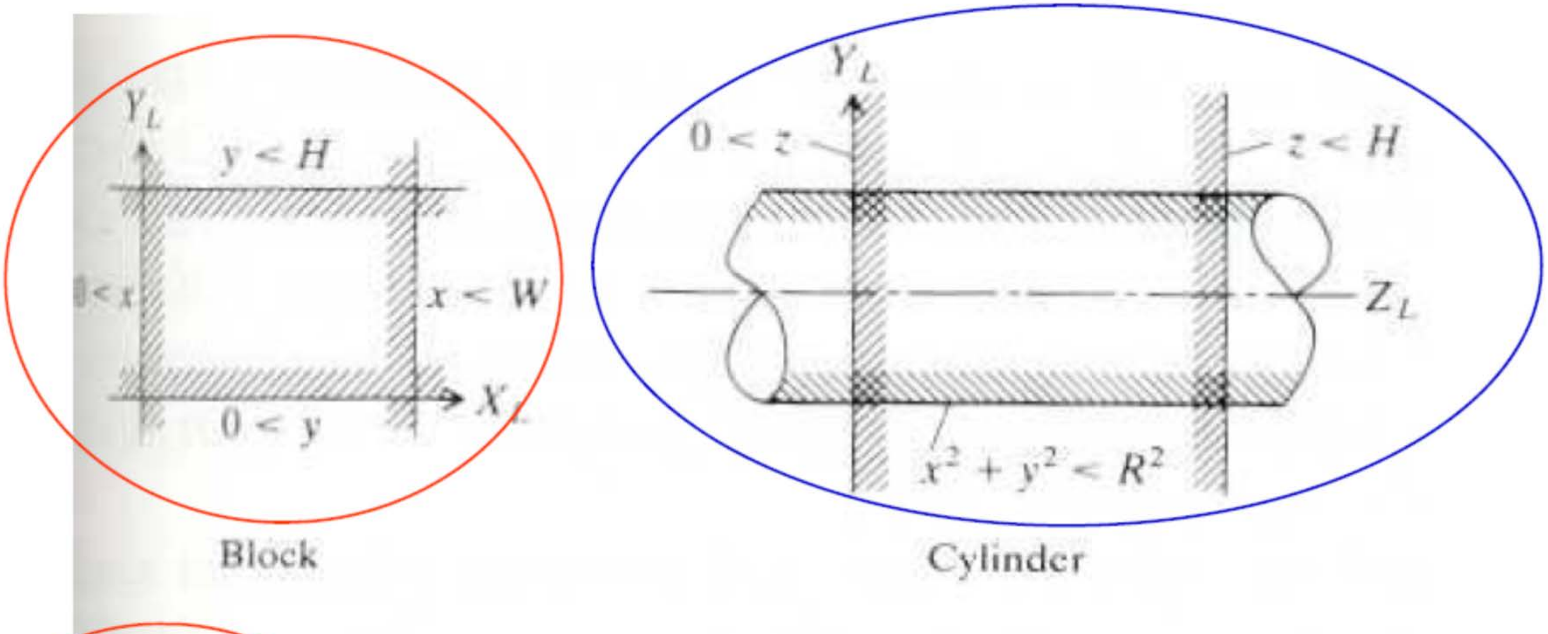


CSG Half Spaces

- Infinite cylinder, I: x^2 + y^2 – r^2 <= 0
- Infinite planar halfspace, P: Ax + By + Cz + D <= 0
- Cylinder with ends: I ^ P1 ^ P2



- Planar half-space $H = \{(x,y,z) : z < 0\}$
- Cylindrical half-space $H = \{(x,y,z) : x^2 + y^2 < R^2\}$

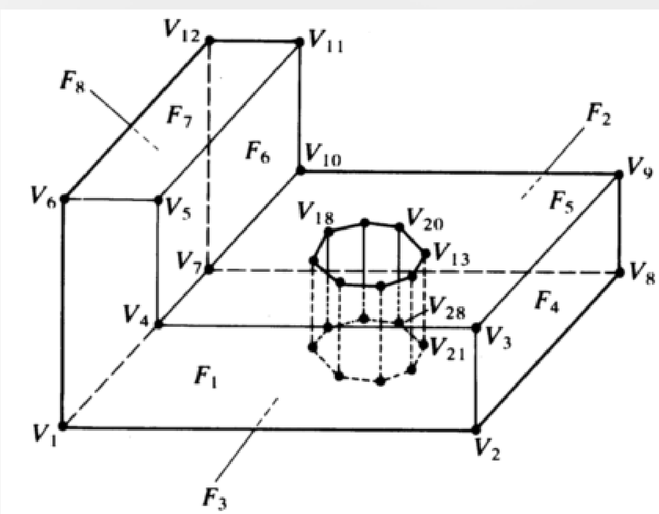


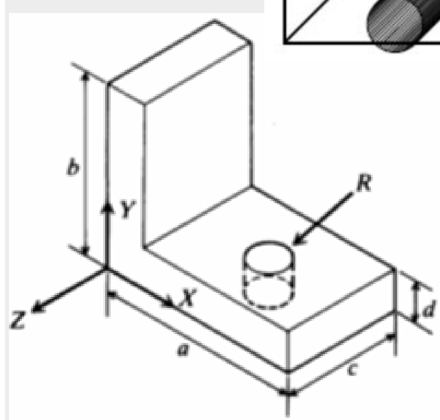
Block: $\{(x,y,z) : 0 < x < W,\ 0 < y < H,\ and\ 0 < z < D\}$

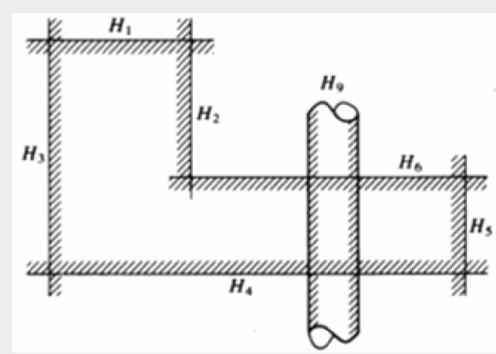
Cylinder: $\{(x,y,z) : x^2 + y^2 < R^2,\ and\ 0 < z < H\}$

CSG modeling by Half Spaces

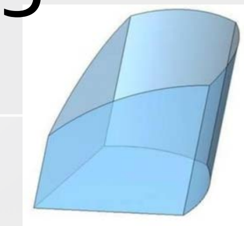
The solid modeling technique is based upon
the "half-space" concept using set operations.
The boundary of the model separates the interior and exterior of the modeled object. Half spaces form a basic representation scheme for bounded solids.
Example of Half Spaces:

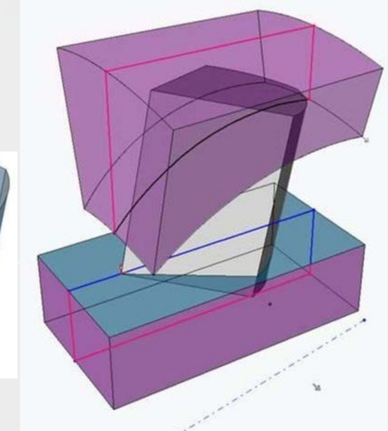






Advantages and Disadvantages of Half Spaces

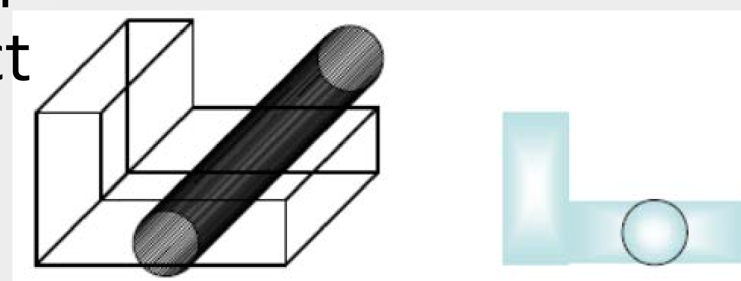




Advantages:

The main advantage is its **conciseness** of representation compared to other modeling schemes.

It is the lowest level representation available for modeling a solid object



$$S = \bigcup \left(\bigcap_{i=0}^{n} H_i \right)$$

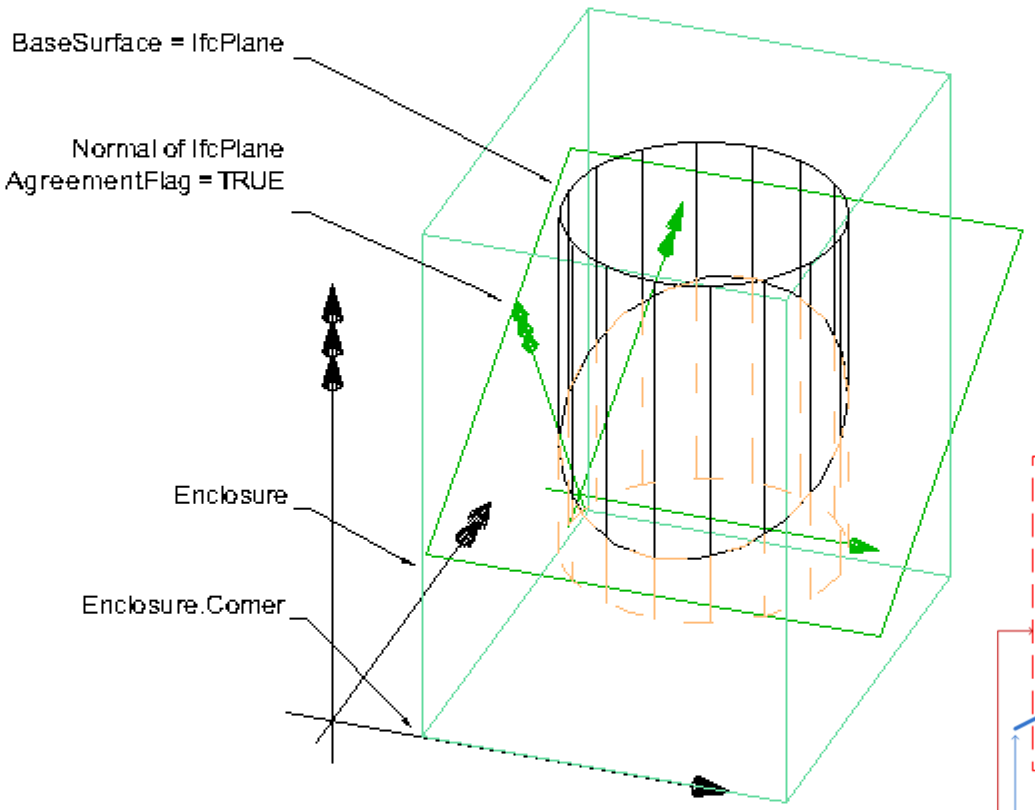
Disadvantages:

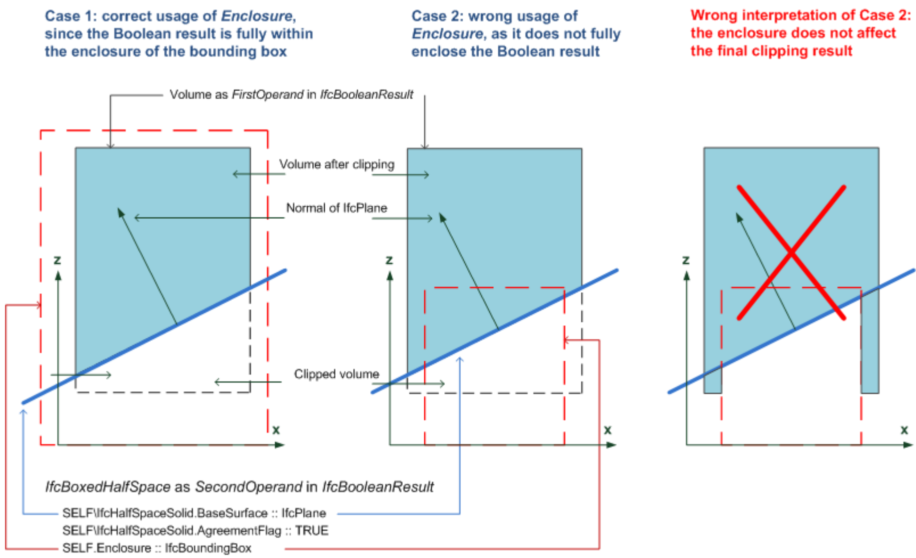
The representation can lead to **unbounded** solid models as it depend on user manipulation of half spaces.

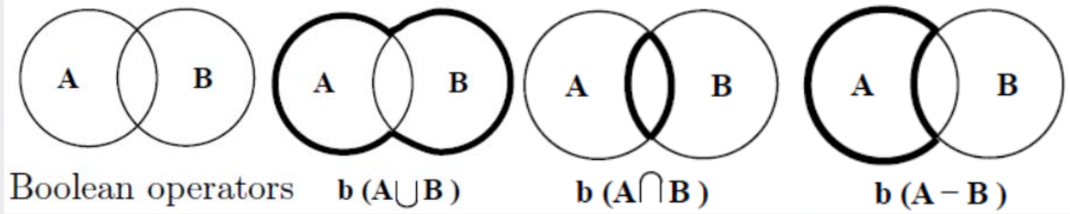
The modeling scheme is cumbersome for ordinary users

Boxed half space geometry

Boxed half space operands





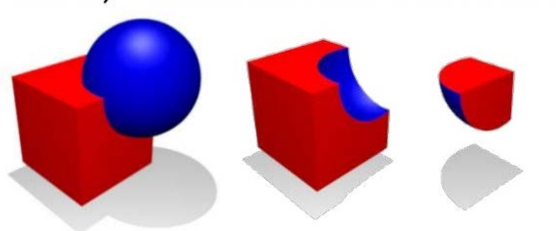


Constructive Solid Geometry

CSG modelers allow designers to combine a set of primitives through Boolean operations:

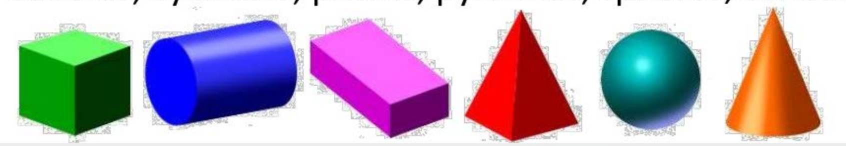
- **Operations**:

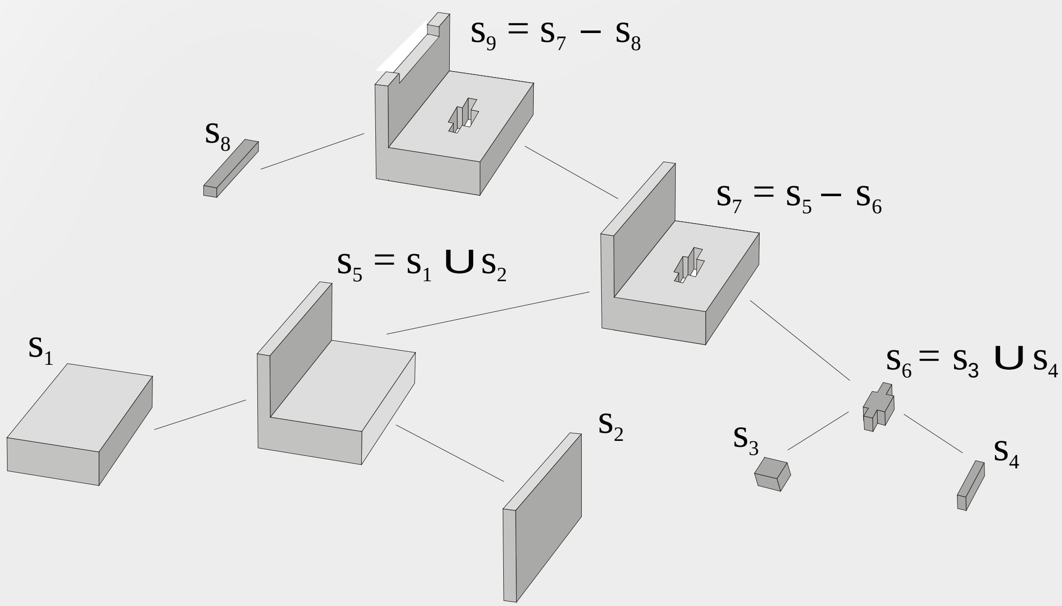
union, intersection and difference.

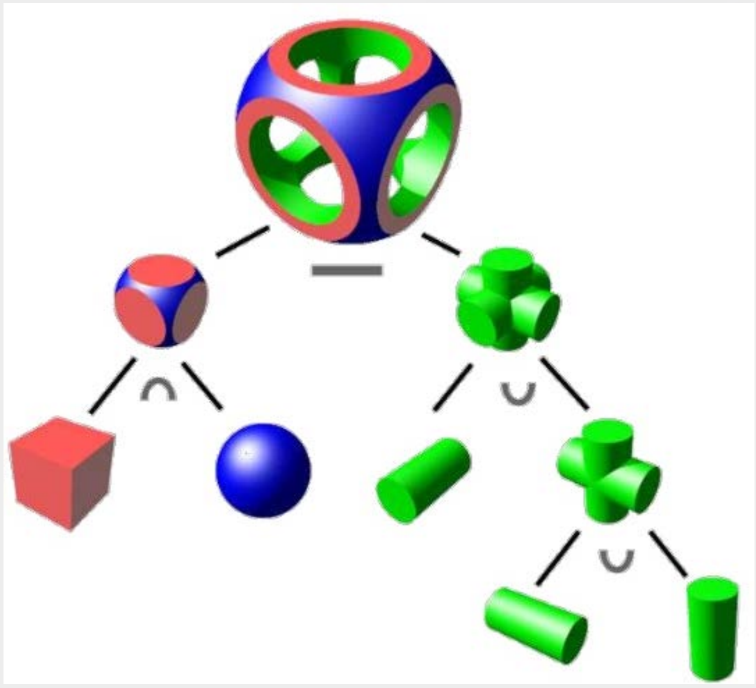


- **Primitives**:

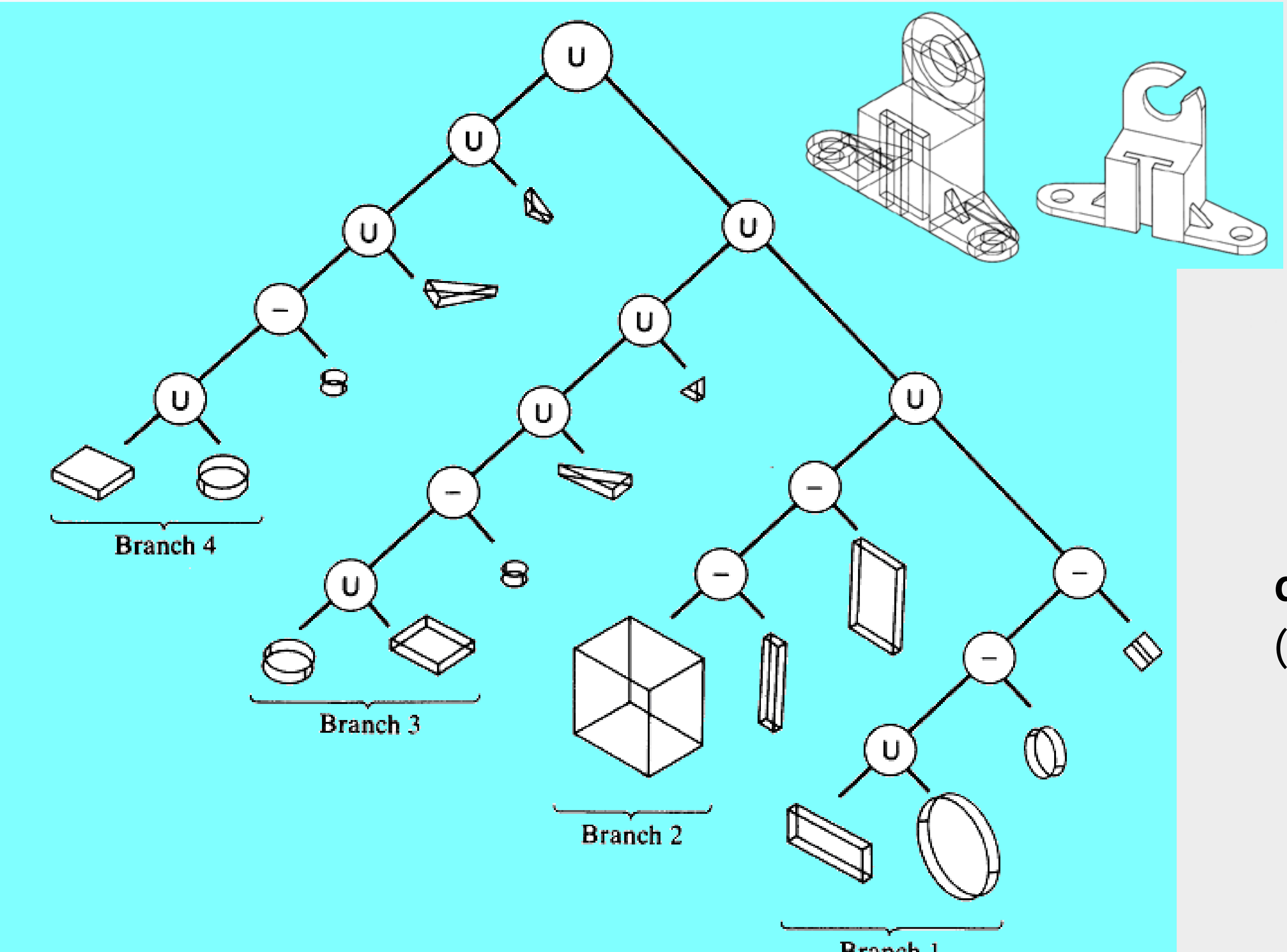
cuboids, cylinders, prisms, pyramids, spheres, cones.



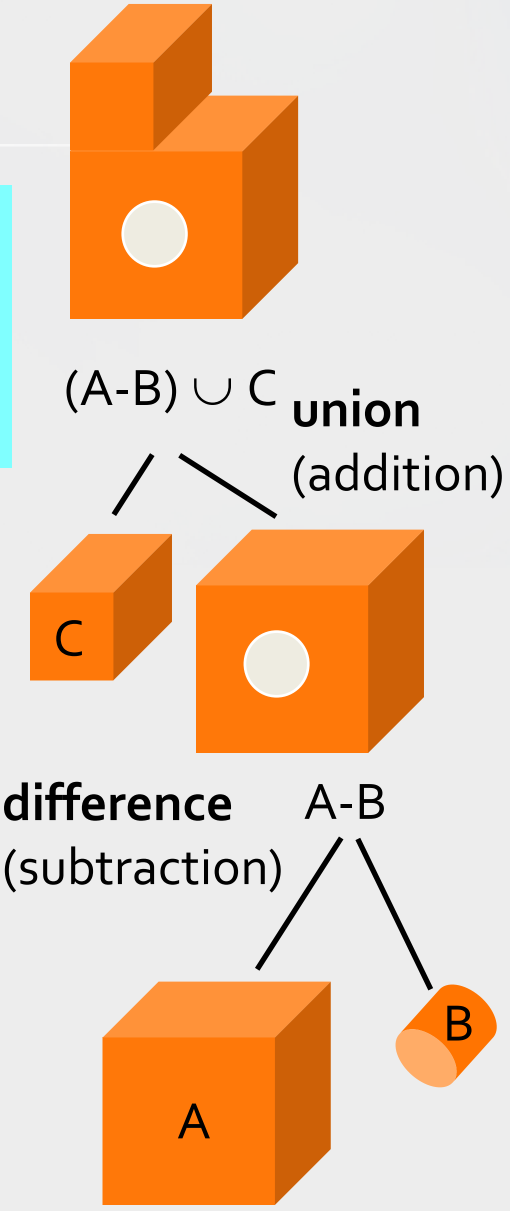




CSG boolean tree Examples



(a) Construction tree

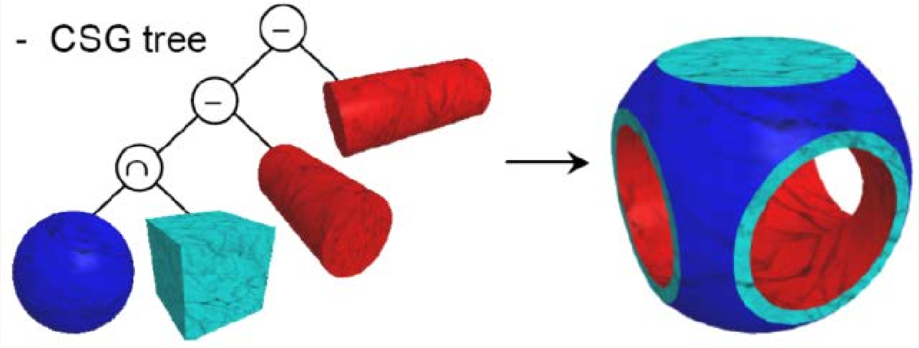


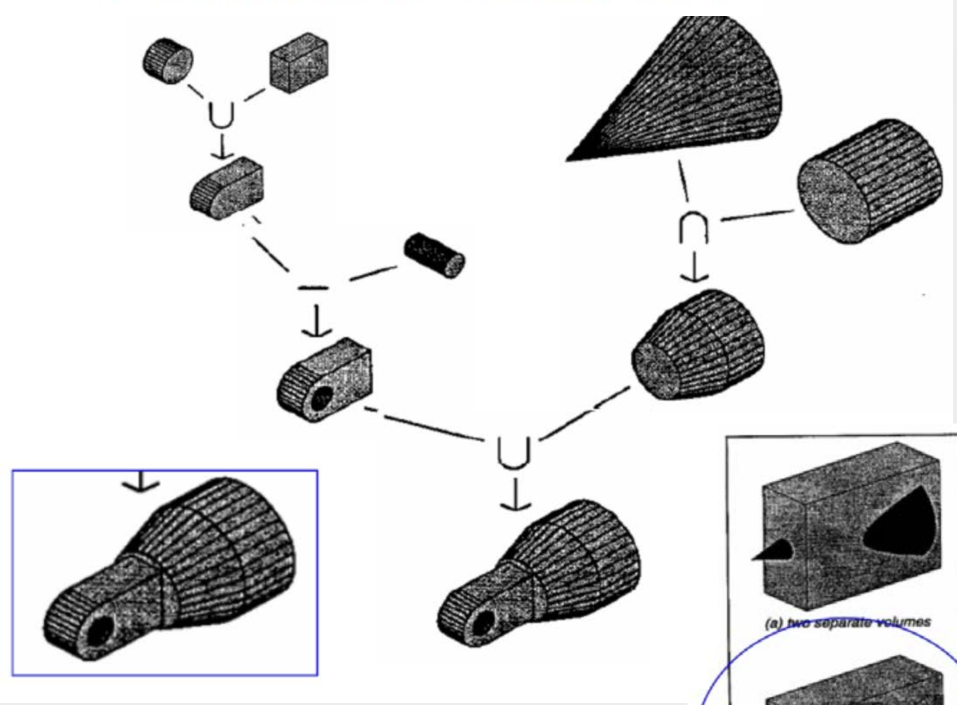
CSG tree examples

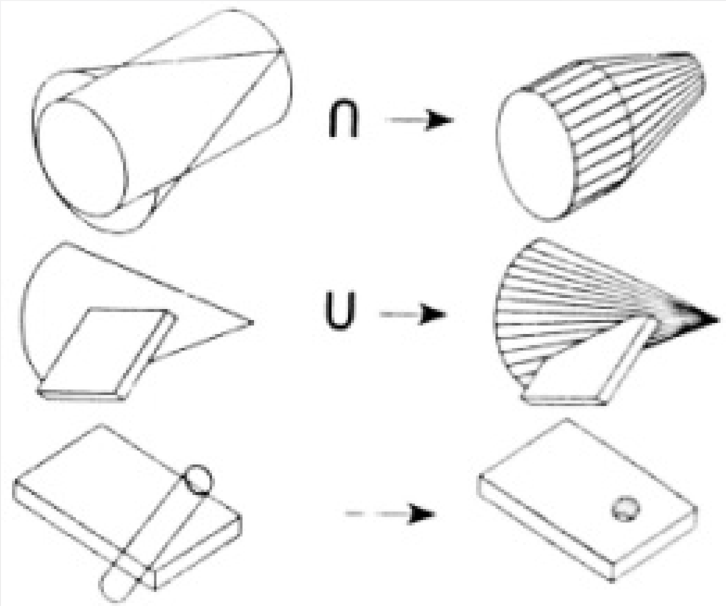
Union, ***U***

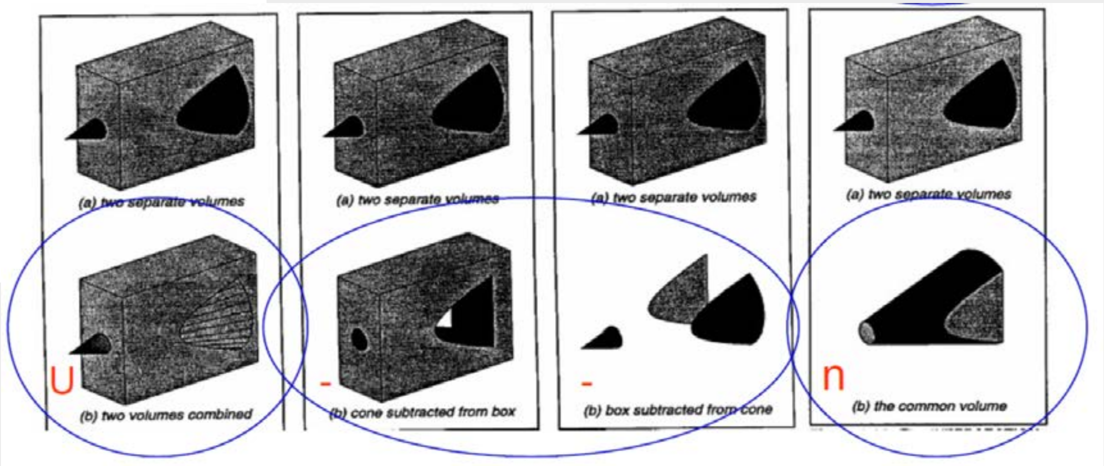
Intersection, n

Difference, or Subtraction, **-**

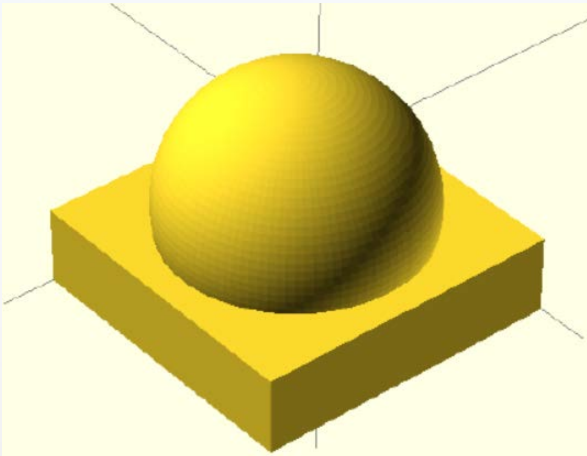




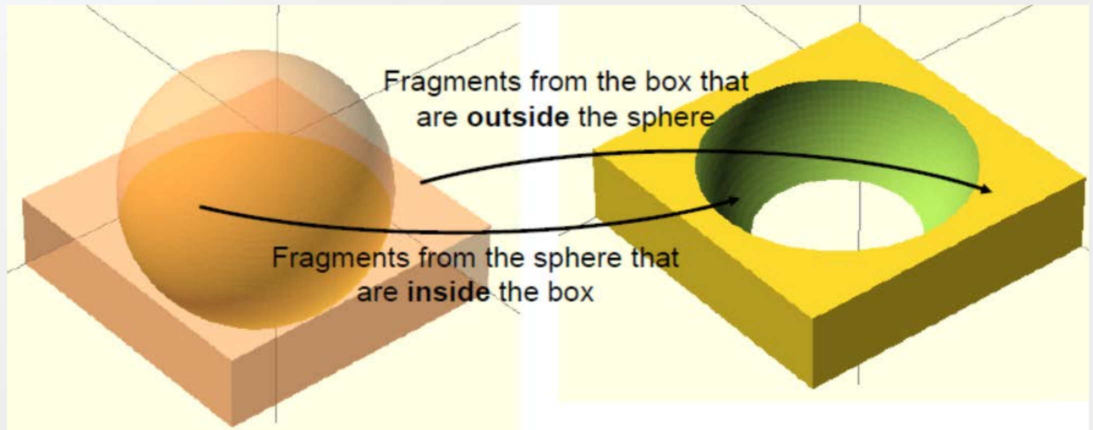




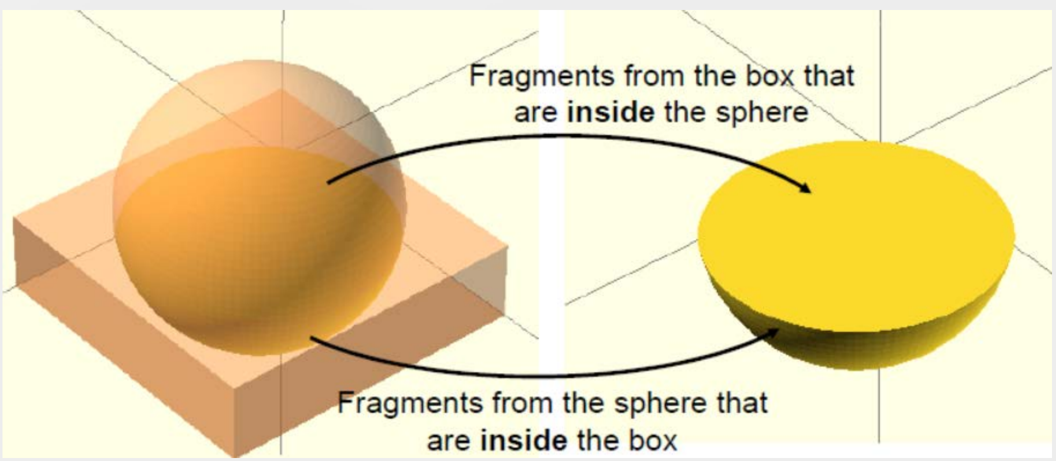
CSG rendering
union



difference



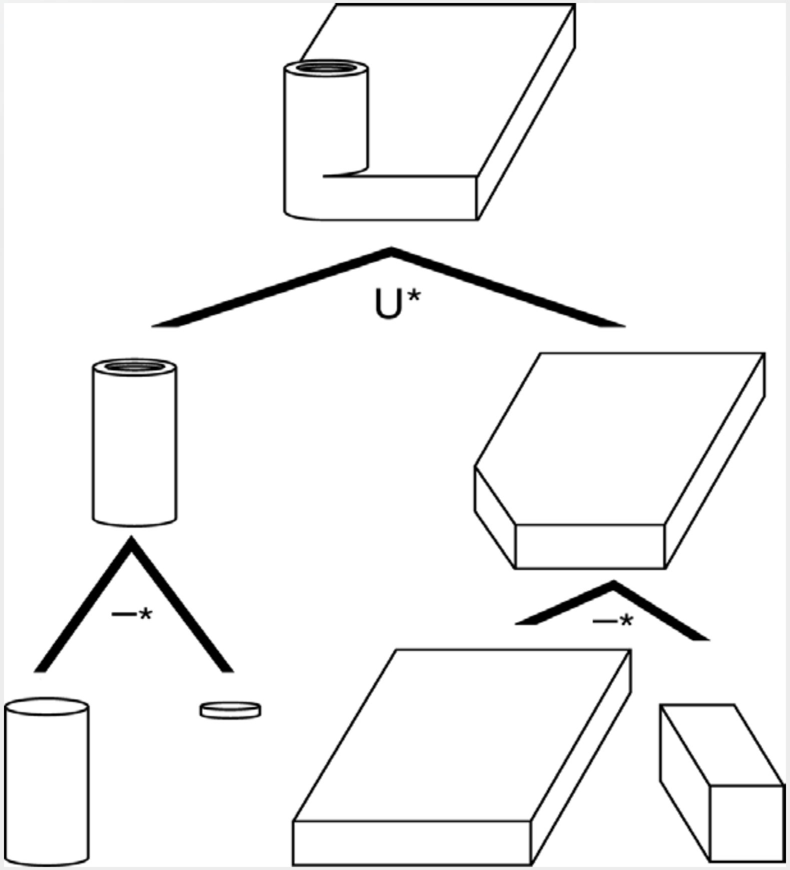
intersection



OpenSCAD demos

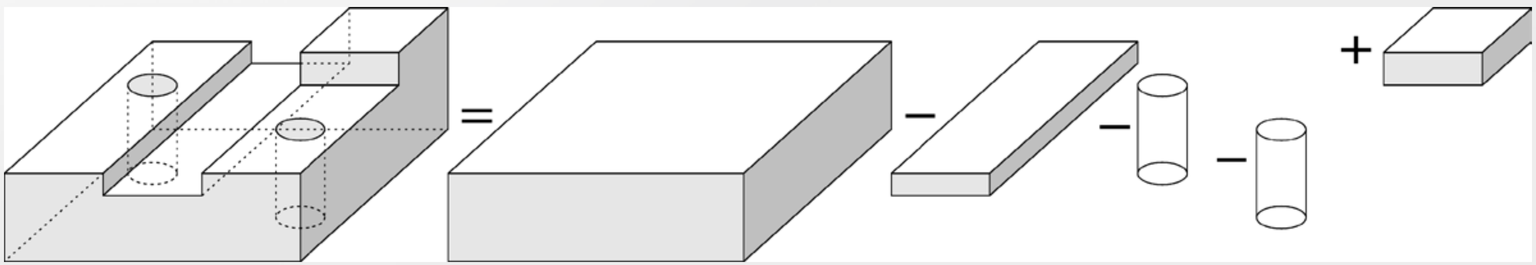
Constructive Solid Geometry (CSG)

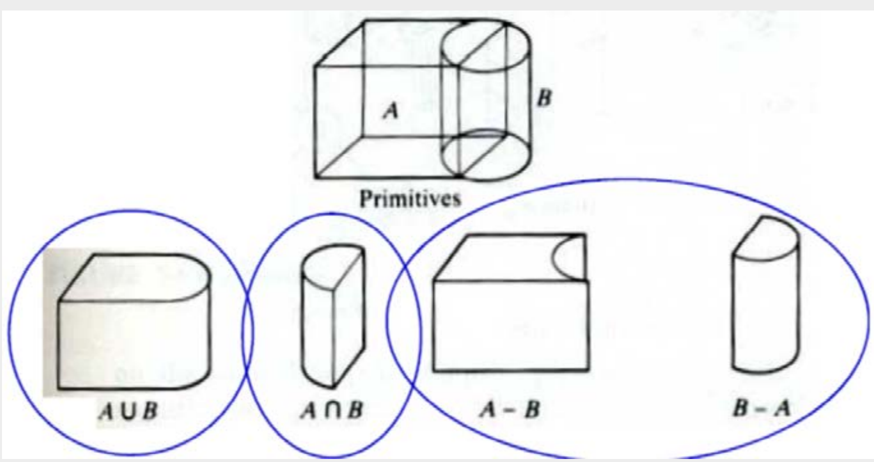
- A tree structure combining primitives via regularized boolean operations
- Primitives can be solids or *half spaces*



A Sequence of Boolean Operations

- Boolean operations
- Rigid transformations

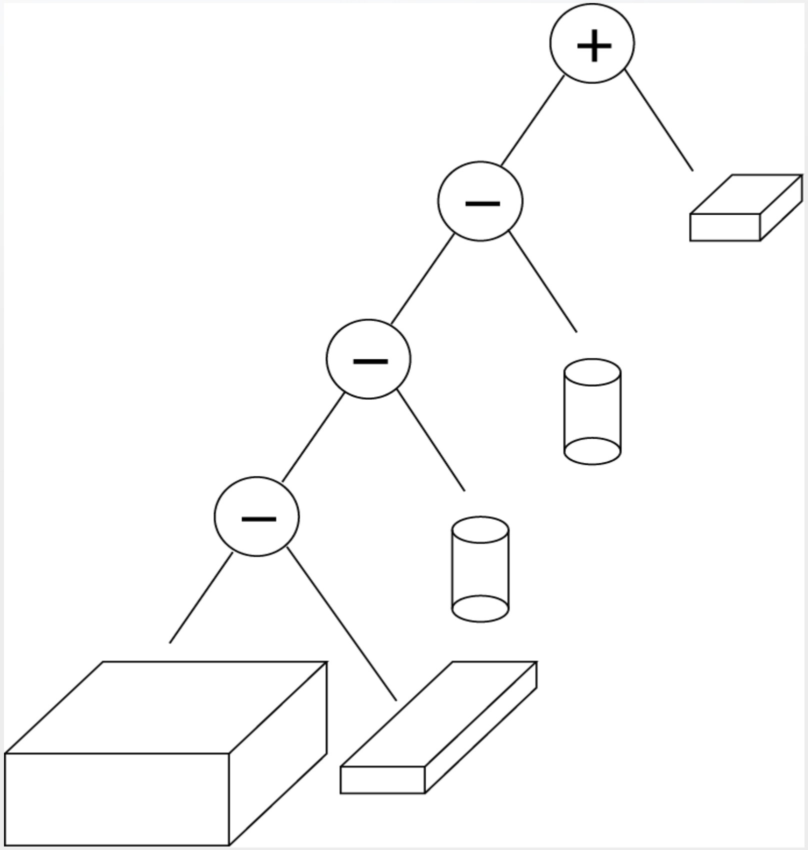


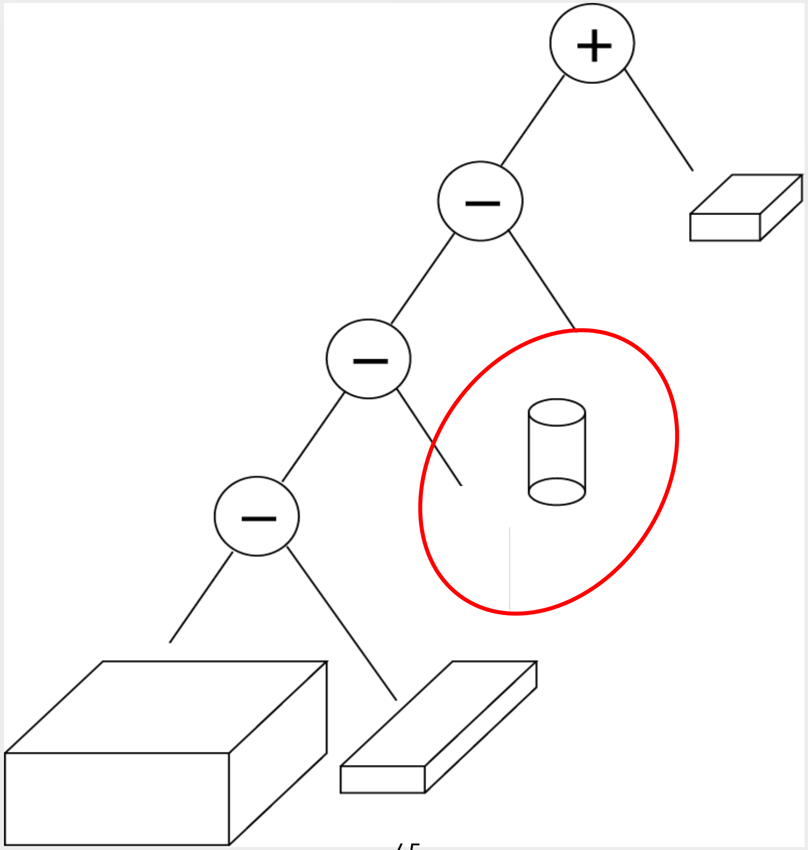


Pics/Math courtesy of Dave Mount @ UMD-CP

The Induced CSG Tree

Can also be represented as a directed acyclic graph (DAG)

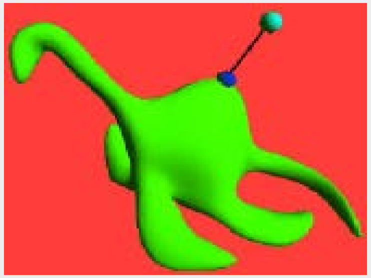




Pics/Math courtesy of Dave Mount @ UMD-CP

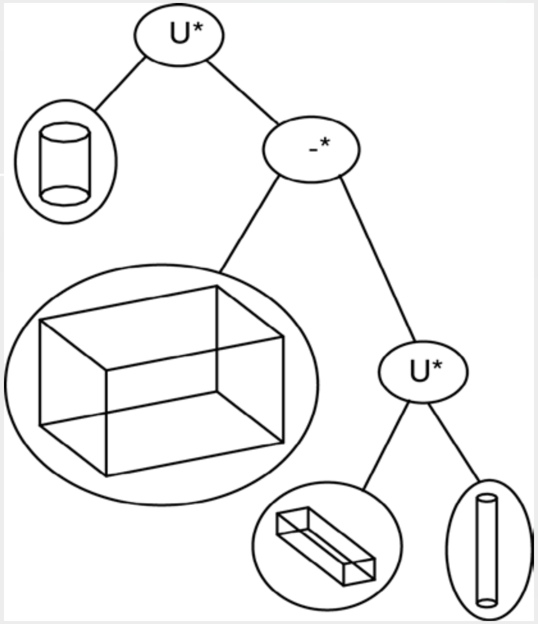
Issues with
Constructive Solid Geometry

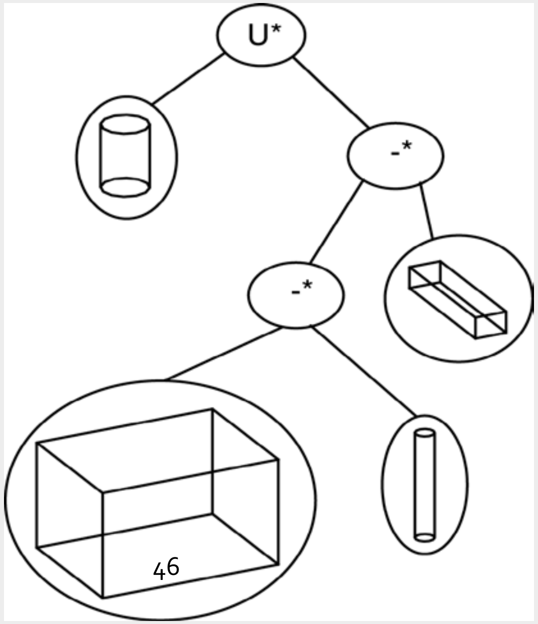
- Non-uniqueness
- Choice of primitives
- How to handle more complex modeling?
  - Sculpted surfaces? Deformable objects?



Non-Uniqueness

There is more than one way to model the same artifact.
Hard to tell if A and B are identical.



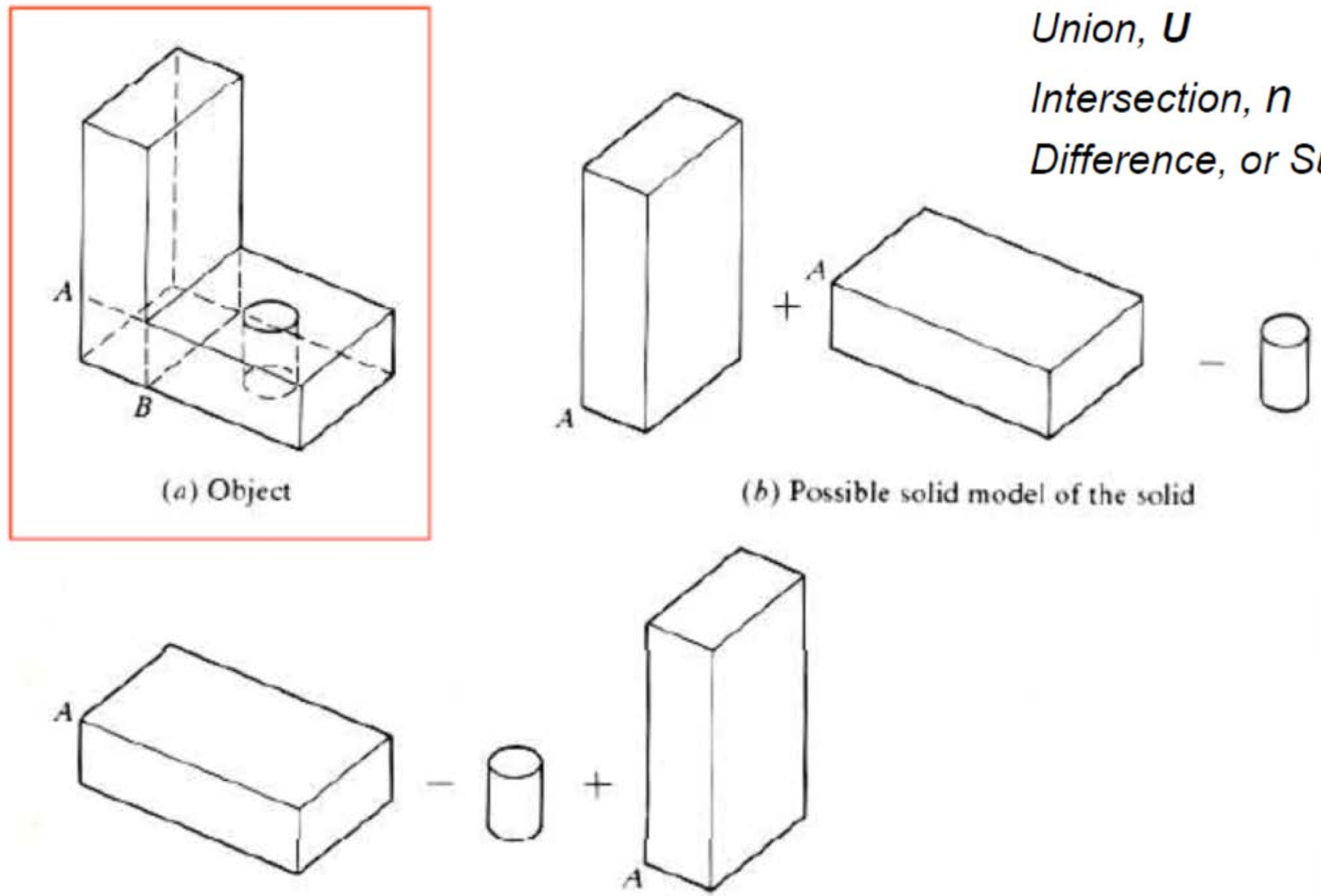


Alternative Paths of Modeling

Union, ***U***

Intersection, n

Difference, or Subtraction, **-**

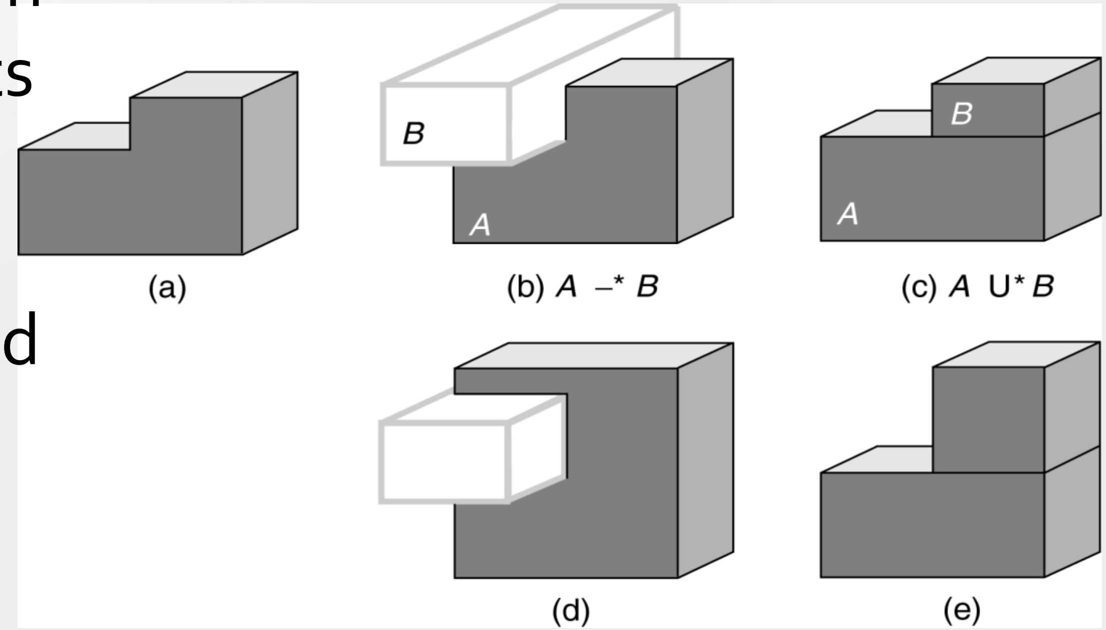


(*a*) Object

(*b*) Possible solid model of the solid

Issues with CSG

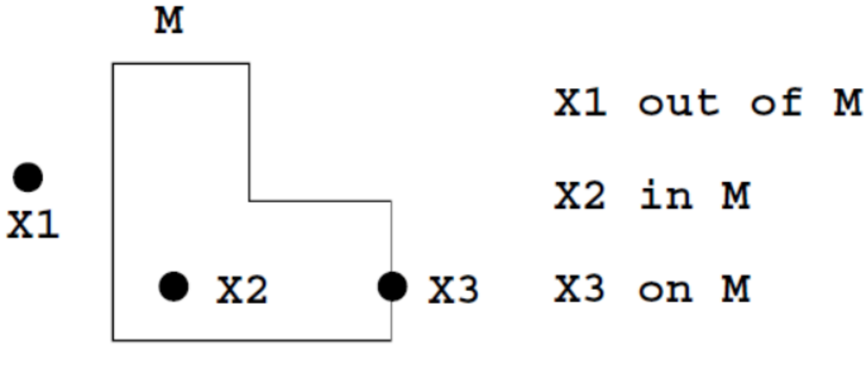
- Minor changes in primitive objects greatly affect outcomes
- Shift up top solid face



Foley/VanDam, 1990/1994

Solid Object Definitions

Solids are point sets : Boundary and interior



Membership classification.

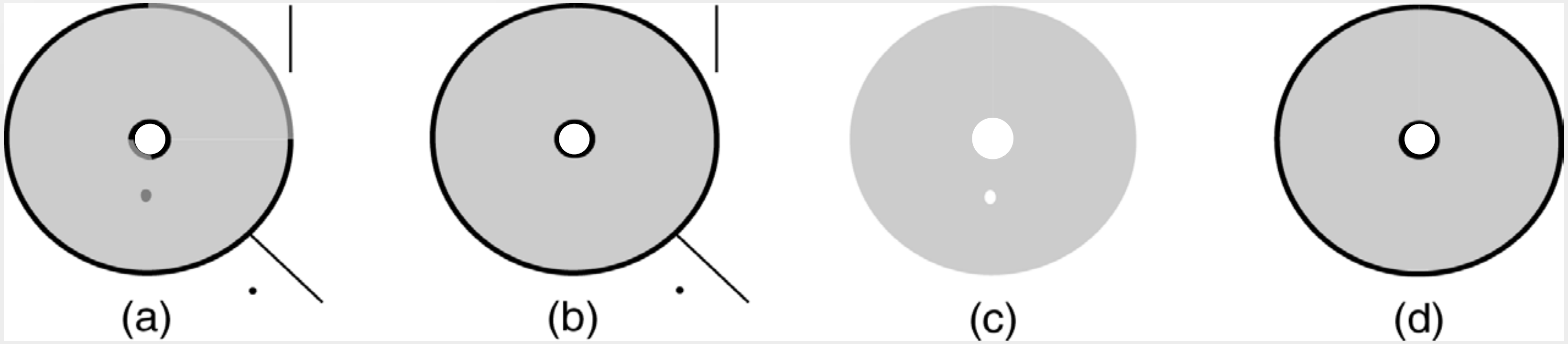
Boundary points
Points where distance to the object and the object's complement is zero

Interior points
All the other points in the object

Closure
Union of interior points and boundary points

Issues with 3D Set Operations

- Ops on 3D objects can create "non-3D objects" or objects with non-uniform dimensions
- Objects need to be "Regularized"
  - Take the closure of the interior

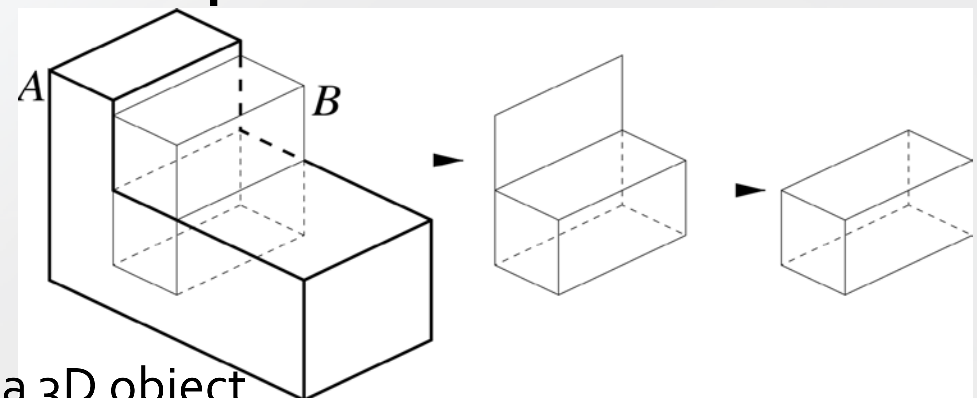


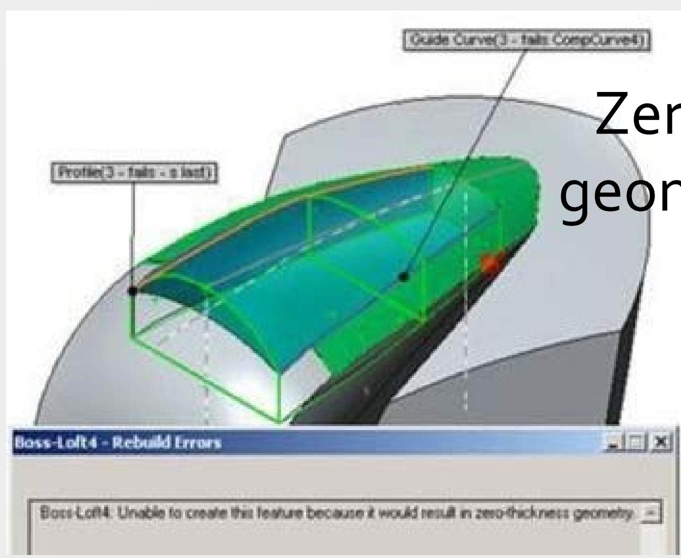
Input set Closure Interior Regularized

Foley/VanDam, 1990/1994

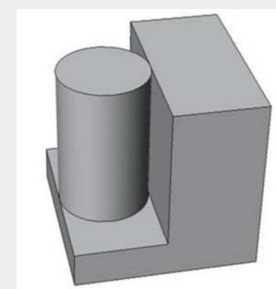
Regularized Boolean Operations

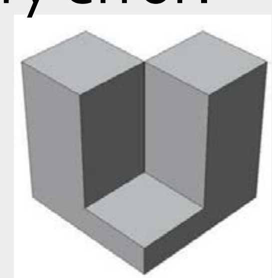
- 3D Example
  - Two solids A and B
  - Intersection leaves a "dangling wall"
    - A 2D portion hanging off a 3D object
  - Closure of interior gives a uniform 3D result

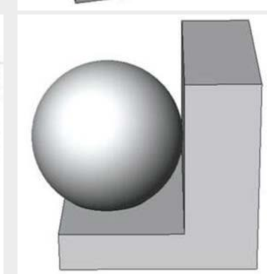




Zero-thickness geometry error?



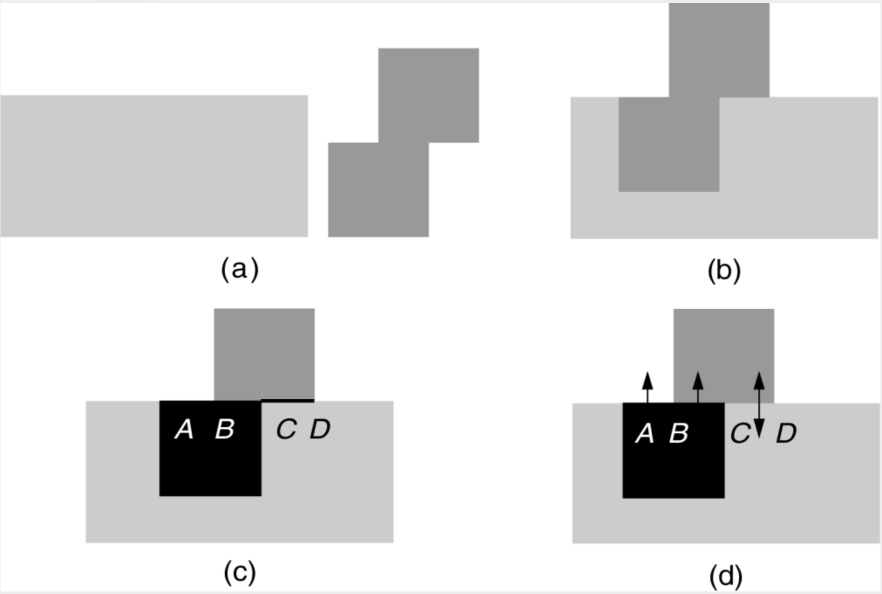




Pics/Math courtesy of Dave Mount @ UMD-CP

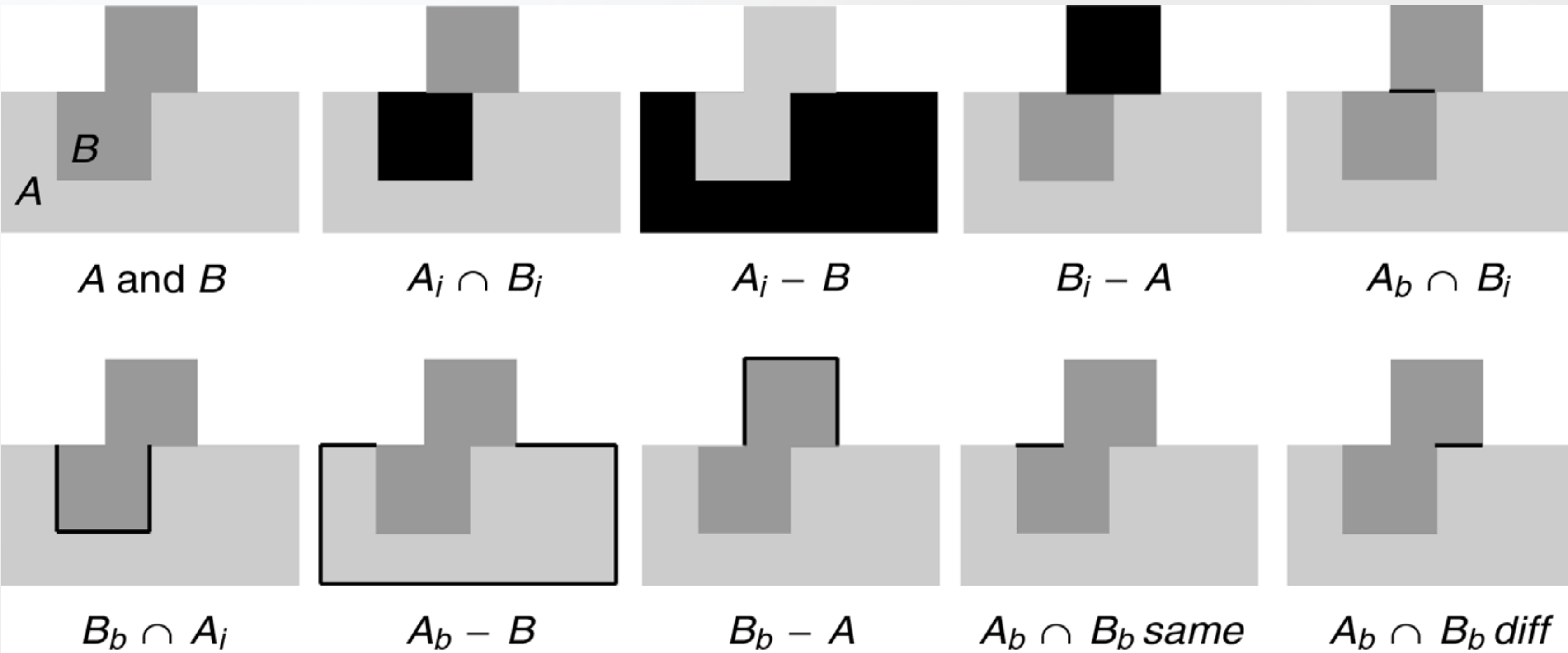
Boolean Operations

- Other Examples:
- (c) ordinary intersection
- (d) regularized intersection
  - AB - objects on the same side
  - CD - objects on different sides



Foley/VanDam, 1990/1994

Boolean Operations



Foley/VanDam, 1990/1994

CSG Building Operations

The main building operations are regularized set operations like **union** (U*), **intersection** (∩*) and **difference** (-*).

Hence the CSG models are known as **set-theoretic**, **boolean** or **combinatorial** models.

The Boolean operations are based on the set theory and the closure property. These operations are considered higher-level operations than B-rep Euler operations.

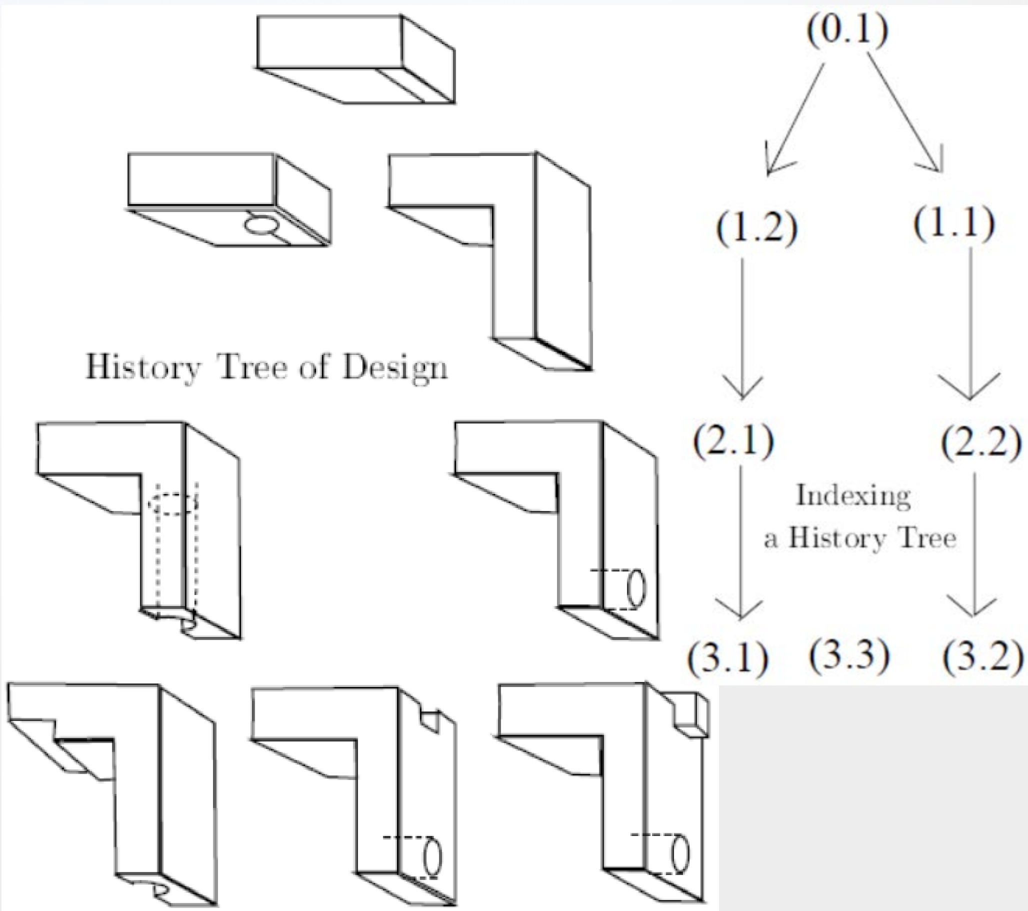
Some implementations of solid modelers provide derived types of operations like **ASSEMBLE** and **GLUE**

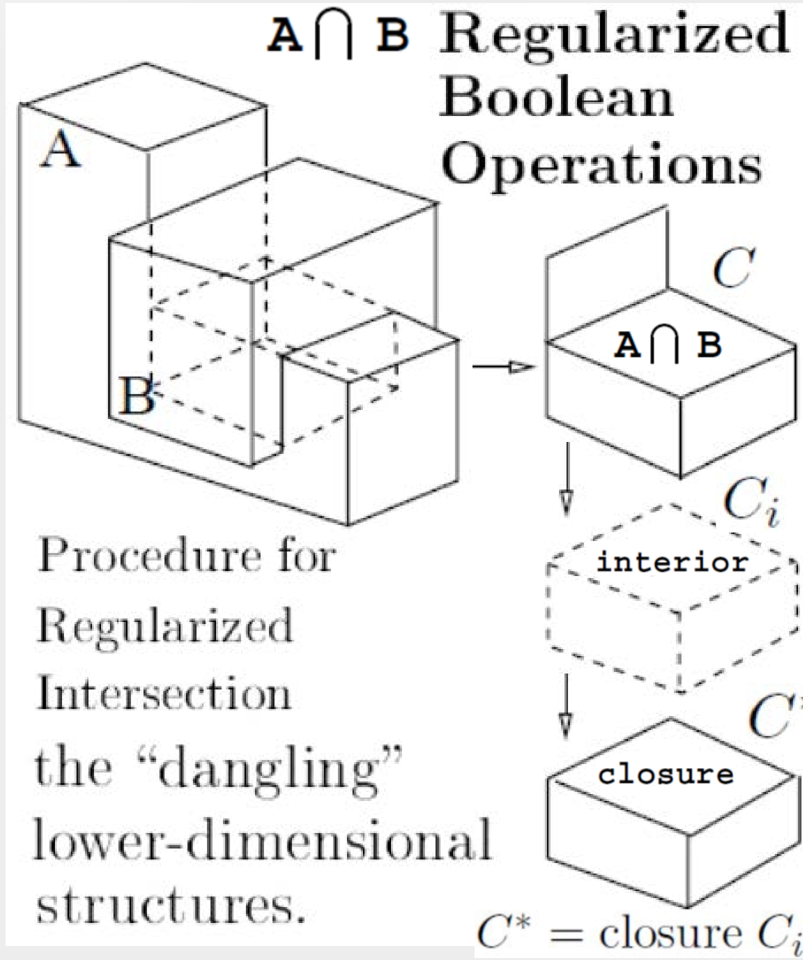
CSG History Tree of Design

$\cap^*$ regularized set operation

1. $C = A \cap B$
2. $C_i = \text{interior } C$
3. $C^* = \text{closure } C_i$

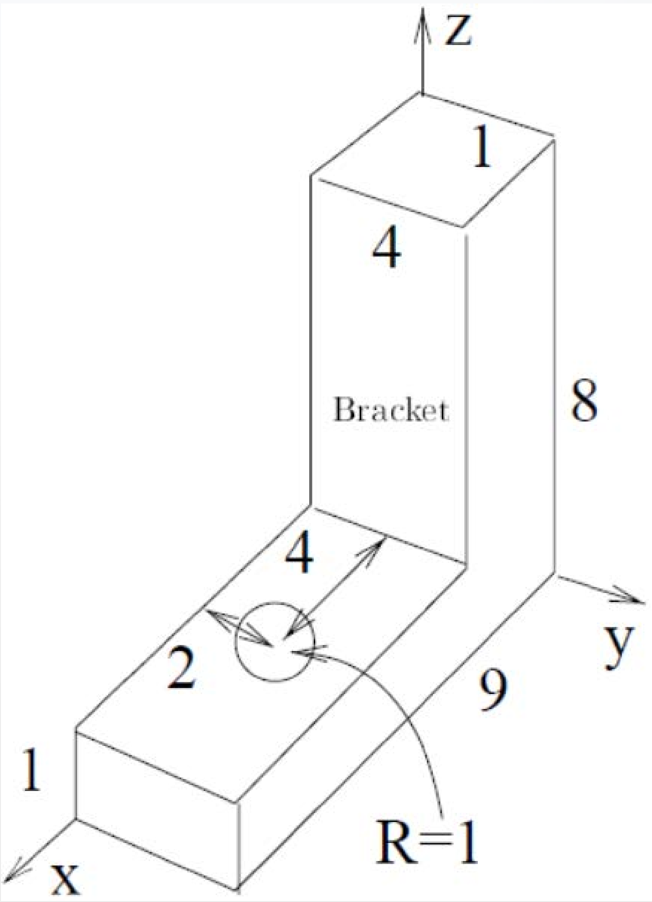
and $C^* = A \cap^* B$

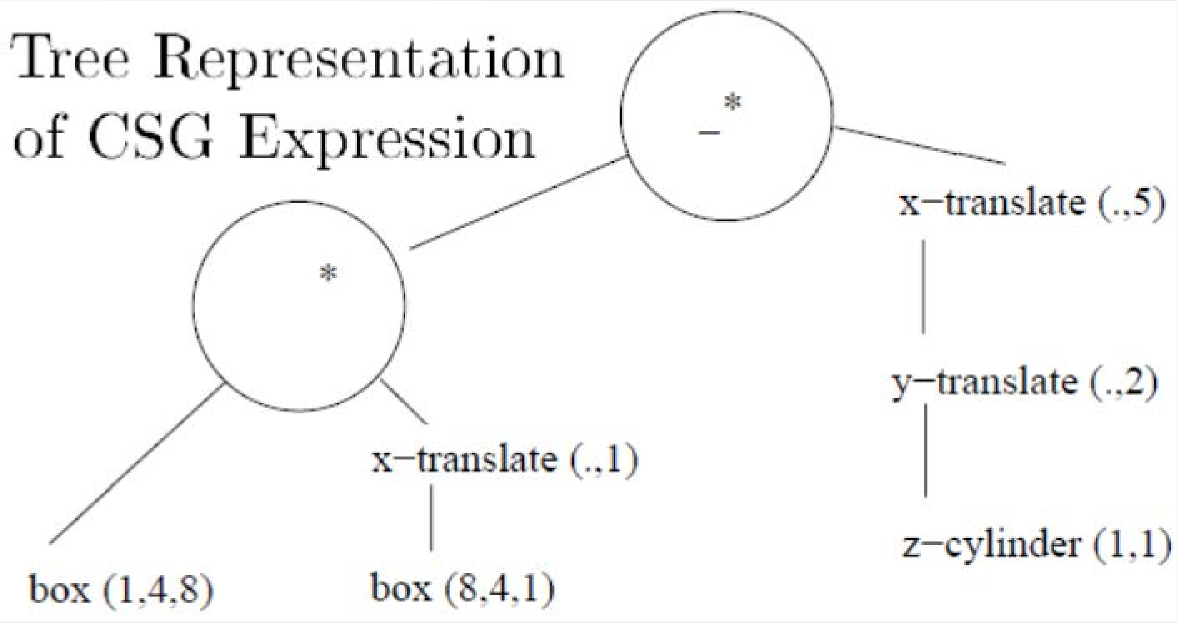




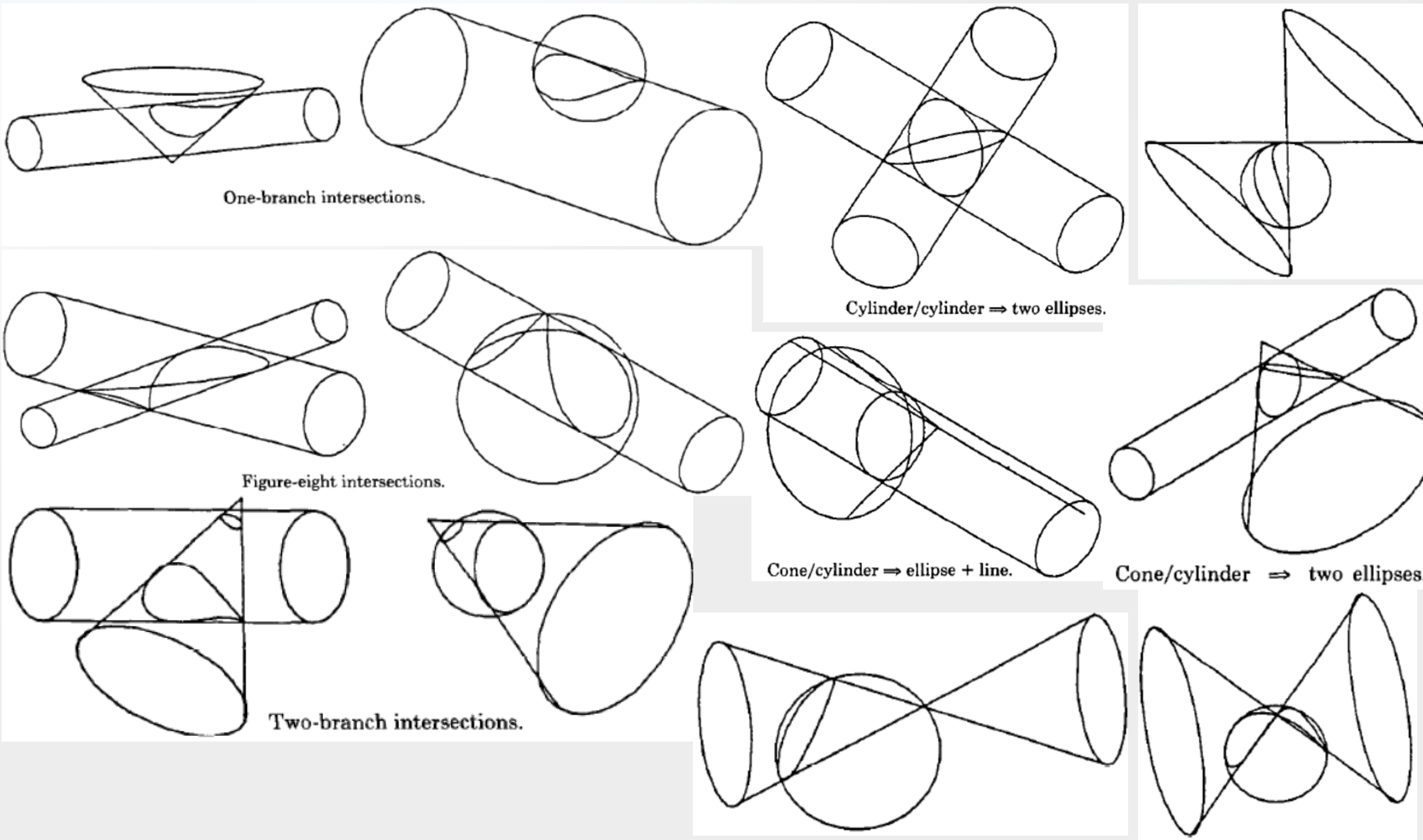
$C = A \cap B$ $\quad$ $C^* = A \cap^* B$

CSG History Tree of Design





Quadric Surface Intersection Curves



Main algorithms in CSG Operations

1. Edge / Solid intersection algorithm

2. Computing set membership classification

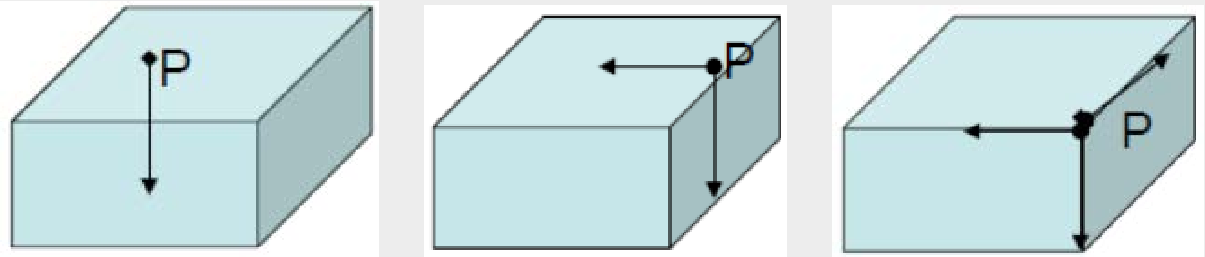
a) Divide and conquer : It is like ray tracing. Instead of a ray an edge is used as a reference

b) Neighborhood : It deals with in, on and out decisions

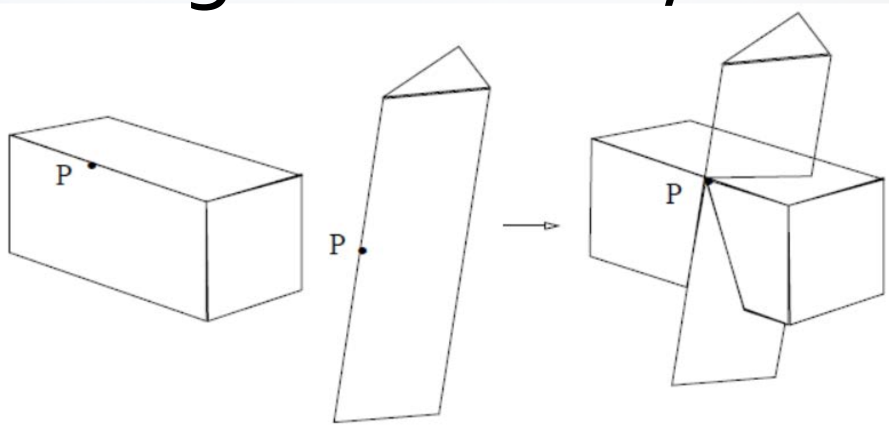
When a point is in the interior of solid face then it is called **face** neighborhood

Edge neighborhood occurs when the point lies on the solid edge

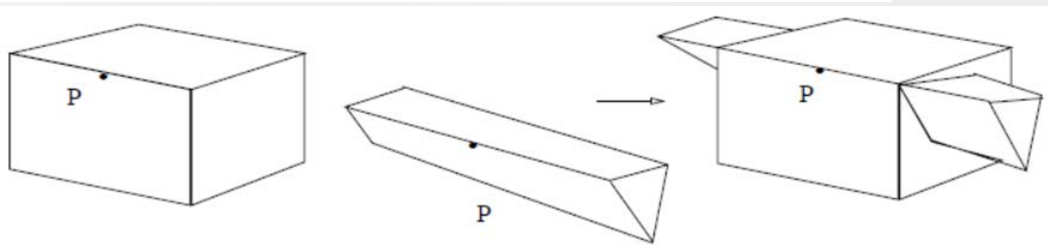
When a point is a vertex, **vertex** neighborhood occurs. This is a complex case because the point is shared between three solid faces.



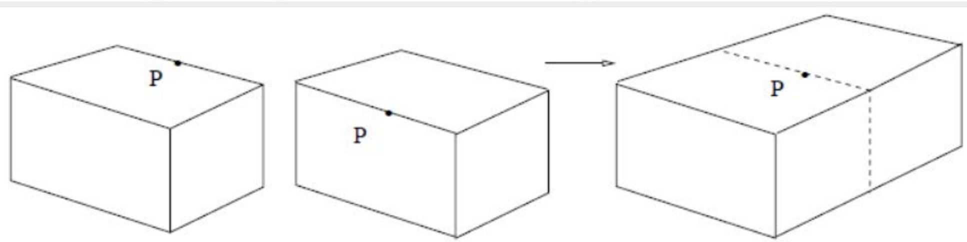
Neighborhoods, vertex, edge, face merge



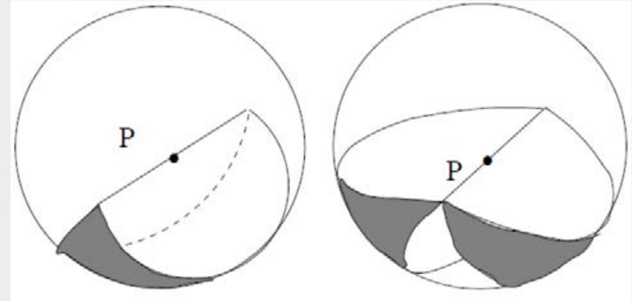
Edge-Neighborhood Merge, General Position



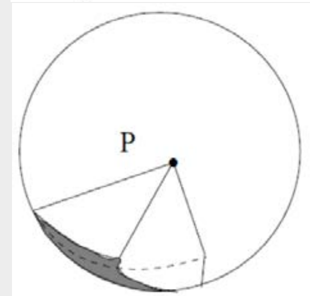
Edge-Neighborhood Merge Producing an Edge



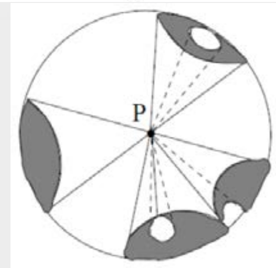
Edge-Neighborhood Merge Producing a Face



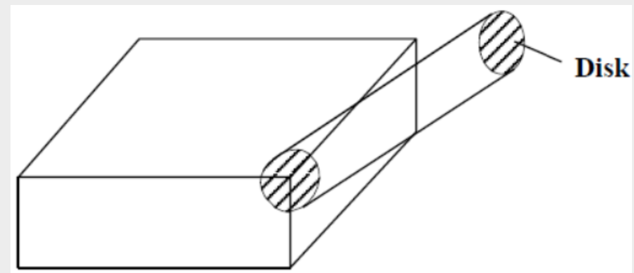
Neighborhood of an Interior Edge Point



Simple Neighborhoods

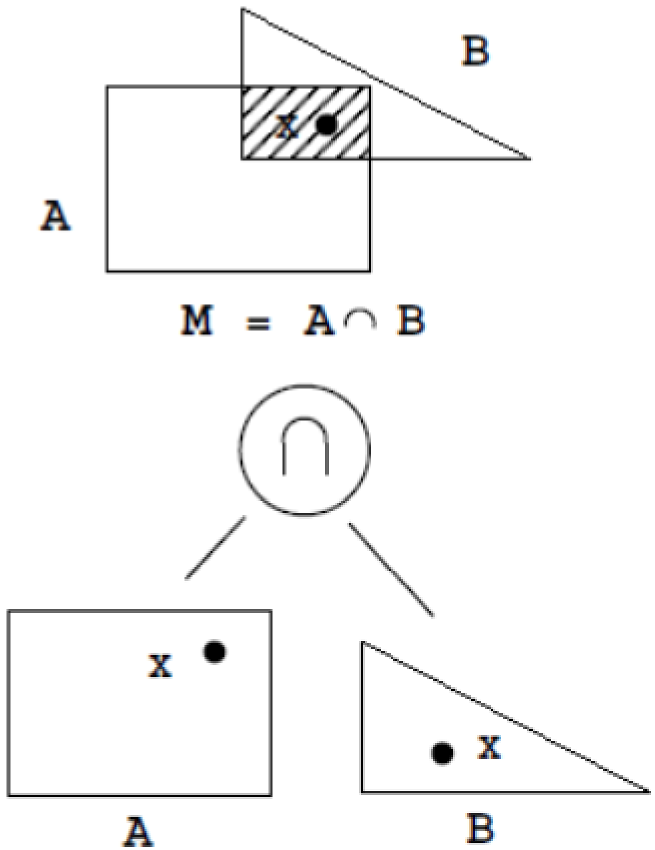


General Neighborhood of a Vertex

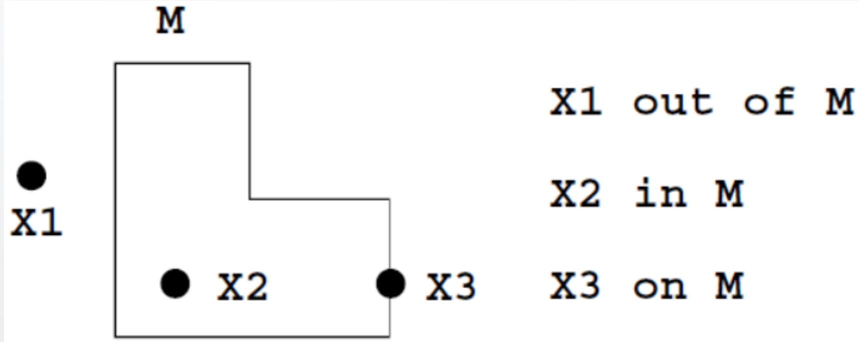


Neighborhood of point on two-manifold object is a disk.

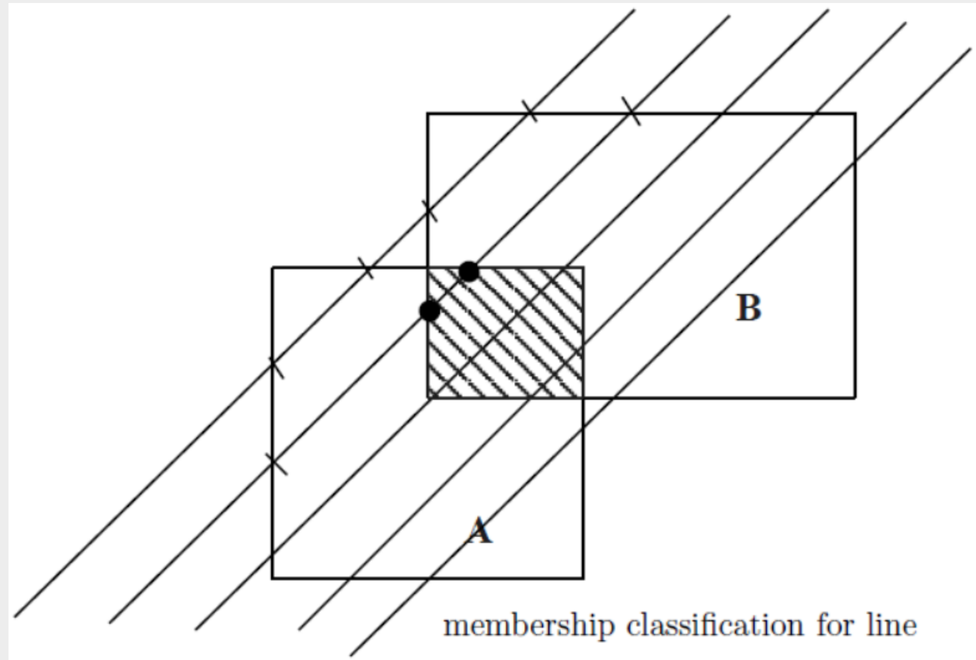
Membership classification



membership classification for point

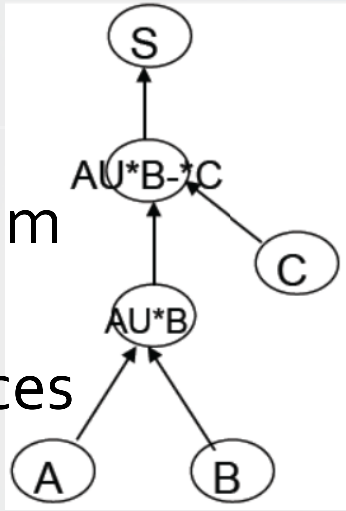


Membership classification.



membership classification for line

Summary of a CSG algorithm



The following steps describe a general CSG algorithm based on divide and conquer (D & C) approach:

1. Generate a sufficient number of t-faces, set of faces of participating primitives, say A and B.

2. Classify self edges of A w.r.t A including neighborhood.

3. Classify self edges of A w.r.t B using D & C paradigm.
If A or B is not primitive then this step is followed recursively.

4. Combine the classifications in step 2 and 3 via Boolean operations.

5. Regularize the ‘on’ segment that result from step 4 discarding the segments that belong to only one face of S.

Summary of a CSG algorithm

6. Store the final ‘on’ segments that result from step 5 as part of the boundary of S. Steps 2 to 6 is performed for each of t-edge of a given t-face of A.

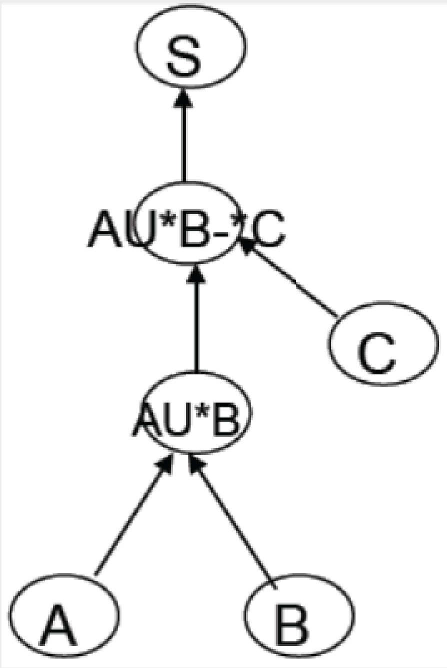
7. Utilize the surface/surface intersection to find cross edges that result from intersecting faces of B (one at a time) with the same t-face mentioned in step 6.

8. Classify each cross edge w.r.t S by repeating steps 2 to 4 with the next self edge of A.

9. Repeat steps 5 and 6 for each cross edge

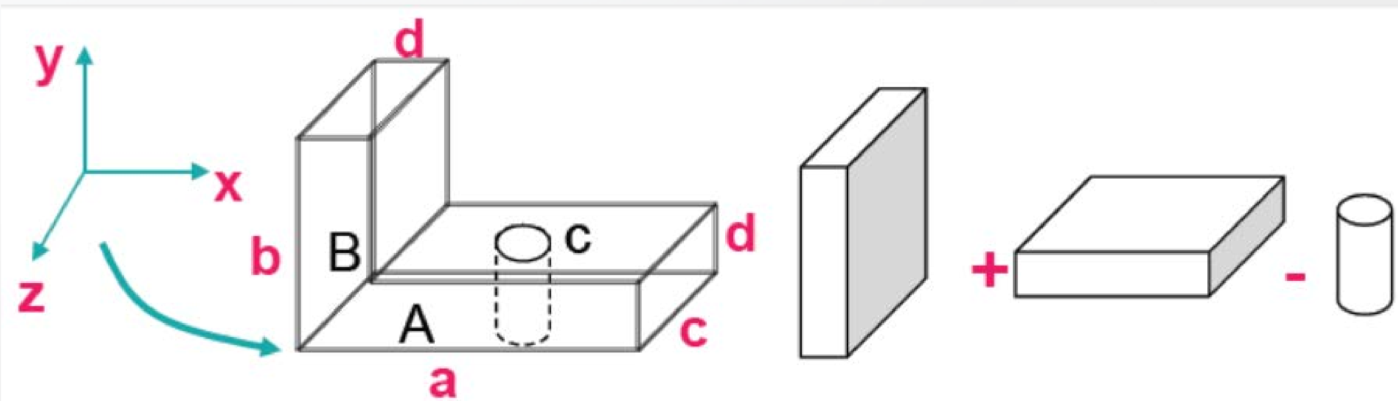
10. Repeat steps 2 to 9 for each t-face of A.

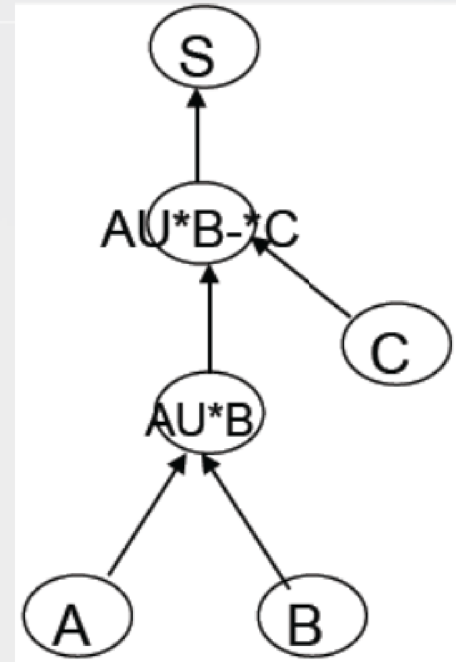
11. Repeat steps 2 to 6 for each t-face of B.



A CSG Example

Create the CSG model of the following solid





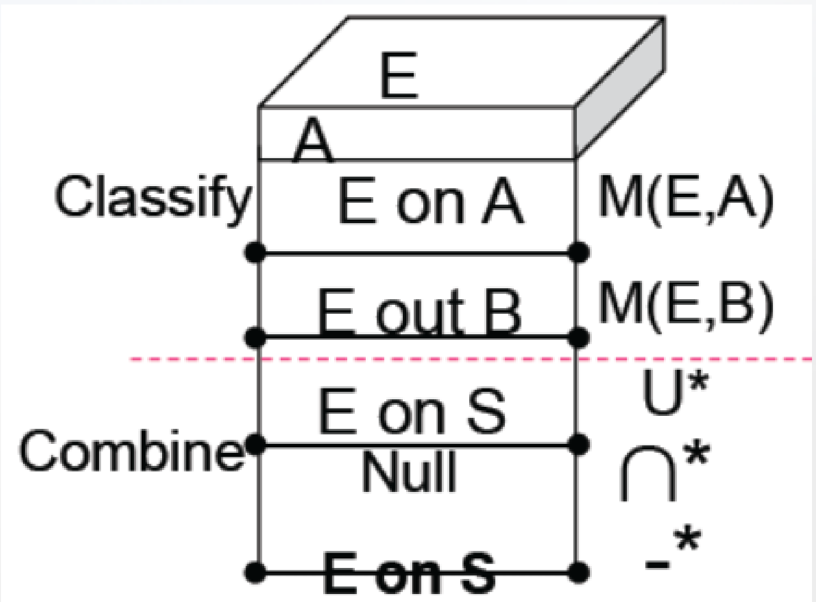
Geometry of the primitives

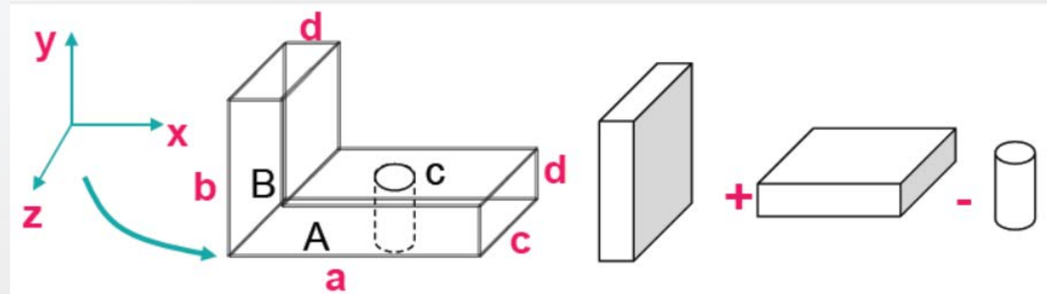
BLOCK A: $x_L = a - d, y_L = d, z_L = c, P_A(x, y, z) = P_A(d, 0, -c)$

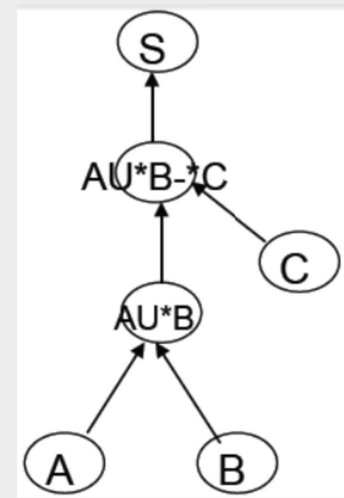
BLOCK B: $x_L = d, y_L = b, z_L = c, P_B(x, y, z) = P_B(0, 0, -c)$

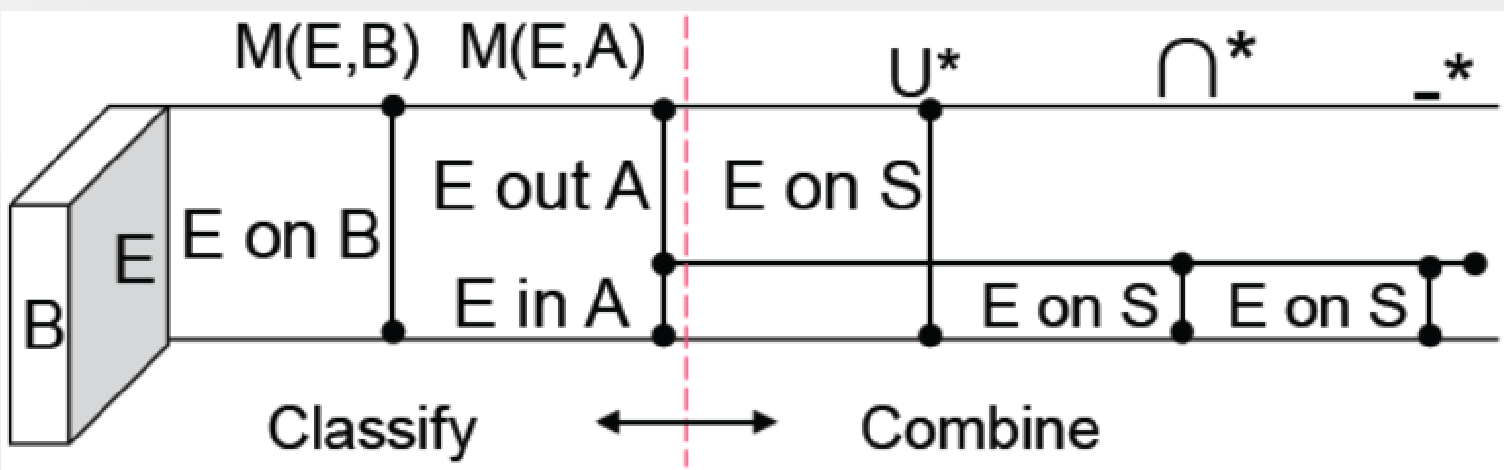
CYLINDER C: $R = R, H = d, \ P_C(x, y, z) = P_C(d + a/2, d, -c/2)$

CSG example, Regularized set operations
Neighborhoods, Memberships

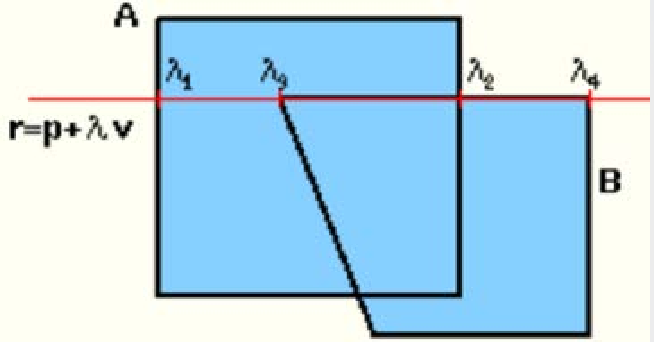








Point-inside-solid test (for CSG)




```
function classify(P:point, n:nodeCSG) return InOnOut
   if isLeaf(n) then
      case (n.type)
         Box: r:=classifyBox(P,n)
         Cylinder: r:=classifyCylinder(P,n)
         Sphere: r:=classifySphere(P,n)
          ...
   else
          rA:= classify (P, n.left)
          rB:= classify (P, n.right)
          r:= combine(n.operation, rA, rB)
   end
    return r
end
```

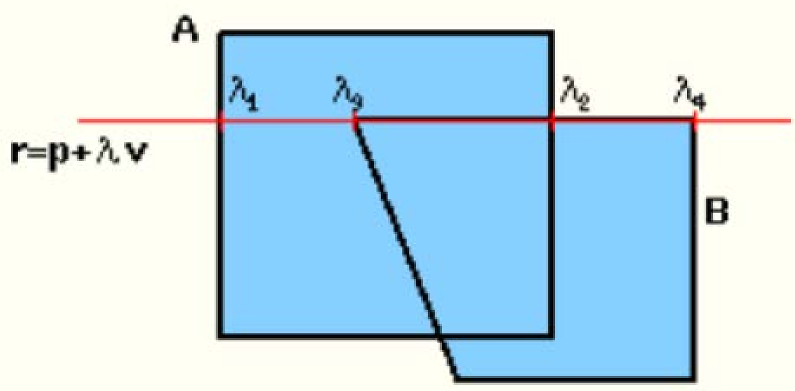
Combina(op, A, B)

AUB	in	on	out
in	in	in	in
on	in	on	on
out	in	on	out

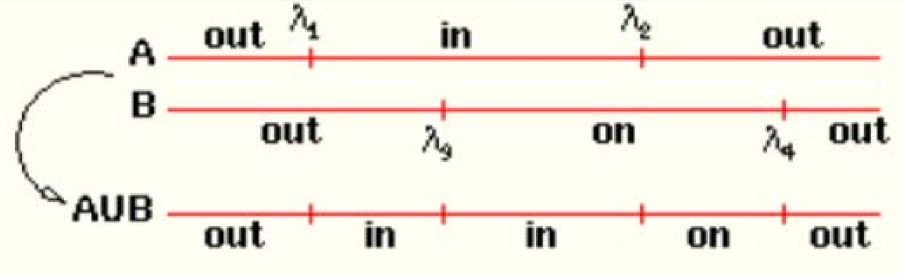
A^B	in	on	out
in	in	on	out
on	on	on	out
out	out	out	out

A-B	in	on	out
in	out	on	in
on	out	on	on
out	out	out	out

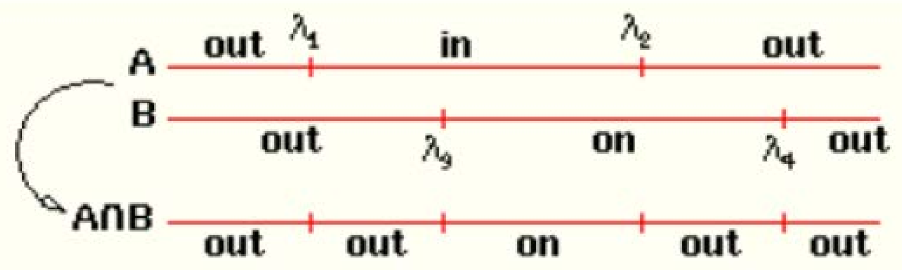
Line-solid classification



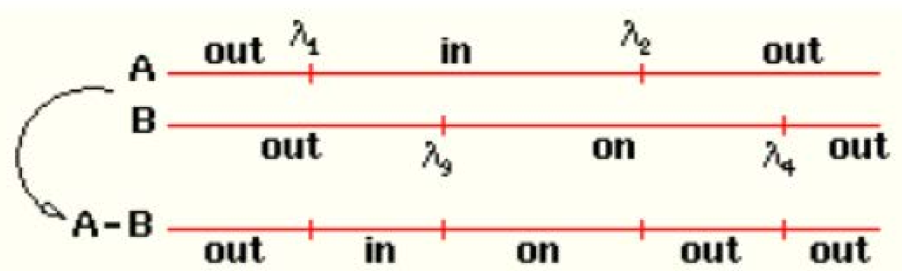
Sòlid A : [l1,in] [l2,out]
Sòlid B : [l3,on] [l4,out]



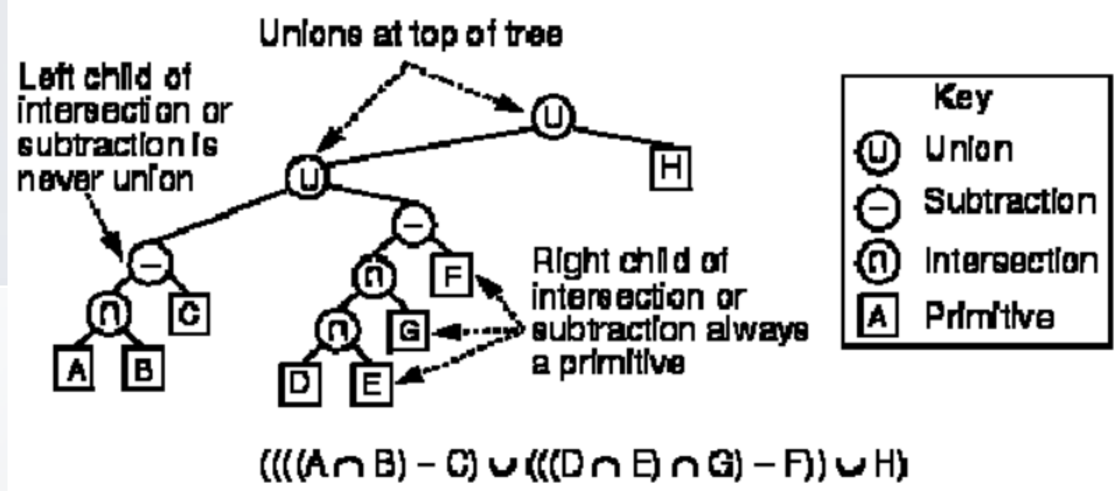
Resultat de la unió : [l1,in] [l2,on] [l4,out]
(s'han hagut de compactar dos intèrvals "in")

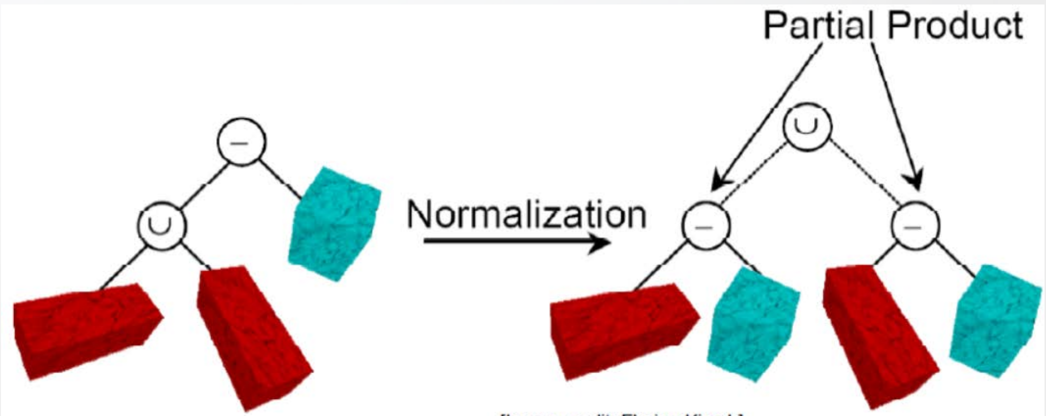


Resultat de la intersecció : [l1,out] [l3,on] [l2,out]
(s'han hagut de compactar dos intèrvals "out")



CSG Normalization





[Image credit: Florian Kirsch]

1. $X - (Y \cup Z) \rightarrow (X - Y) - Z$
2. $X \cap (Y \cup Z) \rightarrow (X \cap Y) \cup (X \cap Z)$
3. $X - (Y \cap Z) \rightarrow (X - Y) \cup (X - Z)$
4. $X \cap (Y \cap Z) \rightarrow (X \cap Y) \cap Z$
5. $X - (Y - Z) \rightarrow (X - Y) \cup (X \cap Z)$
6. $X \cap (Y - Z) \rightarrow (X \cap Y) - Z$
7. $(X - Y) \cap Z \rightarrow (X \cap Z) - Y$
8. $(X \cup Y) - Z \rightarrow (X - Z) \cup (Y - Z)$
9. $(X \cup Y) \cap Z \rightarrow (X \cap Z) \cup (Y \cap Z)$

Push unions towards the root

Properties of CSG models

Advantages:

validity: CSG model is always valid;

conciseness: CSG tree is in principle concise;

computational ease: primitives are easy to handle;

unambiguity: every CSG tree unambiguously models a rigid solid.

Disadvantages:

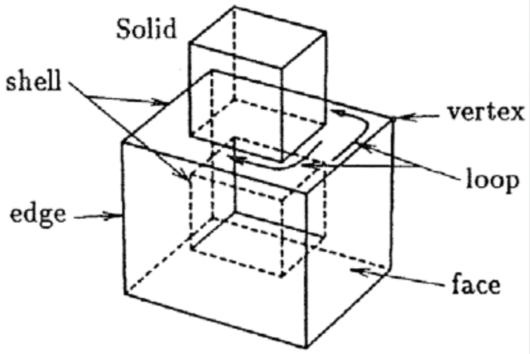
non-uniqueness: a solid could have more than one representation.

limit on primitives: free-form surfaces are excluded, and primitives are bounded by simple low order algebraic surfaces.

redundancy of CSG tree: it may have redundant primitives in tree.

no explicit boundary surface information: CSG needs to be evaluated.

Boundary Representation
B-Rep Solid Modeling



Boundary representation, B-rep is that a 3D object model is enclosed by surfaces (faces) and has its own interior and exterior. It describes the shape as a collection of surfaces which seperate its interior from the external environment. It is suitable for complex designs, Polygon facets are one of the examples of boundary representation. Both polyhedra and curved objects can be modeled using the following topological primitive entities.

Vertex : It is a point where two or more edges meet with another.

Edge : It is a line or curve enclosed between two vertices.

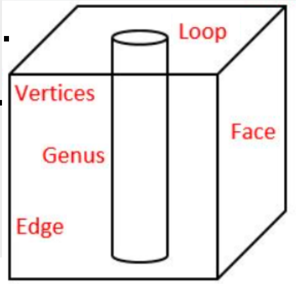
Fin : A fin represents the oriented use of an edge by a loop.

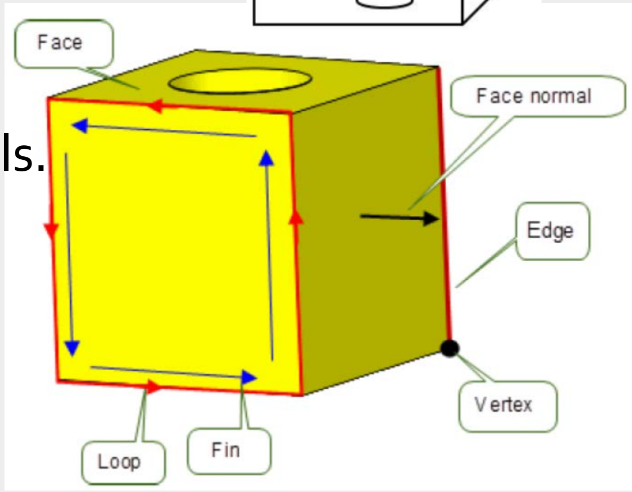
Loop : It is a hole in a face.

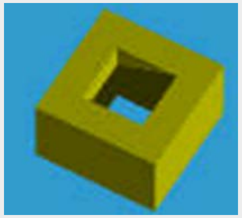
Face : It is a surface or plane of a solid.

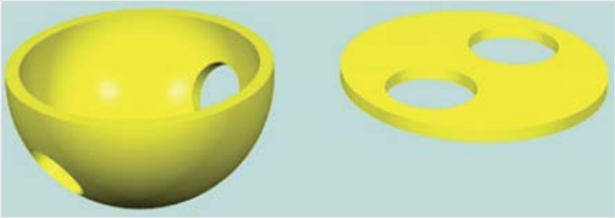
Body : It is an independent solid and has seperate shells.

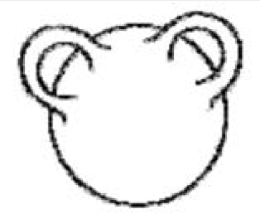
Genus : It is a through hole (handle) in a solid.





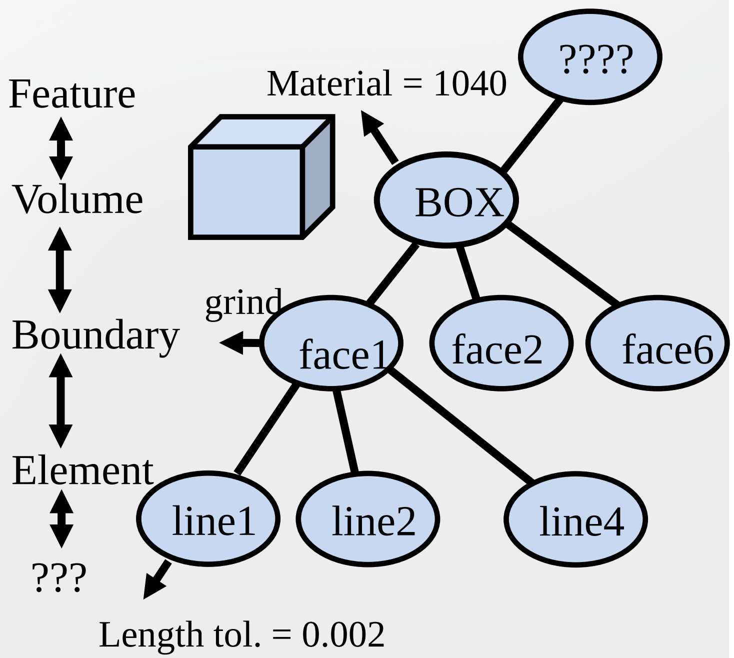


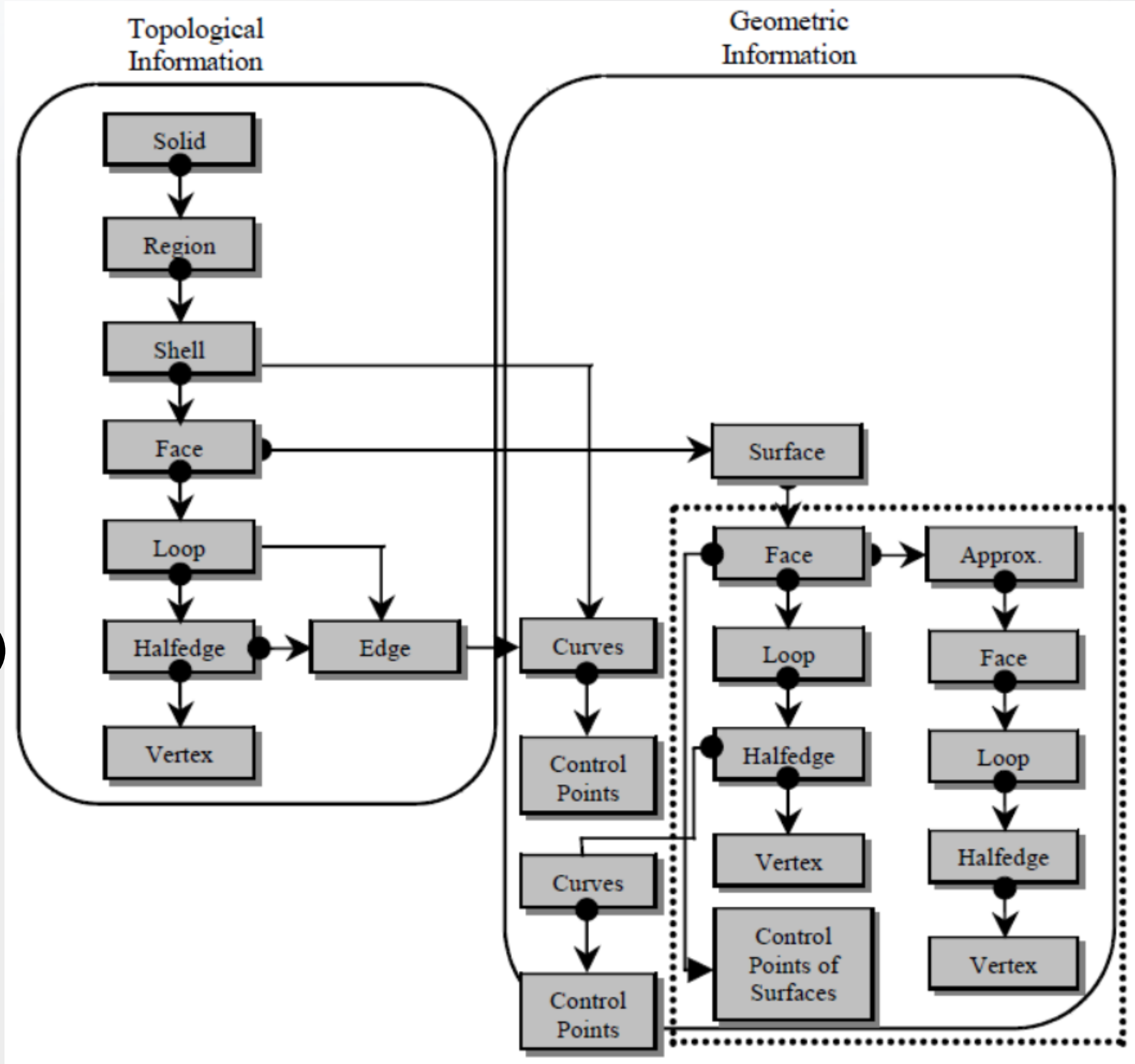




Boundary Representation
B-Rep Solid Modeling

Data storage structure tree

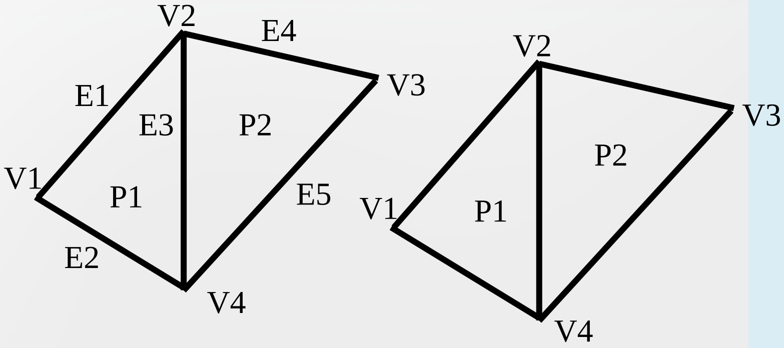




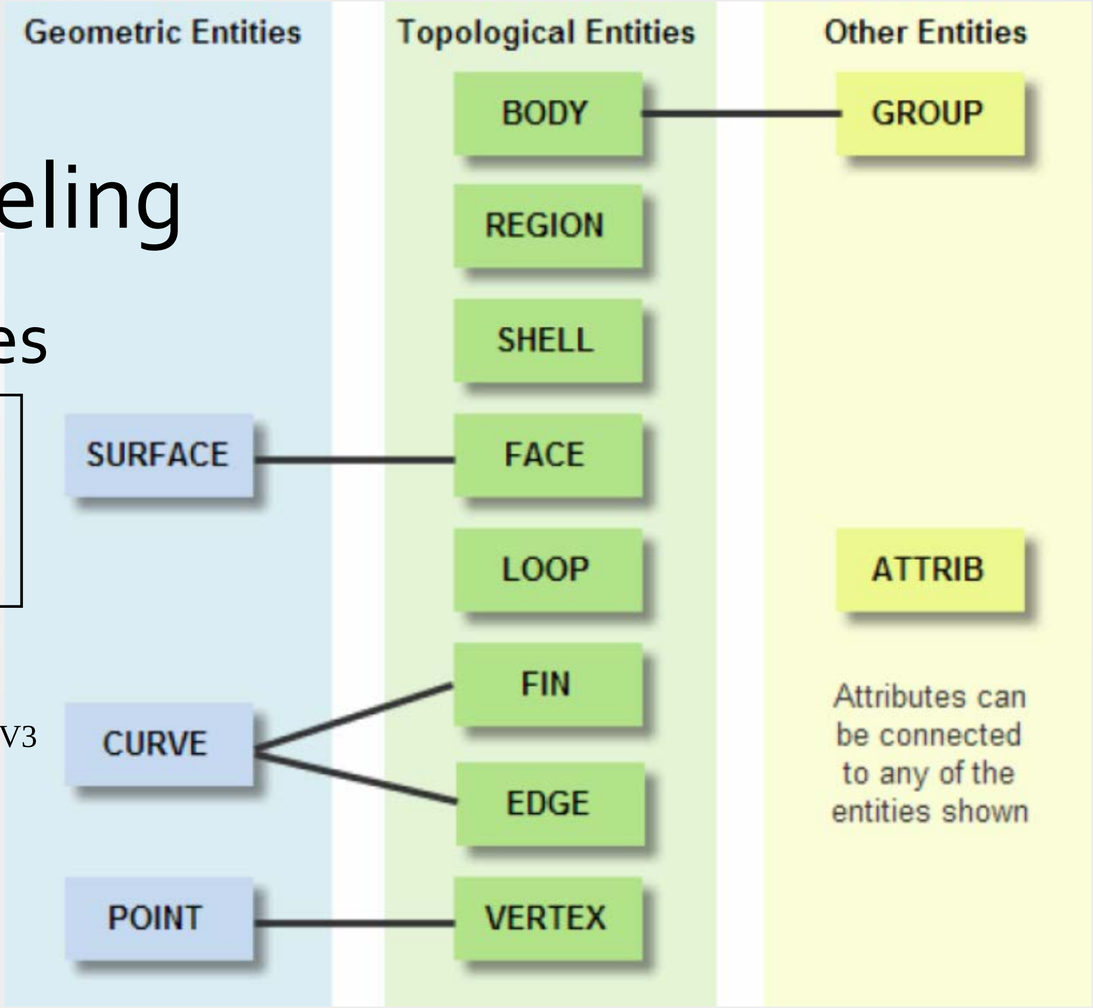
B-Rep Solid Modeling

Data storage structures

Vertex#	Location	Edge#	Vertices	Polygon#	Edges
V1	0,5,0	E1	V1,V2	P1	E1,E3,E2
V2	4,15,0	E2	V1,V4	P2	E3,E4,E5
V3	4,10,0	E3	V2,V4		
V4	8,0,0	E4	V2,V3		
		E5	V3,V4		

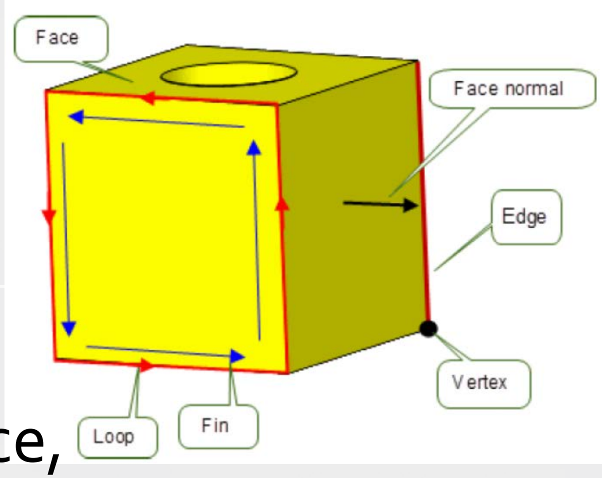


Vertex#	Location	Polygon#	Vertices
V1	0,5,0	P1	V1,V2,V4
V2	4,15,0	P2	V2,V3,V4
V3	4,10,0		
V4	8,0,0		



Relationships between
Parasolid topological entities

Parasolid topological entities in a body



Topology Description

Face A face is a bounded subset of a surface, whose boundary is a collection of zero or more loops. A face with zero loops forms a closed entity, such as a full spherical face.

Loop A loop is a connected component of a face boundary. A loop can have: an ordered ring of distinct fins, a set of vertices

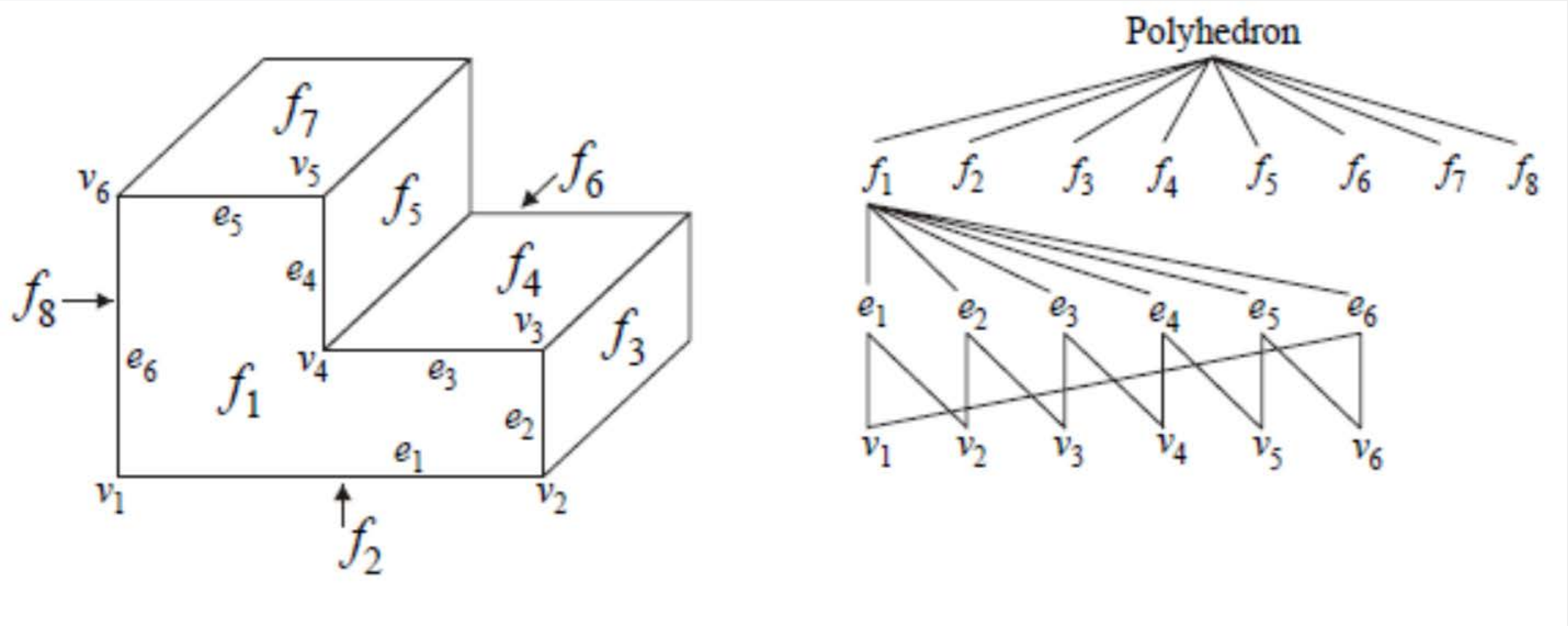
Fin A fin represents the oriented use of an edge by a loop.

Edge An edge is a bounded piece of a single curve. Its boundary is a collection of zero, one or two vertices.

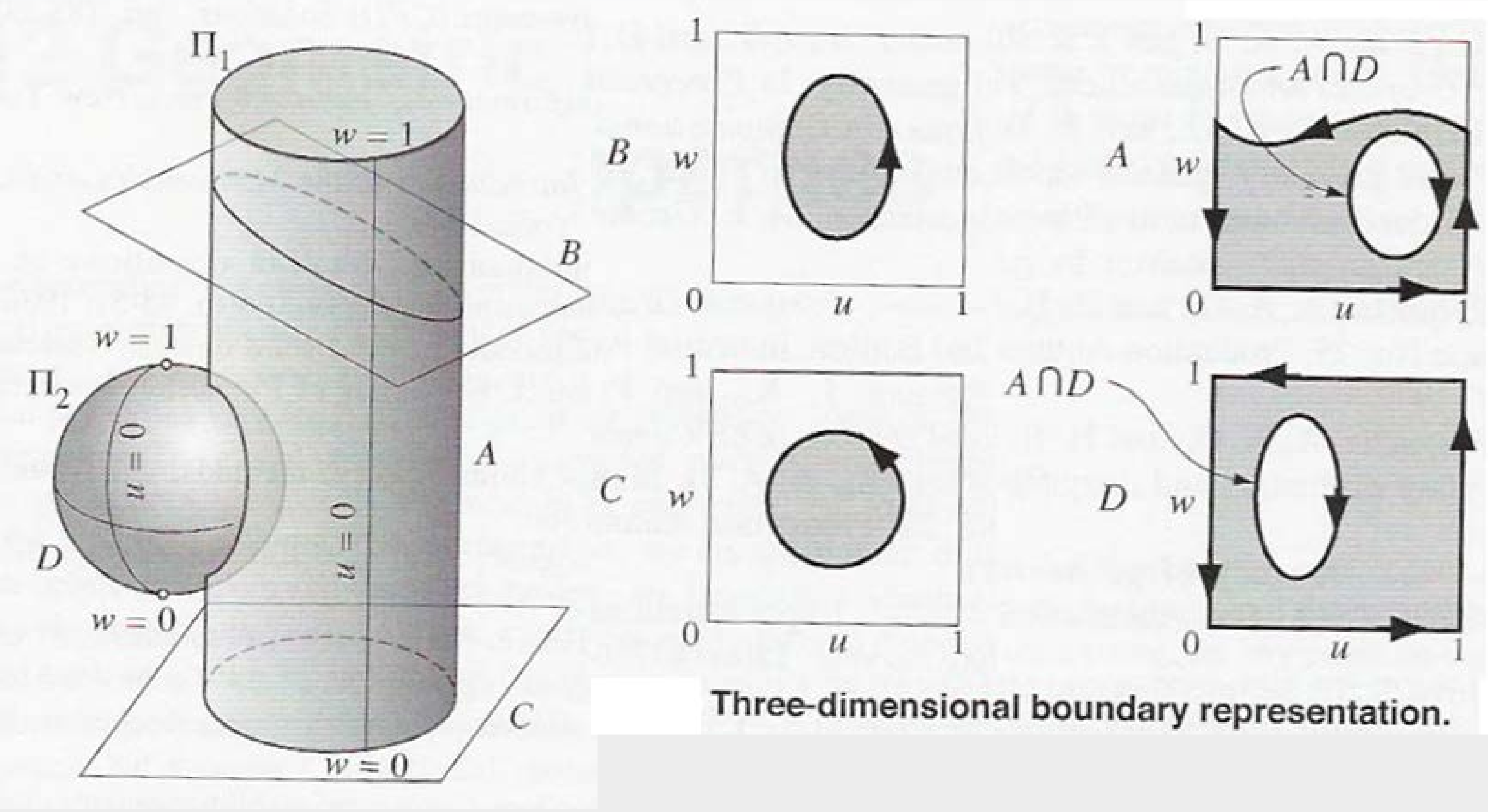
Vertex A vertex represents a point in space. A vertex has a single point, which may be null.

Boundary Representation

B-Rep models describe solids topologically, comprising faces, edges and vertices – surface oriented models:



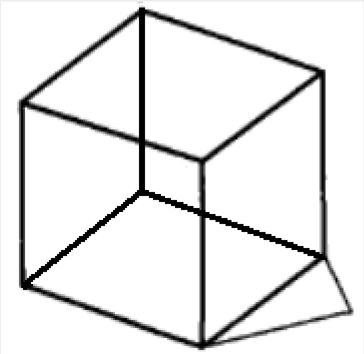
3D B-Rep Boundary Representation model



Three-dimensional boundary representation.

Boundary Representation

- The B-Rep method represents a solid as a collection of boundary surfaces. The **database records** both of the **surface geometry** and the **topological relations** among these surfaces.
- Boundary representation does not guarantee that a group of boundary surfaces (often polygons) form a **closed solid**.
- The data are also not in the ideal form for model calculations.
- This B-Rep representation is used mainly for **graphical displays**.



Boundary Representation (B-rep)

Object List -- giving object name, a list of all its boundary surfaces, and the relation to other objects of the model.

Surface List -- giving surface name, a list of all its component polygons, and the relation to other surfaces of the object.

Polygon List -- giving polygon name, a list of all boundary segments that form this polygon, and the relation to other polygons of the surface.

Boundary List -- giving boundary name, a list of all line segments that for this boundary, and the relation to other boundary lines of the polygon.

Line List -- giving line name, the name of its two end points, and the relation to other lines of the boundary line.

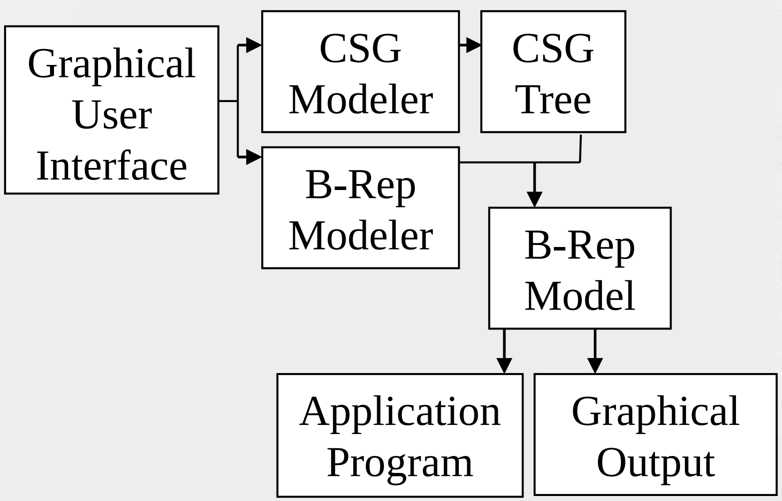
Point List -- giving point name, the X, Y and Z coordinates of the point and, and the relation to other end point of the line.

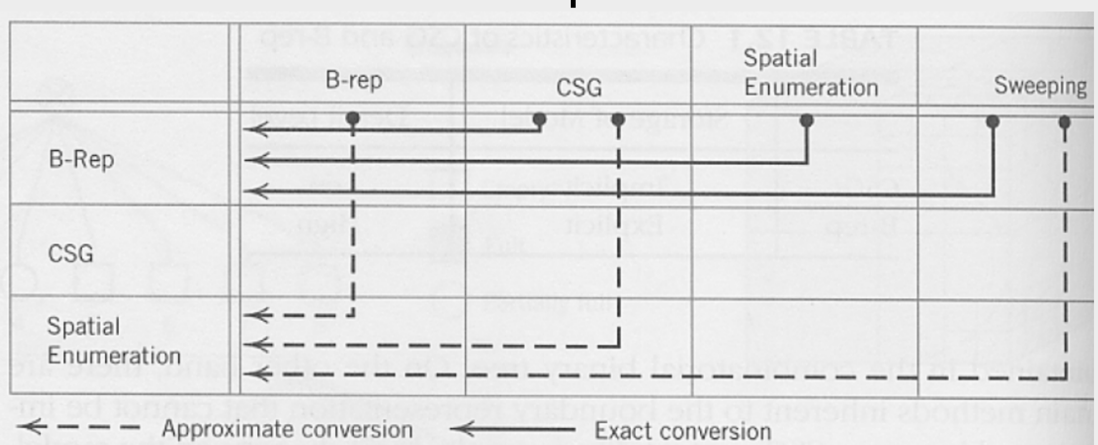
Model Conversions, hybrid solid modelers

CSG models are quite concise and can be converted into B-Rep models, which in turn are useful for graphical outputs.

Many CAD systems have a hybrid data structure, using both CSG and B-rep at the same time.

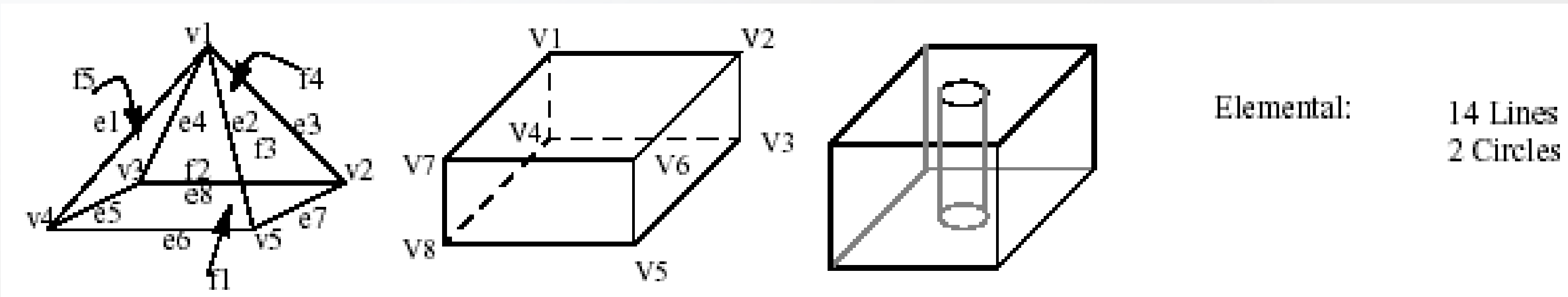
Catia, Solidworks, I-DEAS and Pro-Engineer CAD software packages are hybrid solid modelers that allow user input, and subsequent data storage, in both CSG and B-Rep structures.



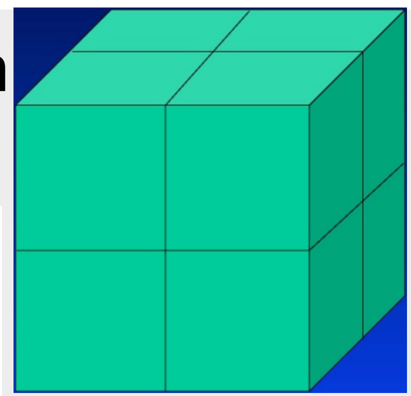


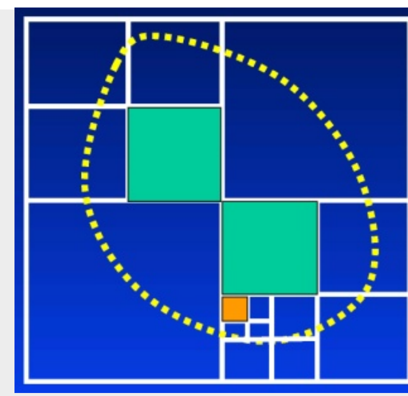
Solid Modeling

B-rep modeling data structure

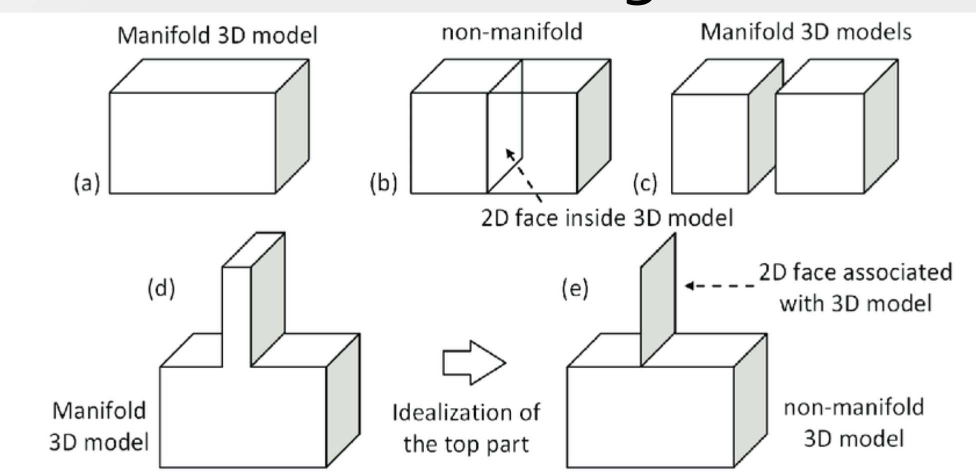


Octree solid representation





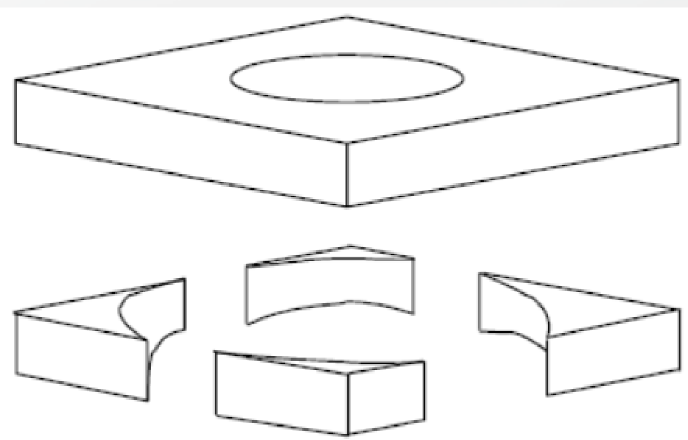
Manifold modeling



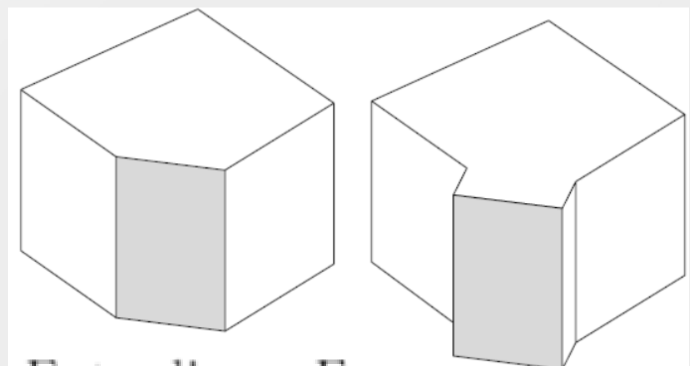
Solid Modeling

Shape Variation Due to Parameter Values

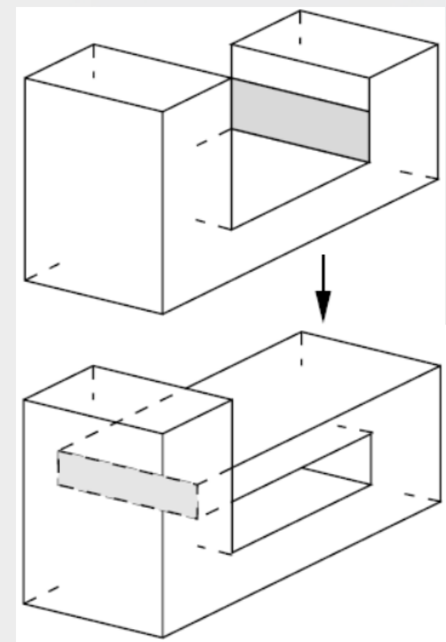
A **CSG** model design cannot be displayed or converted to **Brep** boundary representation, since different parameter assignments could lead to totally different shapes.



Shape Variation Due to Parameter Values

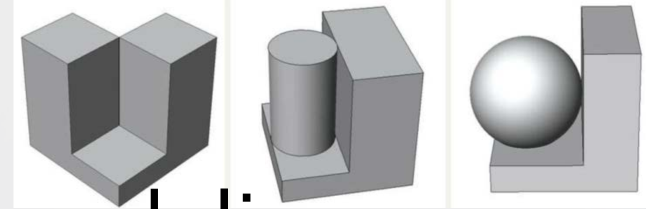


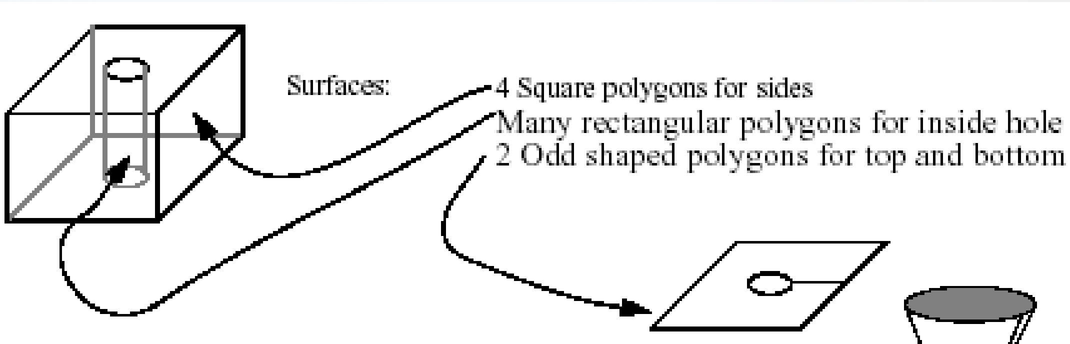
Extruding a Face

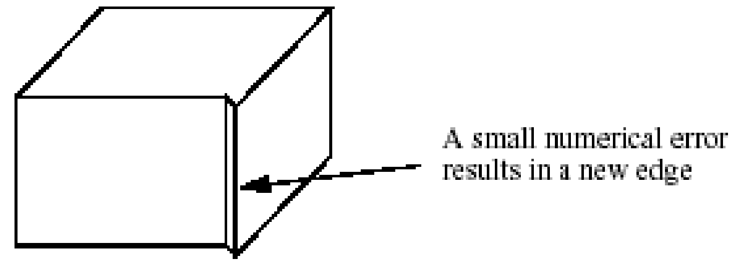


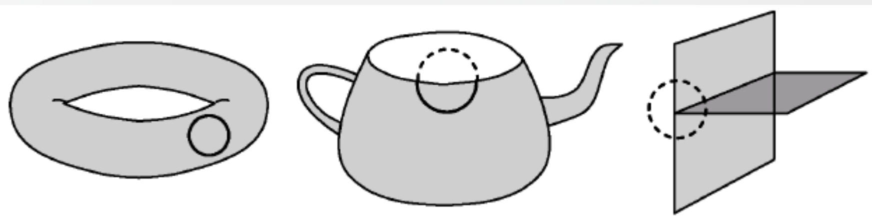
Error in Face Extrusion

Manifold and non-manifold modeling

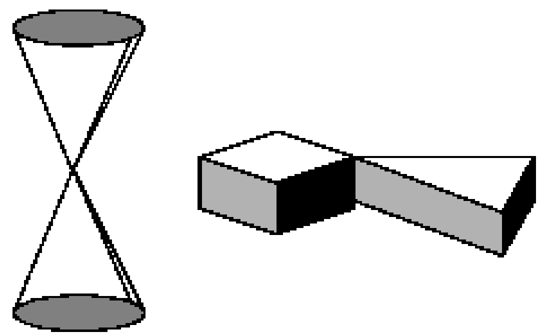




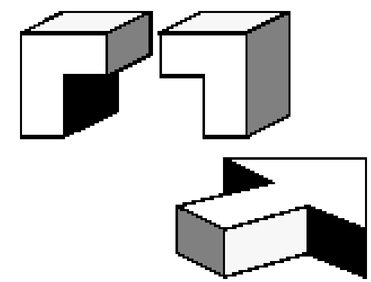




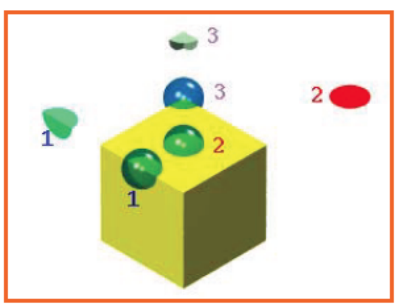
Manifold, manifold-with-boundary, and non-manifold



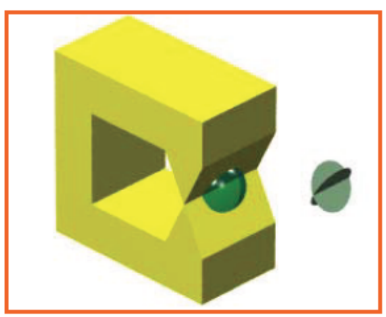
Non-Manifold / Open Parts



Manifold Parts

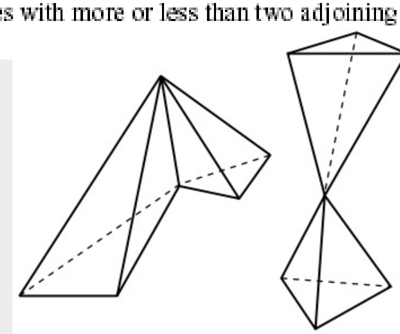


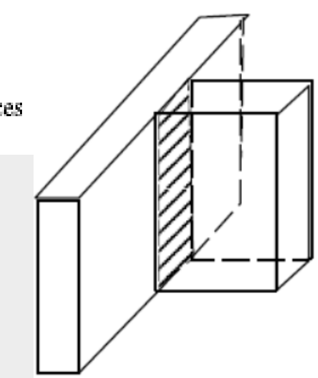
2-D manifold



2-D non-manifold

- Non-manifold parts have,
  - vertices with less than 3 adjoining faces
  - edges with more or less than two adjoining faces
  - etc.

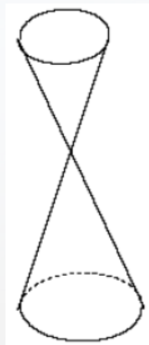


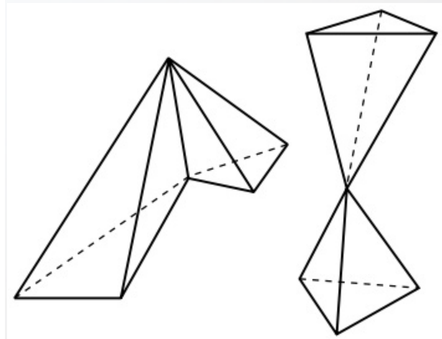


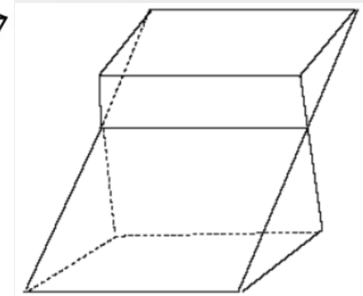
Non two-manifold surface

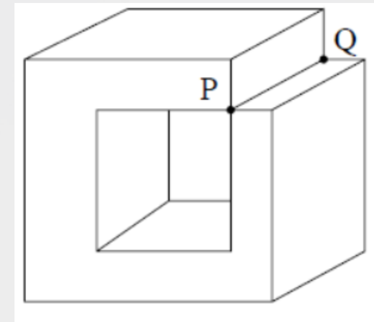
Manifold and non-manifold modeling

Non-manifold Surfaces



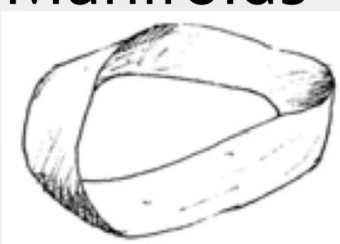




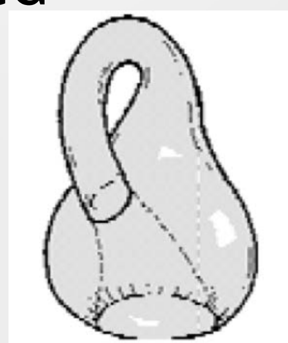


A Non-manifold Object

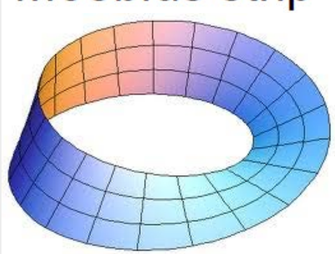
Non-oriented Manifolds

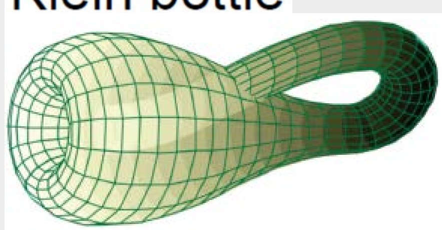


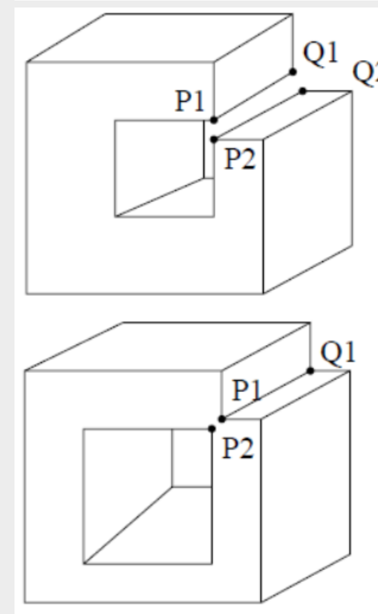
Moebius strip



Klein bottle



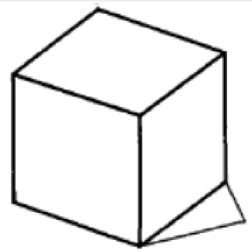


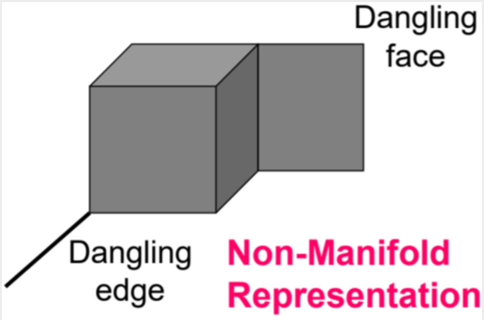


Two Possible Topologies

Manifold and non-manifold modeling

The **2-manifold** is a fundamental concept from **algebraic topology** and **differential topology**. It is a surface embedded in **R**^{3} such that the infinitesimal neighborhood around any point on the surface is topologically equivalent (`locally diffeomorphic') to a disk. Intuitively, the surface is `watertight' and **contains no holes** or **dangling edges**. Typically, **the manifold is bounded (or closed).**
For example, **a plane is a manifold but is unbounded** and thus not watertight in any physical sense.
A manifold-with-boundary is a surface locally approximated by either a disk or a half-disk.
All other surfaces are non-manifold.



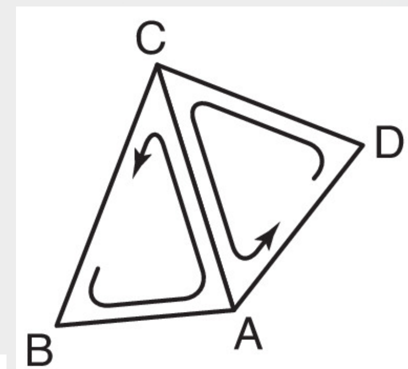


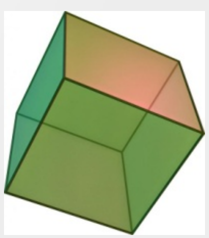
Boundary Representation (B-Rep)

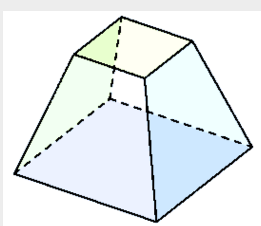
Solids represented by **faces**, **edges** and **vertices**

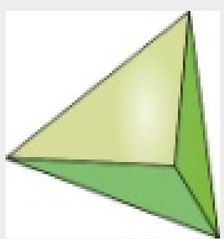
Topological rules must be satisfied to ensure valid objects

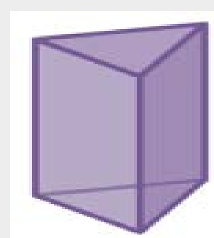
- faces bounded by loop of edges
- each edge shared by exactly two faces
- each edge has a vertex at each end
- at least 3 edges meet at each vertex

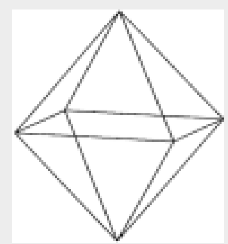


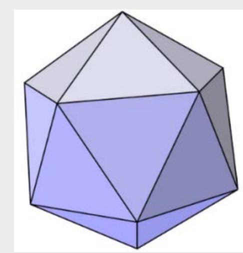


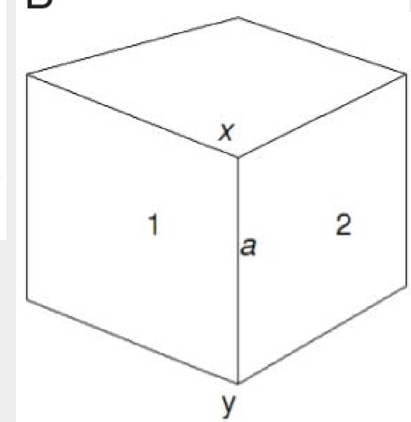




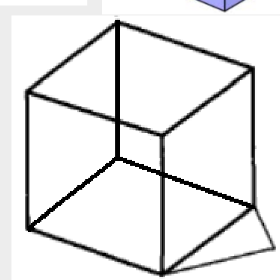






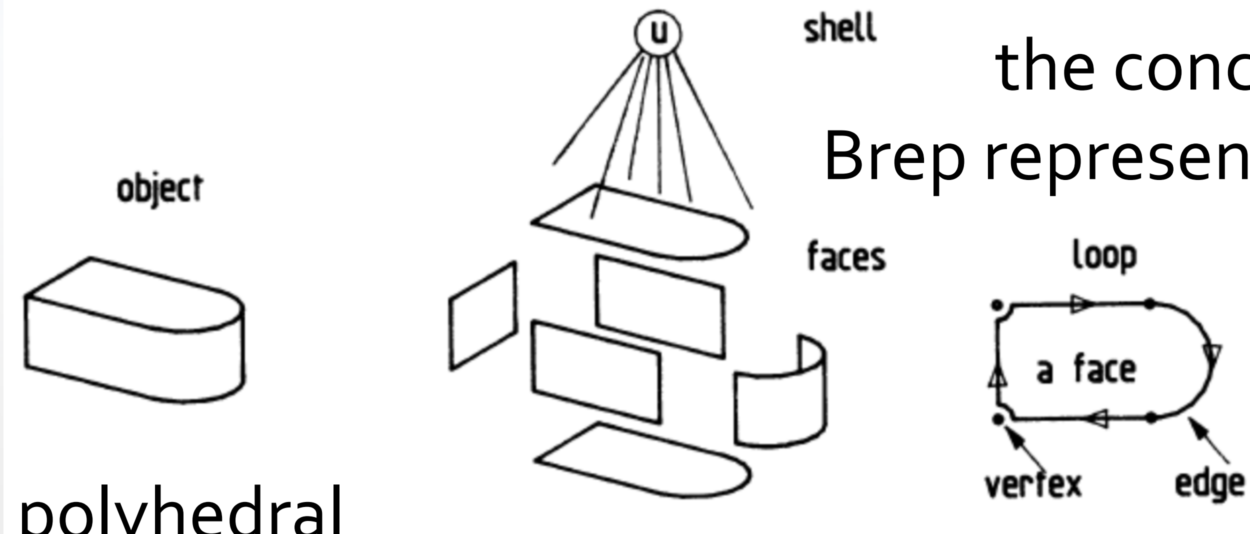


this is not valid solid object:

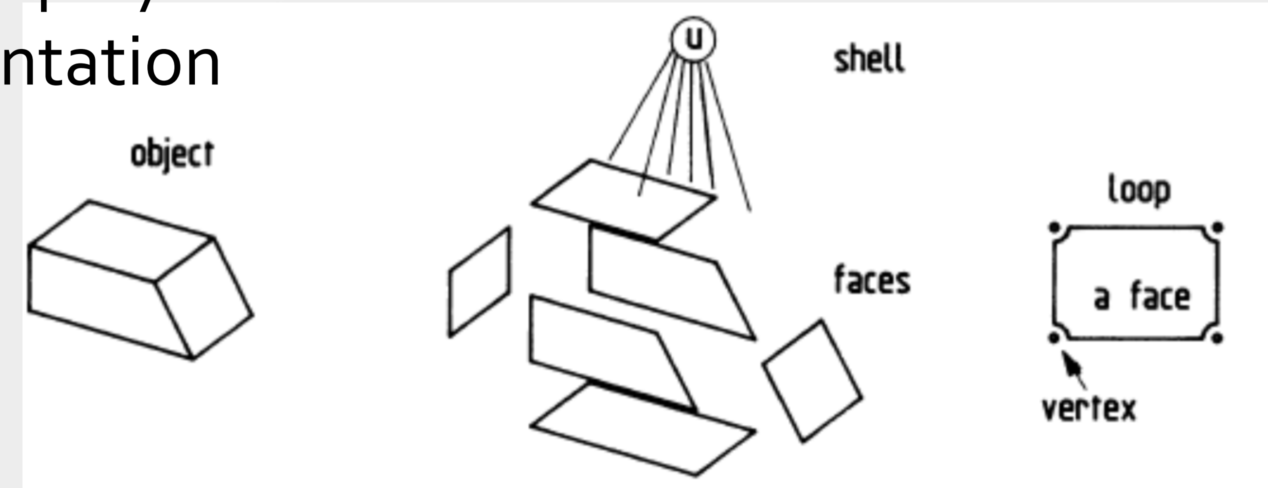


Brep and CSG polyhedral representations

the concept of
Brep representation

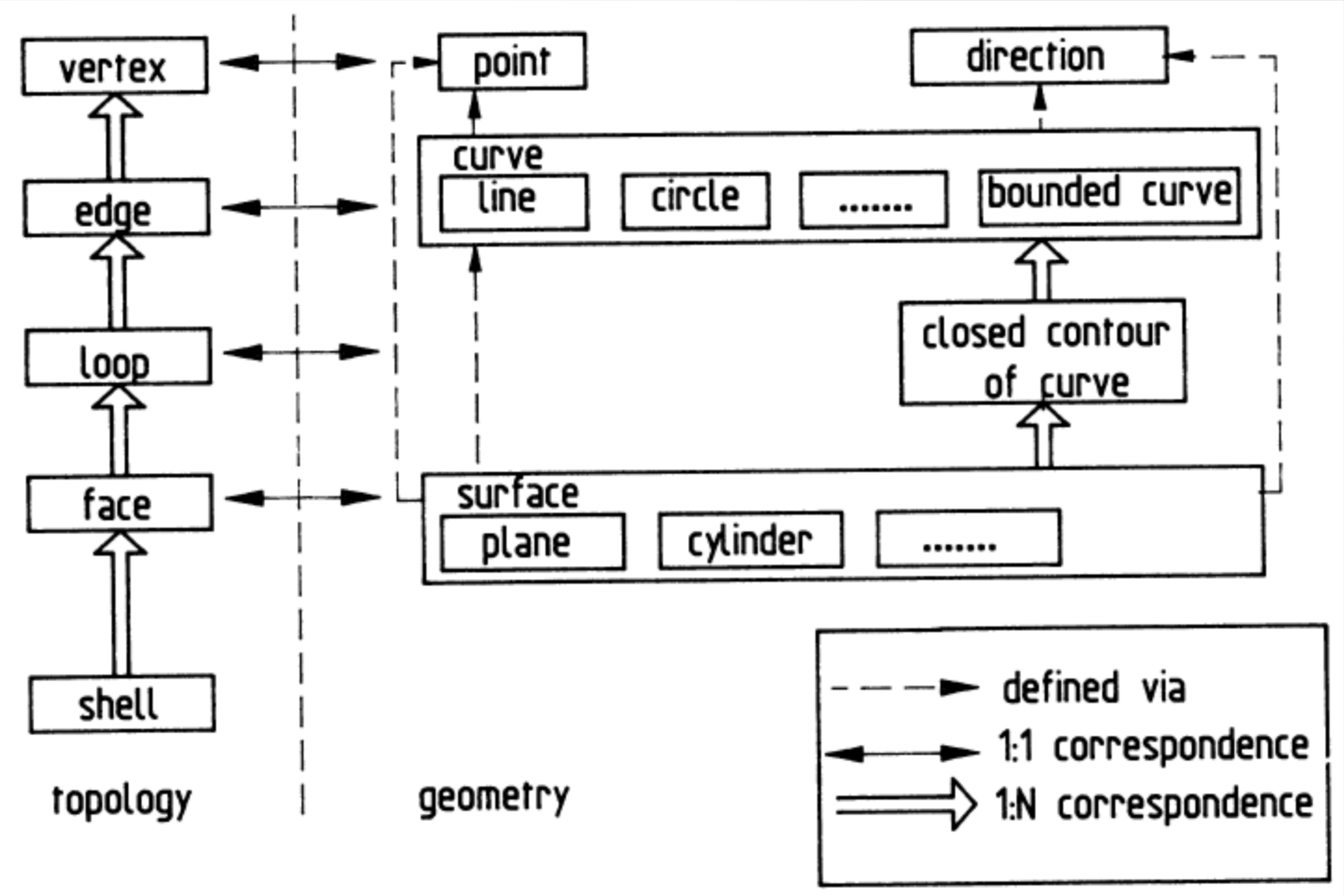


the CSG polyhedral
representation



The underlying structure to be recorded

Brep

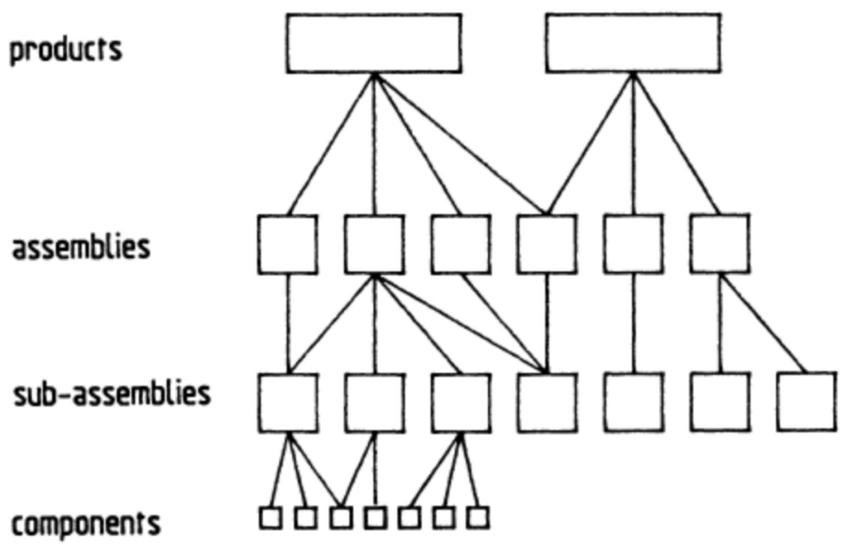


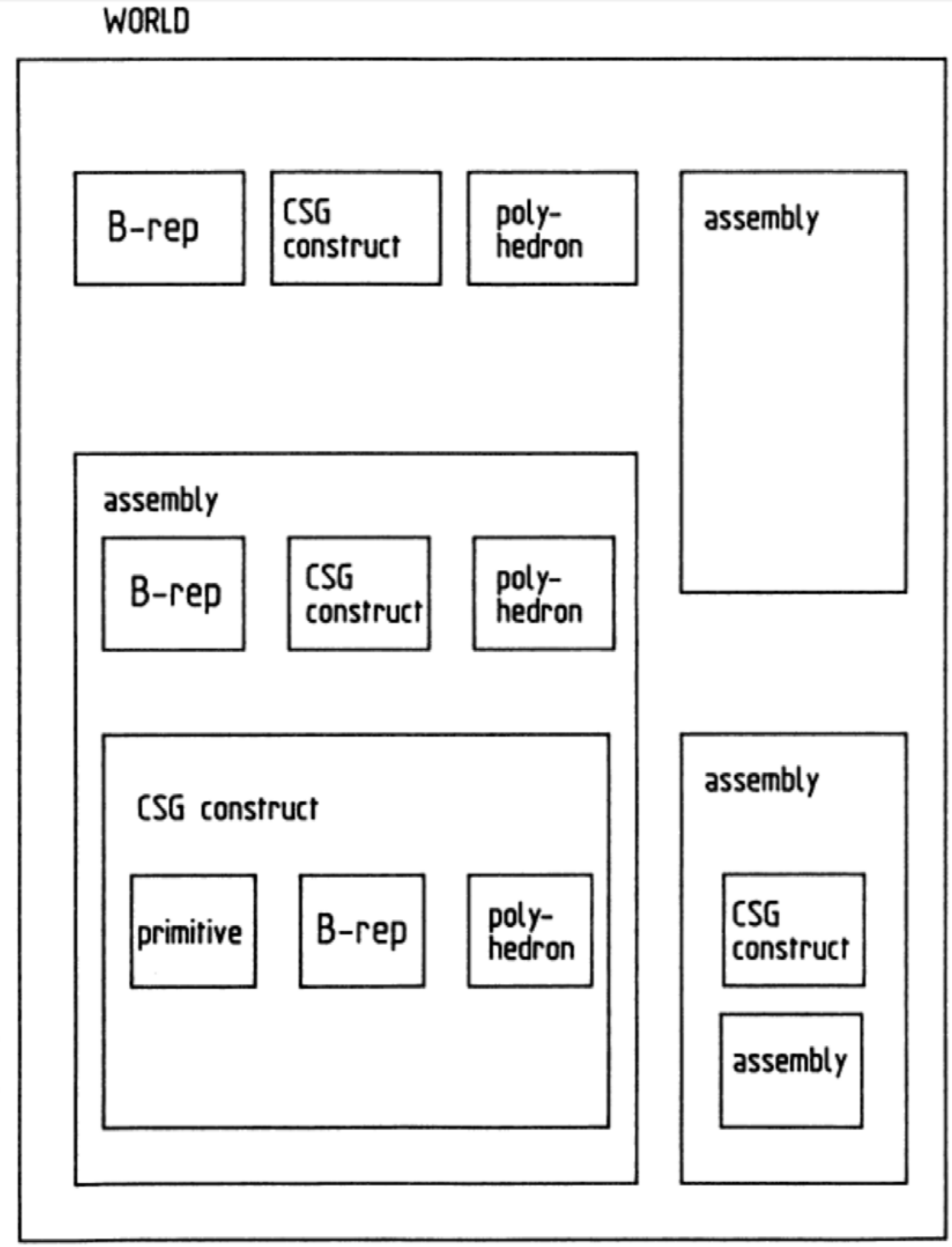
Principal exchange possibilities for solid models

	Receiving system type		
type of sending system	CSG	B-rep	polyhedron
CSG	exact	the CSG expression must be evaluated into B-rep	the CSG primitives must be approximated by polyhedra; the approximate model must then be evaluated to produce a polyhedron model
B-rep	not possible	exact	curves and surfaces in the model must be approximated by straight lines and planes
polyhedron	not possible	exact	exact

Assembly World

An example illustrating the scope aspect of the reference schema





Boundary Representation (B-Rep)

Closed Surface : One that is continuous without breaks.

Orientable Surface : One in which it is possible to distinguish two sides by using surface normals to point to the inside or outside of the solid under consideration.

Boundary Model : Boundary model of an object is comprised of closed and orientable faces, edges and vertices. A database of a boundary model contains both its topology and geometry.

Topology : Created by Euler operations

Geometry : Includes coordinates of vertices, rigid motions and transformations

Boundary Representation (B-Rep)

Involves surfaces that are
– *closed, oriented manifolds embedded in 3-space*

A **manifold surface:**
– each point is *homeomorphic to a disc*
A manifold surface is **oriented if:**
– any path on the manifold maintains the orientation of the normal
An oriented manifold surface is **closed if:**
– it partitions 3-space into points inside, on, and outside the surface
A closed, oriented manifold is **embedded in 3-space if:**
– Geometric (and not just topological) information is known

Object Modeling with B-rep

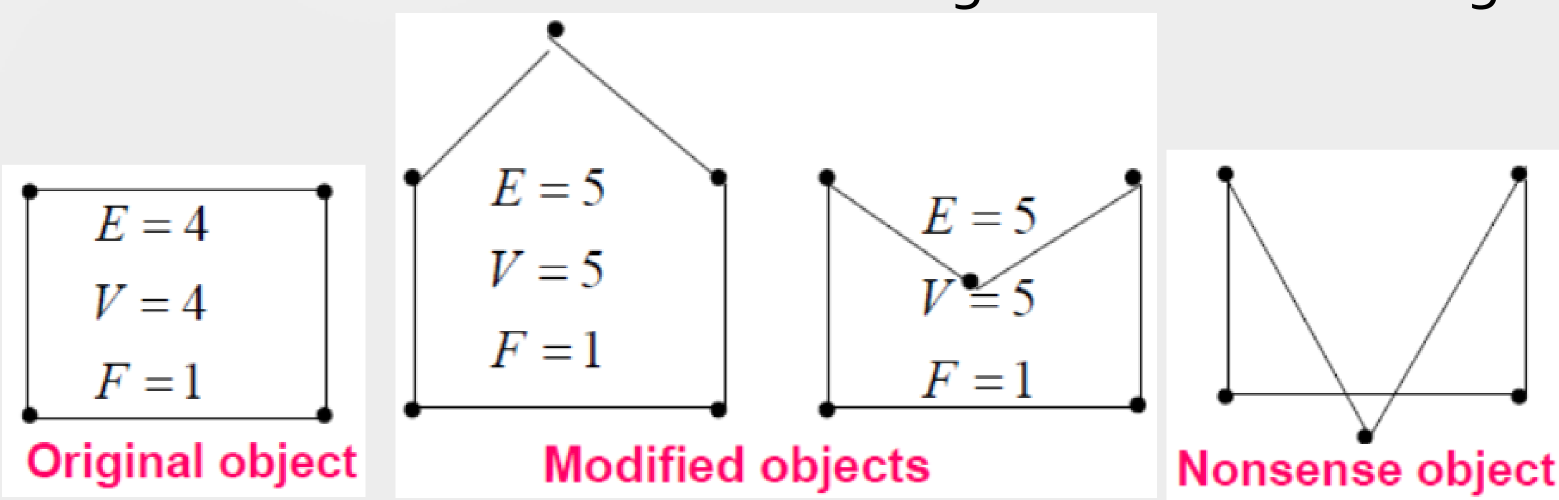
Both polyhedra and curved objects can be modeled using the following primitives

- **Vertex :** A unique point (ordered triplet) in space.
- **Edge :** A finite, non-self intersecting directed space curve bounded by two vertices that are not necessarily distinct.
- **Face :** Finite, connected, non-self intersecting region of a closed, orientable surface bounded by one or more loops.
- **Loop :** An ordered alternating sequence of vertices and edges. A loop defines non-self intersecting piecewise closed space curve which may be a boundary of a face.
- **Body :** An independent solid. Sometimes called a shell has a set of faces that bound single connected closed volume. A minimum body is a point (vortex) which topologically has one face one vortex and no edges. A point is therefore called a seminal or singular body.
- **Genus :** Hole or handle.

Boundary Representation

Euler Operations (Euler –Poincare�much Law): The validity of resulting solids is ensured via Euler operations which can be built into CAD/CAM systems.

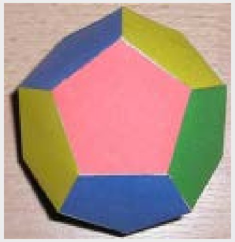
Volumetric Property calculation in B-rep: It is possible to compute volumetric properties such as mass properties (assuming uniform density) by virtue of **Gauss divergence** theorem which converts volume integrals to surface integrals.

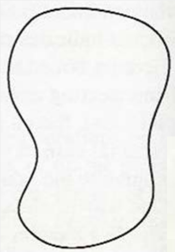


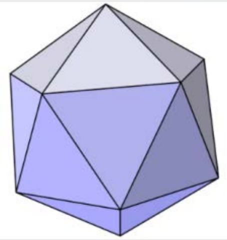
Euler-Poincare Law

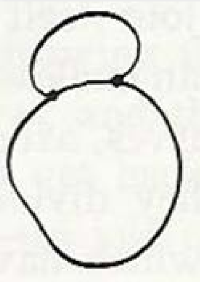
Leonhard Euler (1707-1783),Henri Poincaré (1854-1912)

Euler (1752) proved that polyhedra that are homeomorphic to a sphere are topologically valid if they satisfy the equation:



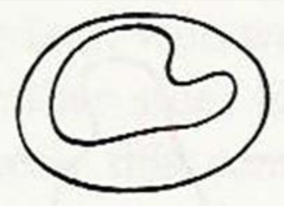


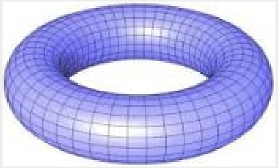




Equation	
$F - E + V - L = 2(B - G)$	General
$F - E + V = 2$	Simple Solids
$F - E + V - L = B - G$	Open Objects

F=Face	E=Edge	V=Vertices
B=Bodies	L=Faces' inner Loop	G=Genus



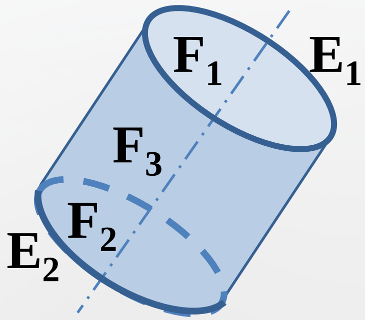


Polygonal Loops satisfy $(L)(V)-(L)(E_l)=0$

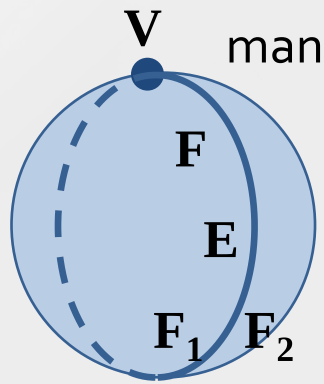
B-Rep of cylinder and circle

The extended Euler-Poincarré formula allow test the topology for polyhedral solids :

$$F - E + V - L = 2(B - G)$$



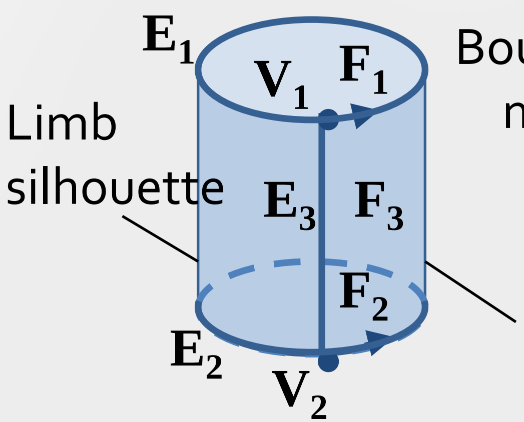
Faces = F= 3
Vertices = V = 0
Edges = E = 2
3+0-2-0 ≠ 2(1-0)



Boundary Model of Sphere, manifold topology test:

F=1 V=1 E=1
1+1-1-0 ≠ 2(1-0)

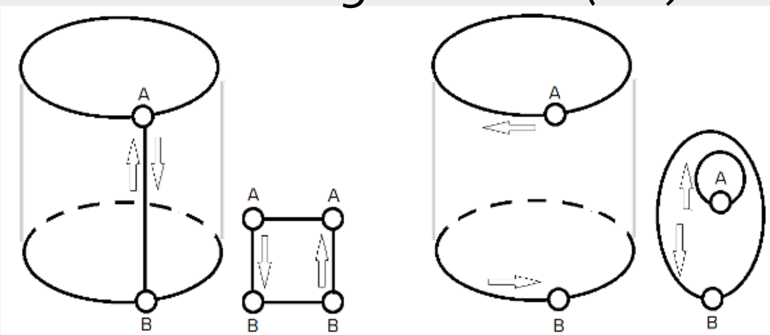
F=2 V=1 E=1
2+1-1-0 = 2(1-0)



Boundary Model of Cylinder, manifold topology test:

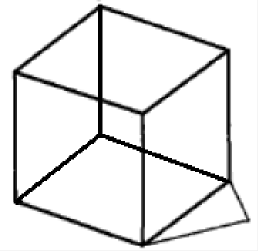
F=3 V=2 E=3
3+2-3-0 = 2(1-0)

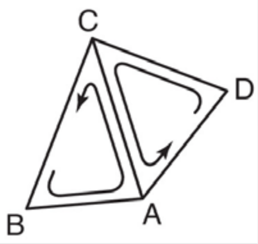
Cylinder with upper and lower cap: *F=3 V=2 E=2*
3+2-2-1 = 2(1-0)



Euler Operations

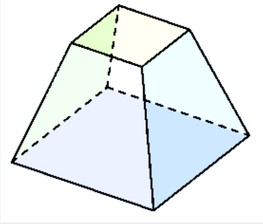
A connected structure of vertices, edges and faces that always satisfies Euler's formula is known as **Euler object.** The process that adds and deletes these boundary components is called an **Euler operation**.

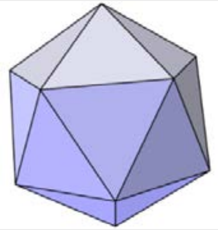


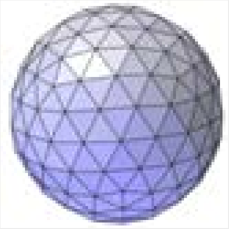


Applicability of Euler formula to solid objects:

- At least three edges must meet at each vertex.
- Each edge must share two and only two faces
- All faces must be simply connected (homeomorphic to disk) with no holes and bounded by single ring of edges.
- The solid must be simply connected with no through holes

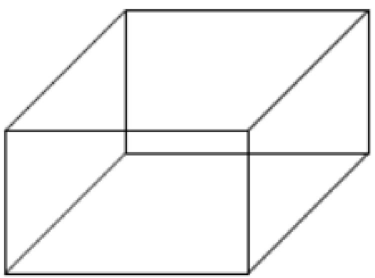






Validity Checking for Simple Solids

$F - E + V = 2$ Simple Solids

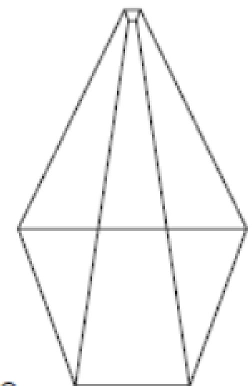


$E = 12$

$V = 8$

$F = 6$

$6 - 12 + 8 = 2$

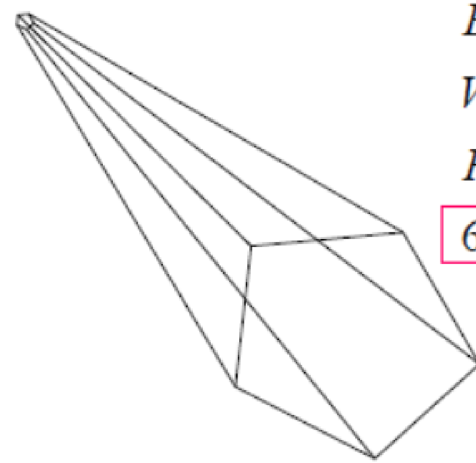


$E = 8$

$V = 5$

$F = 5$

$5 - 8 + 5 = 2$

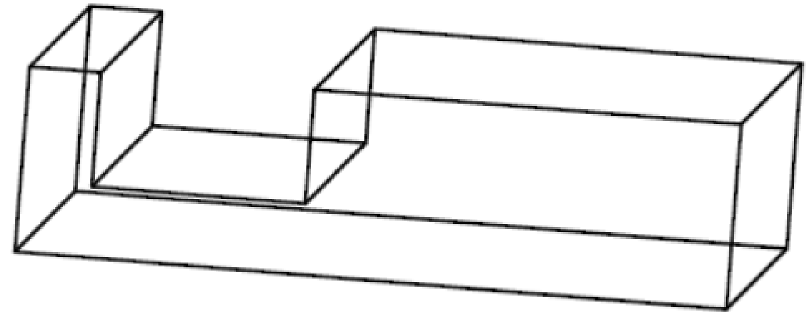


$E = 10$

$V = 6$

$F = 6$

$6 - 10 + 6 = 2$



$E = 24$

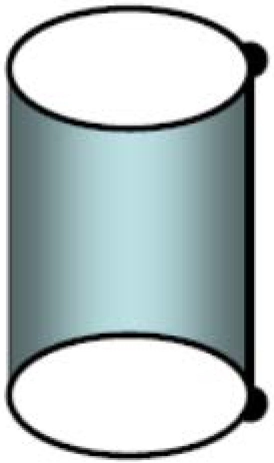
$V = 16$

$F = 10$

$10 - 24 + 16 = 2$

Validity Checking for Simple Solids

$F - E + V = 2$ Simple Solids



$E = 3$

$V = 2$

$F = 3$

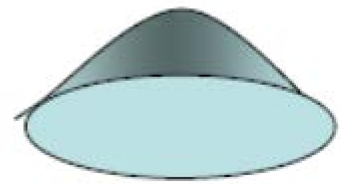
$3 - 3 + 2 = 2$

$E = 2$

$V = 2$

$F = 2$

$2 - 2 + 2 = 2$



$E = 2$

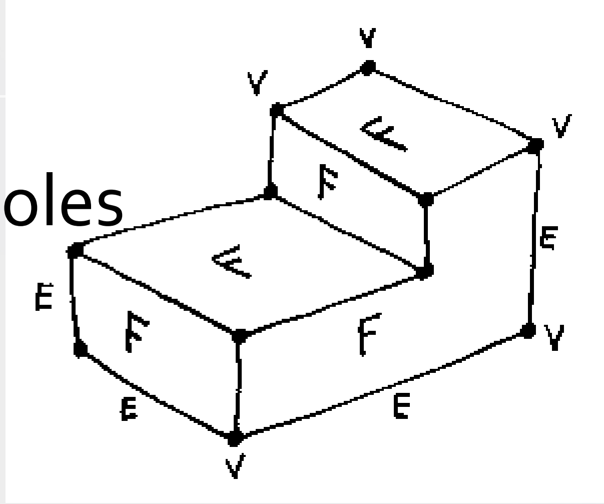
$V = 2$

$F = 2$

$2 - 2 + 2 = 2$

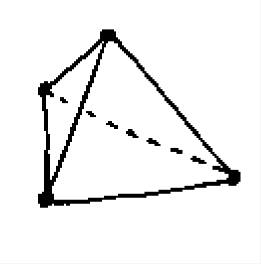
B-rep Models

Suppose a solid with flat faces and no holes has *F* faces, *E* edges, and *V* vertices.



A tetrahedron is the simplest:

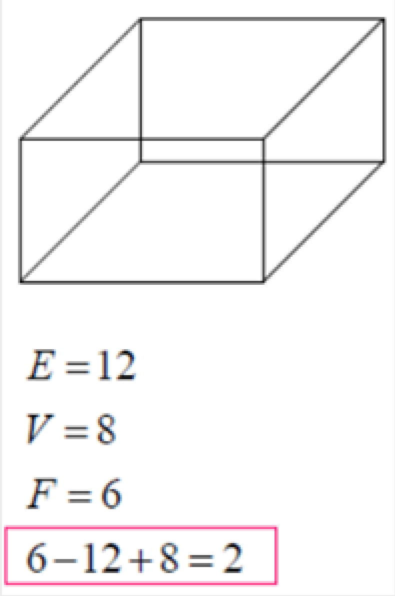
$F = 4$, $E = 6$, $V = 4$



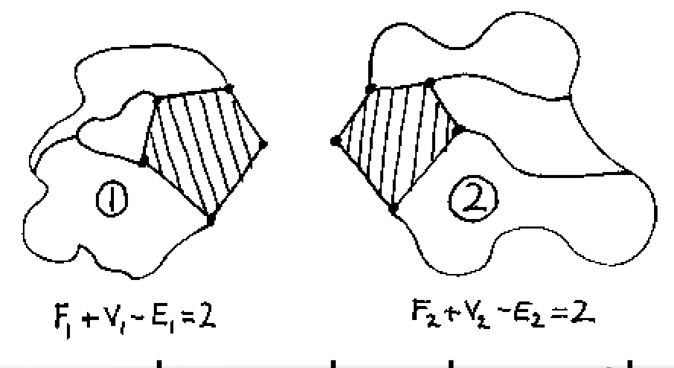
In this case $F + V - E = 2$.

This is also true for a cuboid (try it).

Is it true in general?



B-rep Models



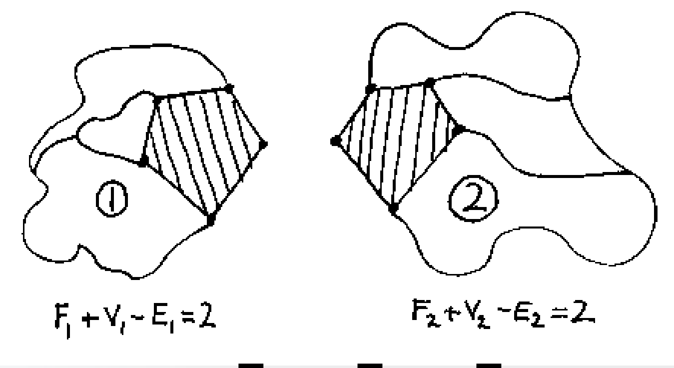
Suppose we have two solids, 1 and 2, and we know that the formula is true for each of them because we've counted. Suppose also that the solids each have a face which is the mirror image of the corresponding face on the other (the shaded pentagons). These **faces** don't have to be pentagons; say in general that they each have ***n*** **edges**.

What happens if we **glue** the solids together at the shaded faces to make a more complicated object, called 3?

The two faces disappear, so we know that: $F3 = F1 + F2 - 2$

Two sets of *n* vertices become one : $V3 = V1 + V2 - n$

Two sets of *n* edges become one : $E3 = E1 + E2 - n$



B-rep Models

The two faces disappear : $F_3 = F_1 + F_2 - 2$

Two sets of *n* vertices become one : $V_3 = V_1 + V_2 - n$

Two sets of *n* edges become one : $E_3 = E_1 + E_2 - n$

So $F_3 + V_3 - E_3 = F_1 + F_2 - 2 + V_1 + V_2 - n - (E_1 + E_2 - n)$

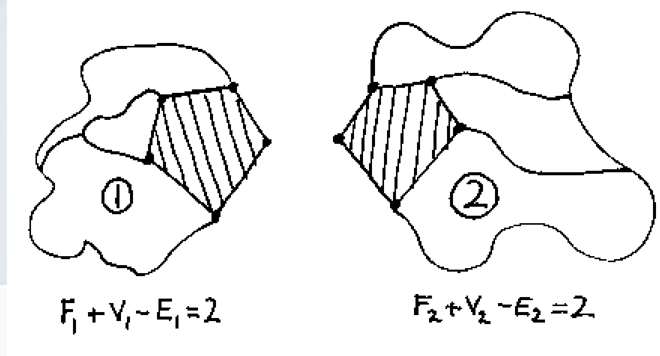
we can rearrange:

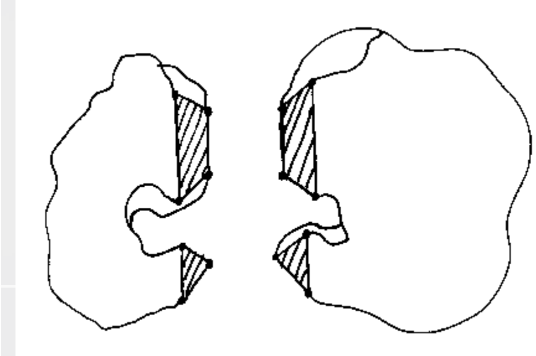
$$F_3 + V_3 - E_3 = (F_1 + V_1 - E_1) + (F_2 + V_2 - E_2) - n + n - 2$$

But we know that the first two parts in brackets both equal 2.
The *n* terms cancel, leaving us with: $F_3 + V_3 - E_3 = 2$

So the formula $F + V - E = 2$ works for all solids without holes, because we can start with simple solids (like the tetrahedron).

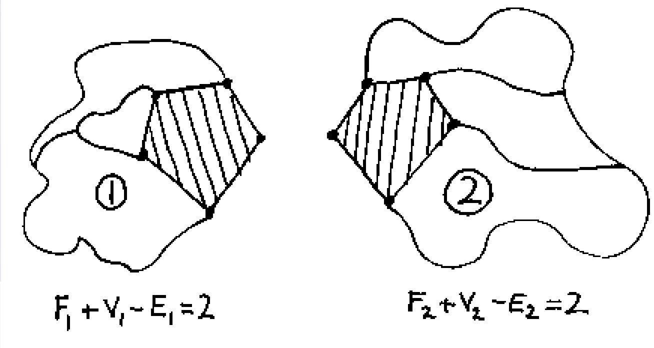
B-rep Models

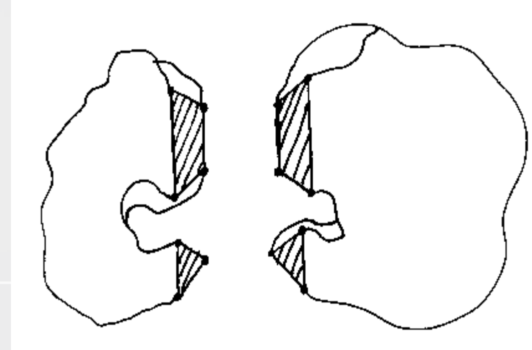




So the formula $F + V - E = 2$ works for all solids without holes, because we can start with simple solids (like the tetrahedron) for which we know the formula is true, and build complicated solids by gluing faces together. $F_3+V_3-E_3=F_1+F_2-2+V_1+V_2-n-(E_1+E_2-n)$
This is known as the *Euler-Poincaré* formula, after its discoverers.
What about solids with holes? Most real engineering components have holes, so we have to be able to deal with them. Think about **gluing together two objects** such that they will make an object **with a hole**:
The argument in the proof above about edges and vertices stays the same, but now $F_3 = F_1 + F_2 - 2\,(1 + H)$
where there are *H* holes. This gives us: $F + V - E = 2 - 2H$

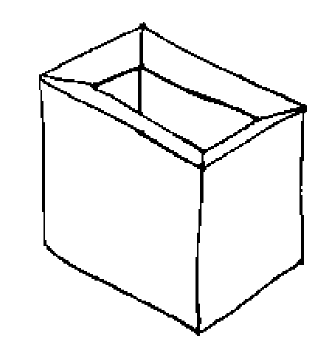
B-rep Models





So the formula *F* + *V* - *E* = 2 works for all solids without holes,

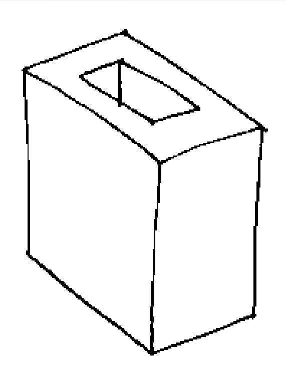
the formula where there are *H* holes *F* + *V* - *E* = 2 (1 + H)

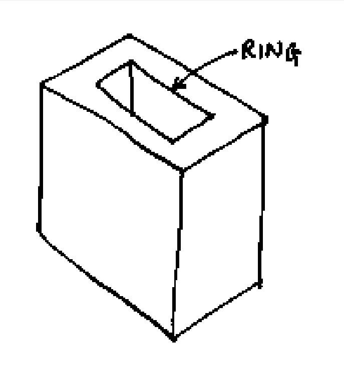


Check for this object: *F* = 16 , *E* = 32 , *V* = 16 , *H* = 1

So it works for that.

What about this one?



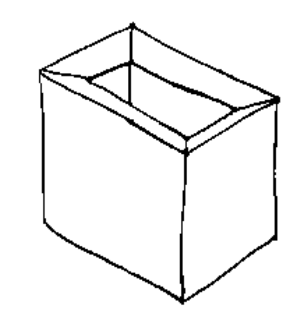


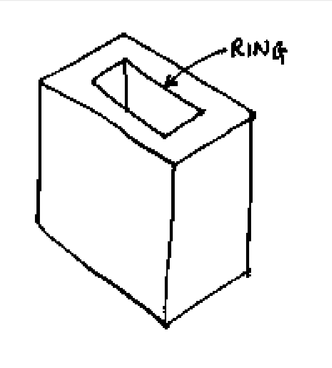
F = 10 , *E* = 24 , *V* = 16 , *H* = 1

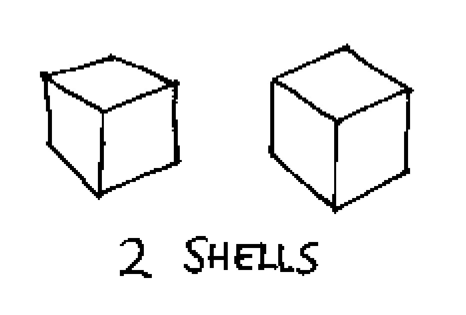
WRONG!

The problem is caused by the flat faces with *rings* of edges and vertices `floating' in them unconnected by edges to the other vertices.

B-rep Models







$F = 10$, $E = 24$, $V = 16$, $H = 1$

WRONG! $F + V - E = 2\,(1 + \mathrm{H})$

The problem is caused by the flat faces with ***rings*** of edges and vertices `floating' in them **unconnected** by edges to the other vertices.

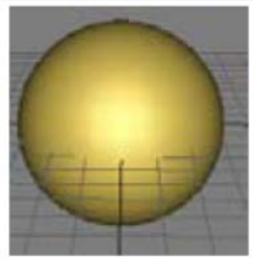
If we fix that up (say there are ***R*** **rings**), and also allow for the fact that we may want to describe two or more completely separate objects (called **shells**; suppose there are ***S*** of them), we come to the final version of the **Euler-Poincaré formula**:

$F + V - E - \mathrm{R} = 2\,(\mathrm{S} - \mathrm{H})$

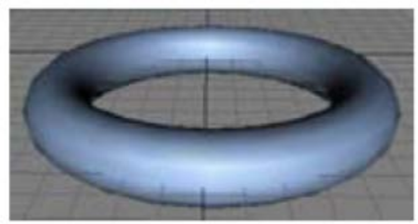
The number of **holes** through an object, ***H***, is called the ***genus*** of the object.

Loops (rings),Genus & Bodies

Genus zero

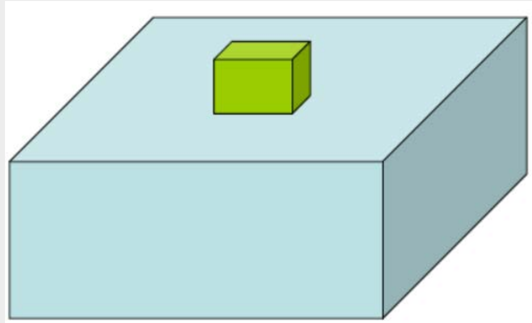


Genus one



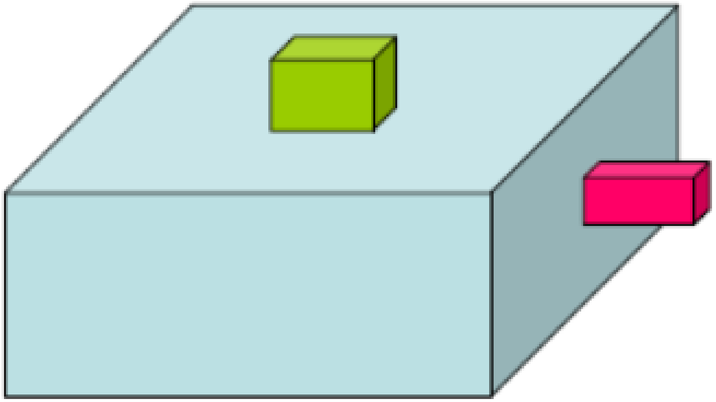
Genus two

One inner loop



Validity Checking for Polyhedra with inner loops

$$F - E + V - L = 2(B - G) \quad \text{General}$$



$E = 36$

$F = 16$

$V = 24$

$L = 2$

$B = 1$

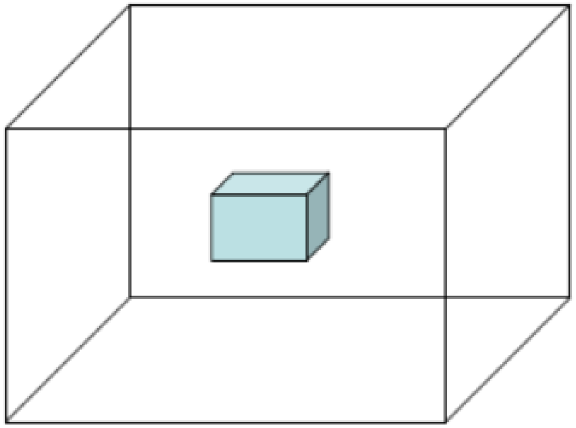
$G = 0$

$$16 - 36 + 24 - 2 = 2(1 - 0) = 2$$

Validity Checking for Polyhedra with holes

$$F - E + V - L = 2(B - G)$$ General

Interior hole (void)



$E = 24$

$F = 12$

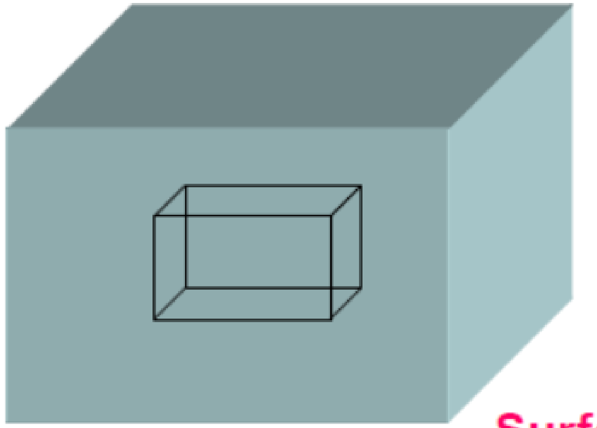
$V = 16$

$L = 0$

$B = 2$

$G = 0$

$$12 - 24 + 16 - 0 = 2(2 - 0) = 4$$



Surface hole

$E = 24$ $F = 11$

$V = 16$ $L = 1$

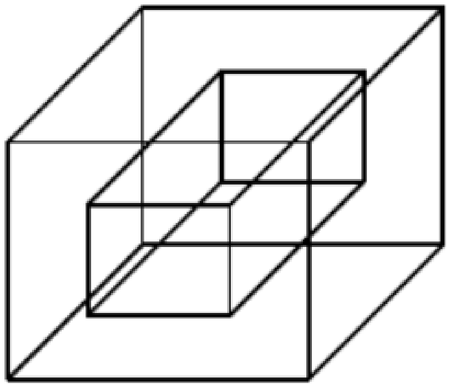
$B = 1$ $G = 0$

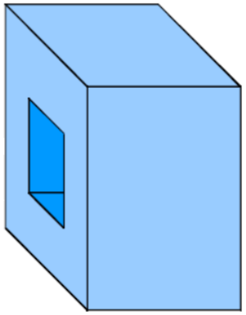
$$11 - 24 + 16 - 1 = 2(1 - 0) = 2$$

Validity Checking for Polyhedra with through holes (handles)

$$F - E + V - L = 2(B - G) \qquad \text{General}$$

Through hole





$E = 24$

$F = 10$

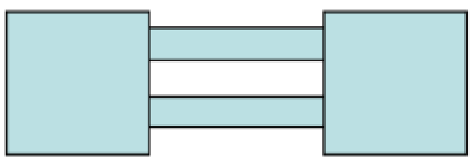
$V = 16$

$L = 2$

$B = 1$

$G = 1$

$$10 - 24 + 16 - 2 = 2(1 - 1) = 0$$



Handles/through hole

$E = 48 \quad F = 20$

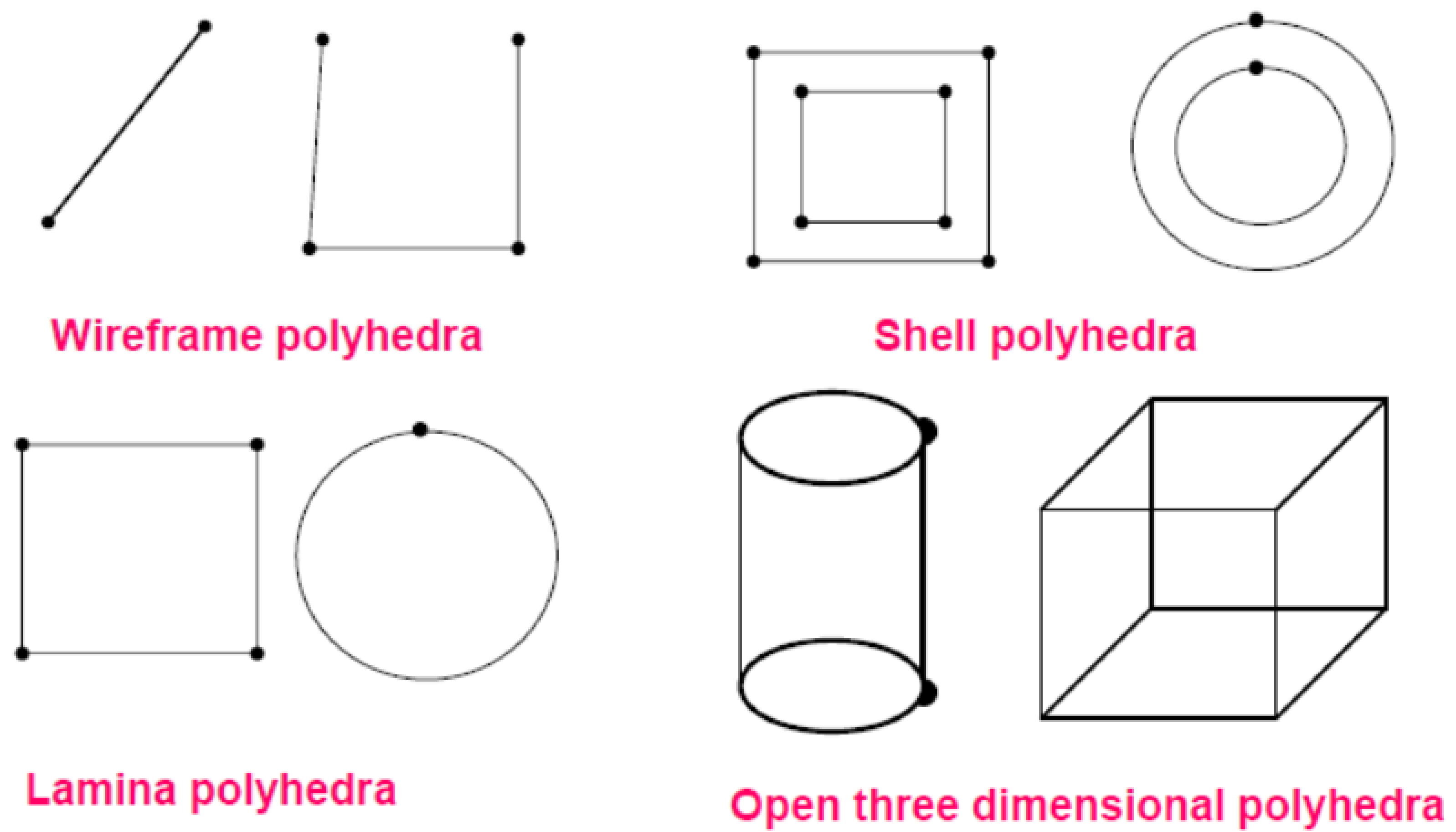
$V = 32 \quad L = 4$

$B = 1 \quad G = 1$

$$20 - 48 + 32 - 4 = 2(1 - 1) = 0$$

Validity Checking for Open Objects

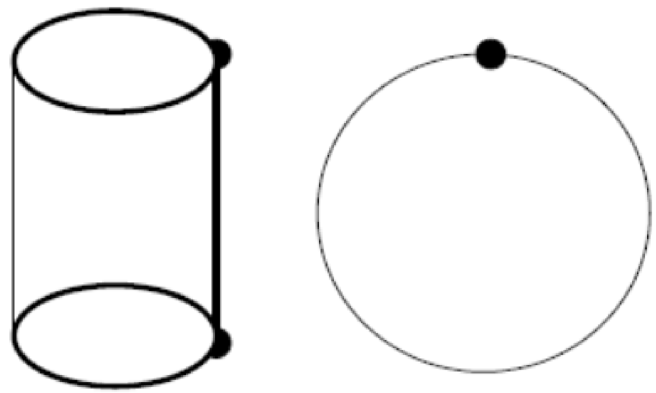
$$F - E + V - L = B - G$$



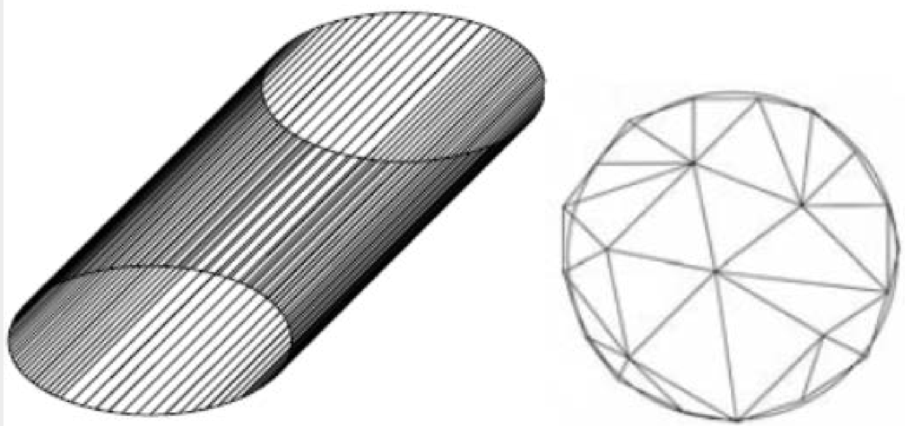
Exact vs. Faceted B-rep Schemes

Exact B-rep : If the curved objects are represented by way of equations of the underlying curves and surfaces, then the scheme is Exact B-rep.

Approximate or faceted B-rep : In this scheme of boundary representation any curved face divided into planar faces. It is also know as tessellation representation.



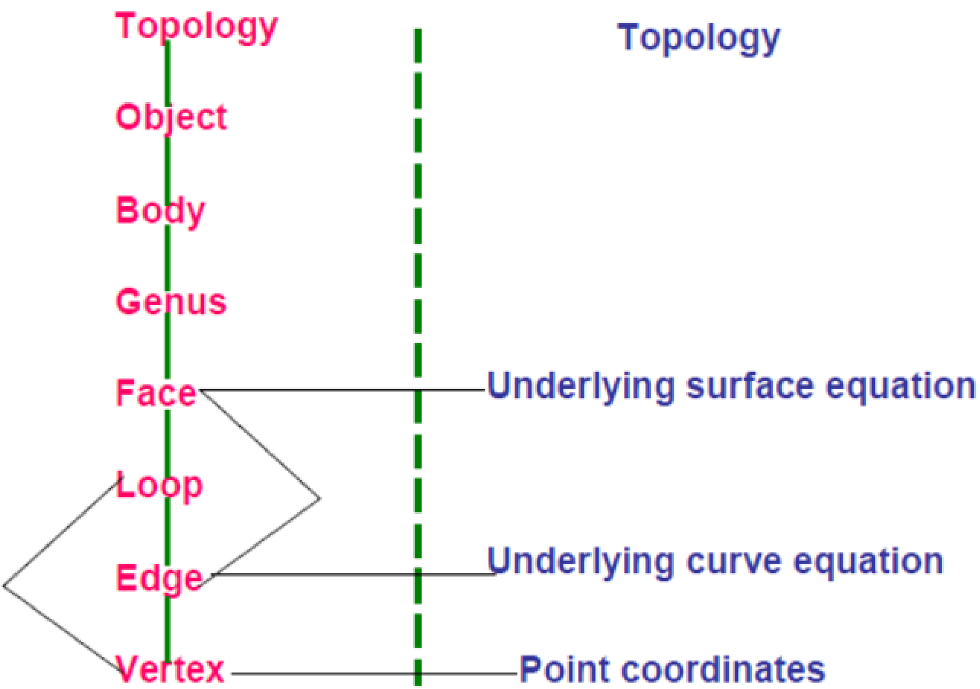
Exact B-rep: Cylinder and Sphere

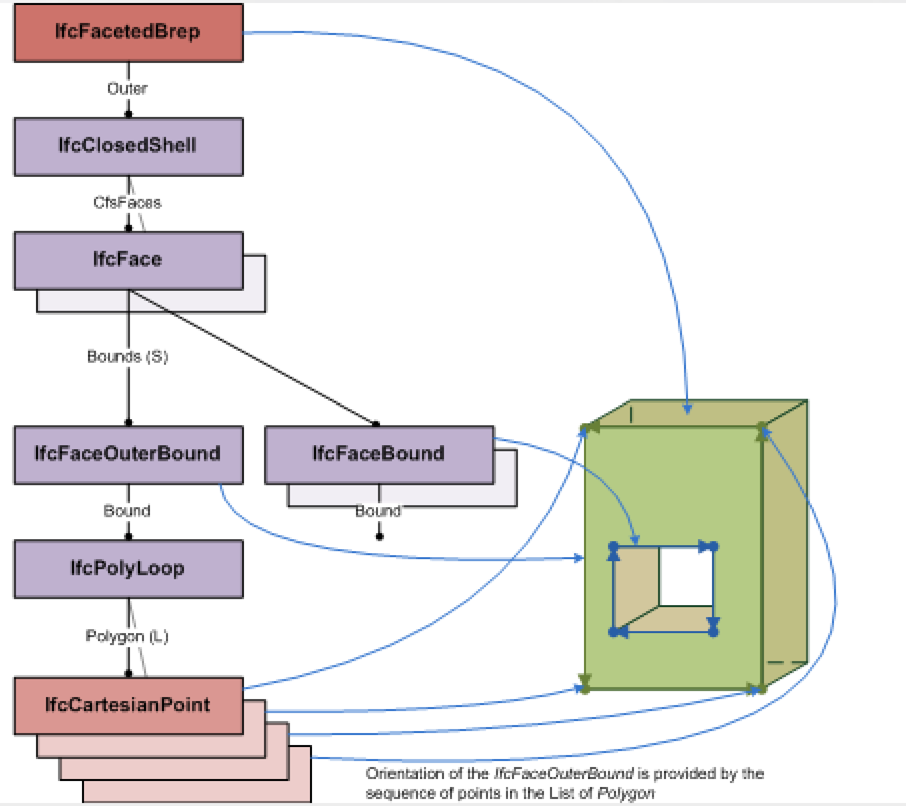


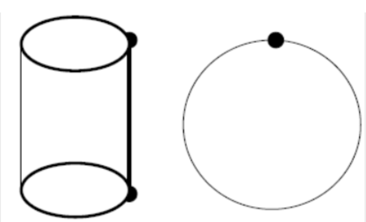
Faceted cylinder and sphere

Data structure for B-rep models

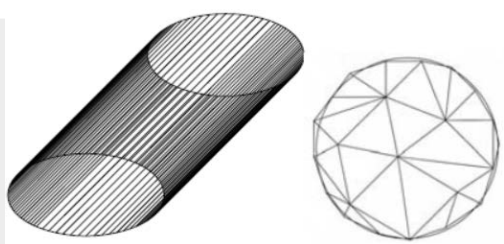
$$F - E + V - L = 2(B - G) \quad \text{General}$$







Exact B-rep: Cylinder and Sphere



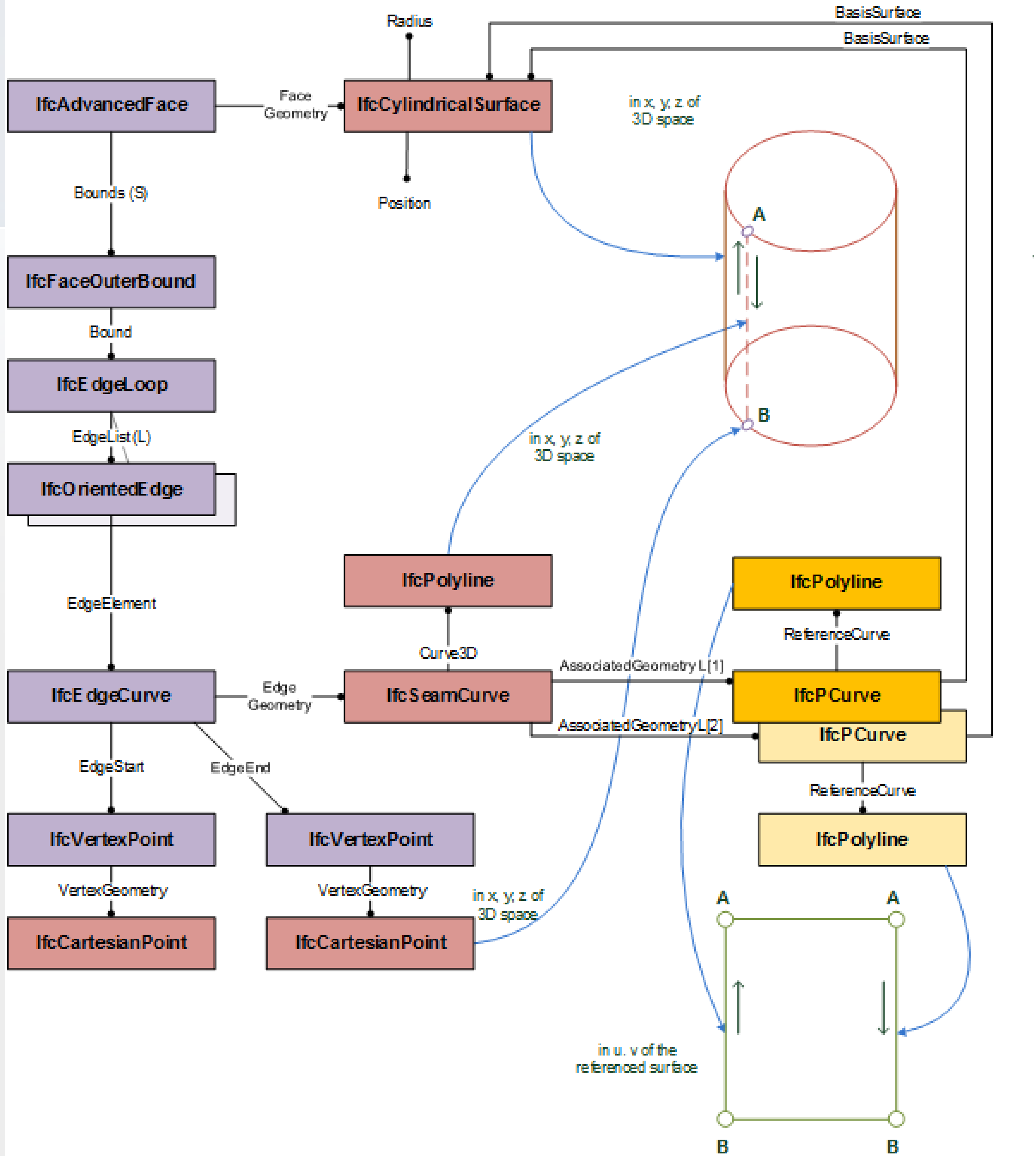
Faceted cylinder and sphere

Faceted B-rep

IfcSeamCurve

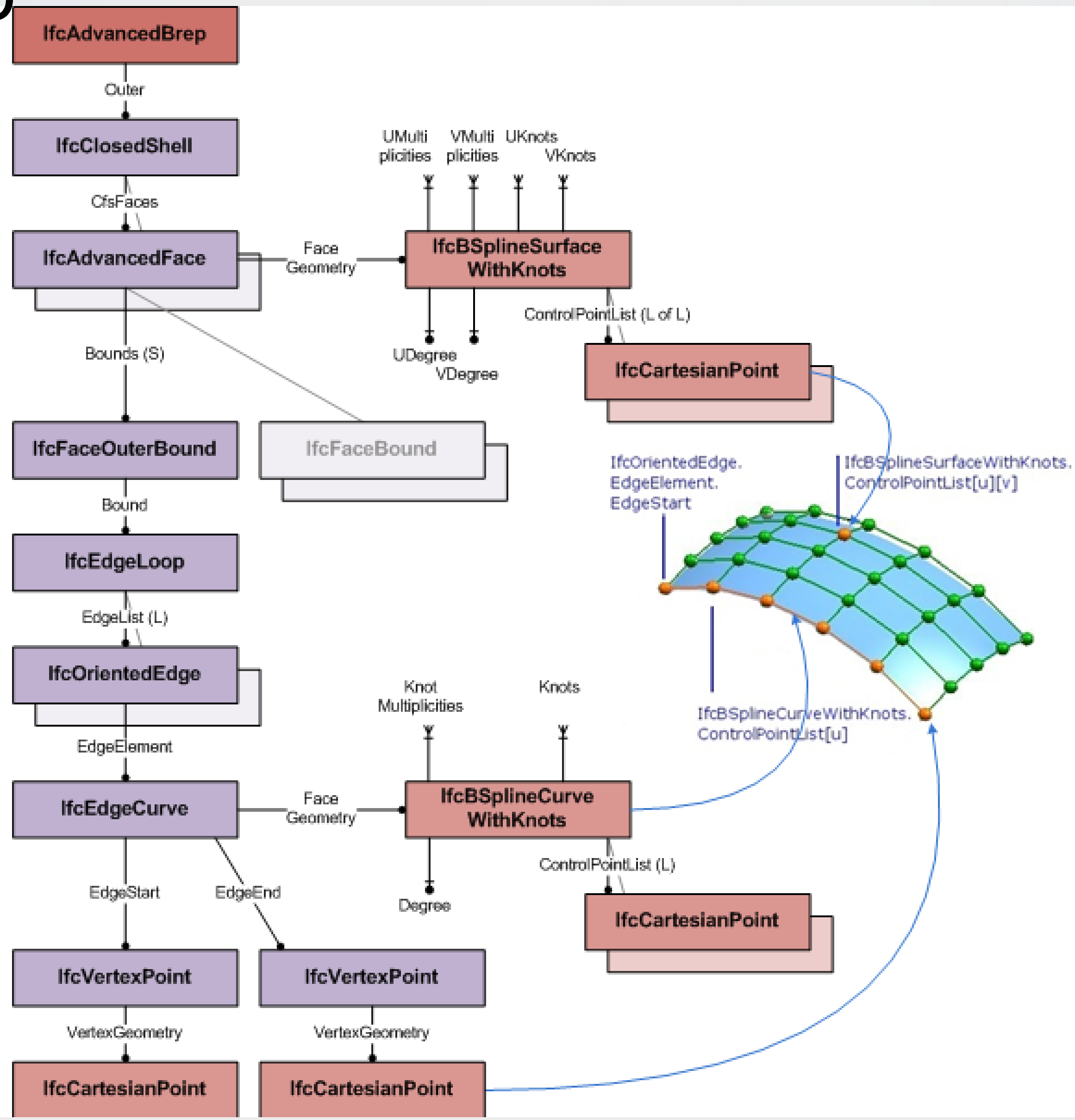
SeamCurve entity definition in B-rep

Use of a Seam Curve bounding a cylindrical surface



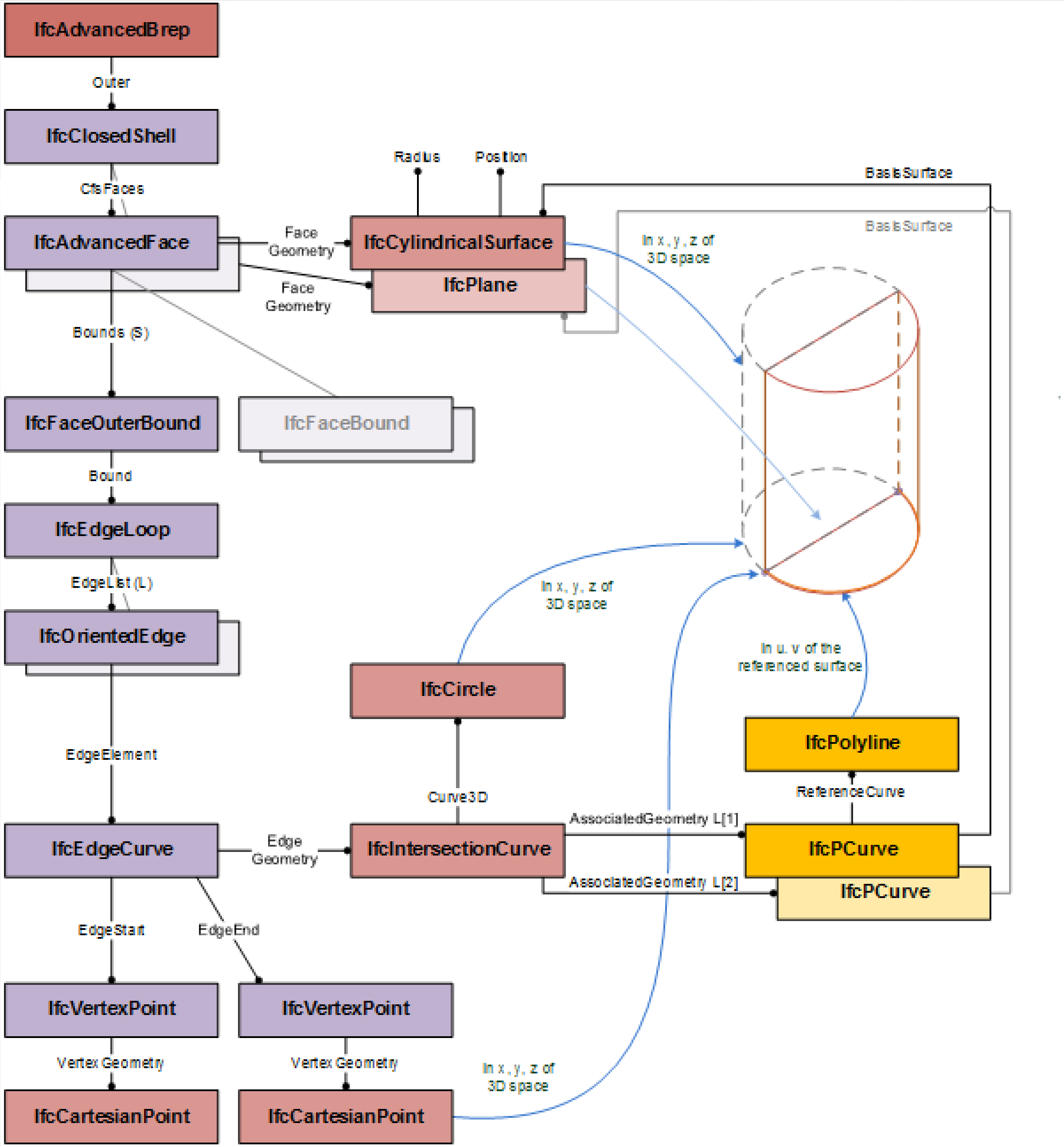
Advanced B-rep

The diagram shows the topological and geometric representation items that are used for advanced B-reps, based on IfcAdvancedFace.

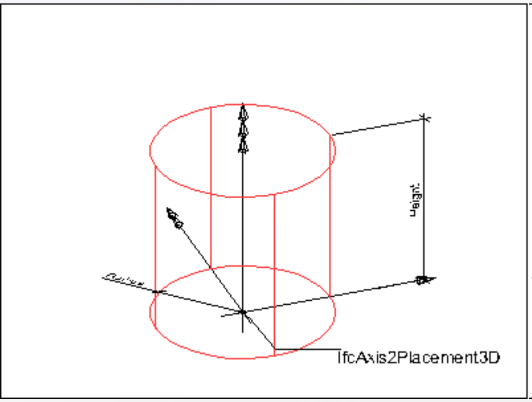


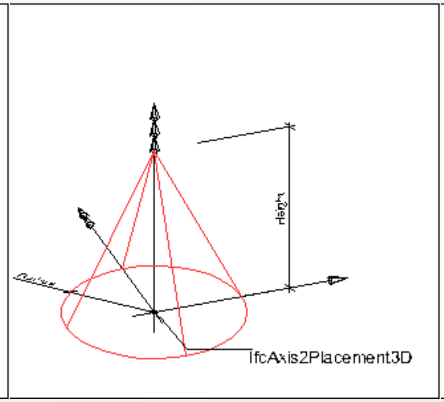
OrientedEdge Advanced B-rep

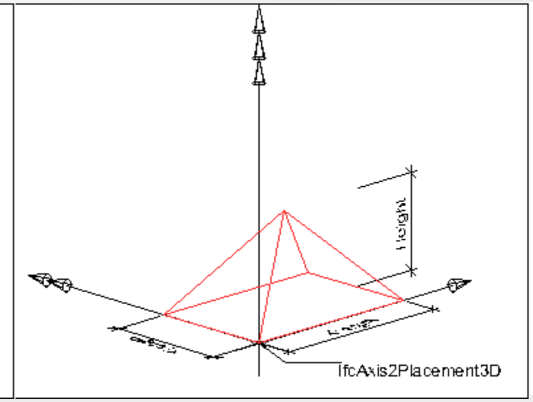
The diagram shows the topological and geometric representation items that are used for advanced B-reps, based on IfcAdvancedFace.

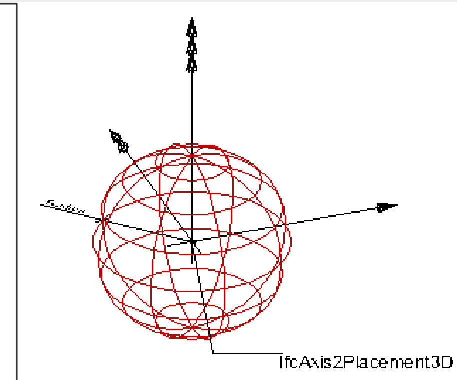


Right circular cone and cylinder geometry



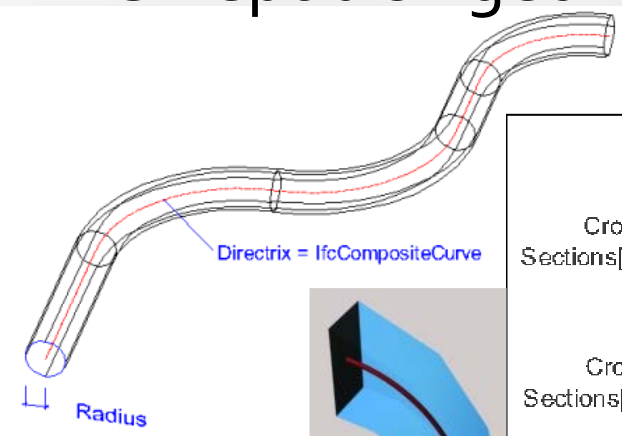


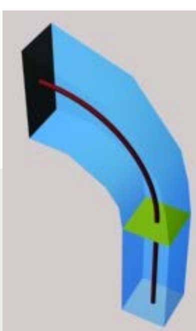


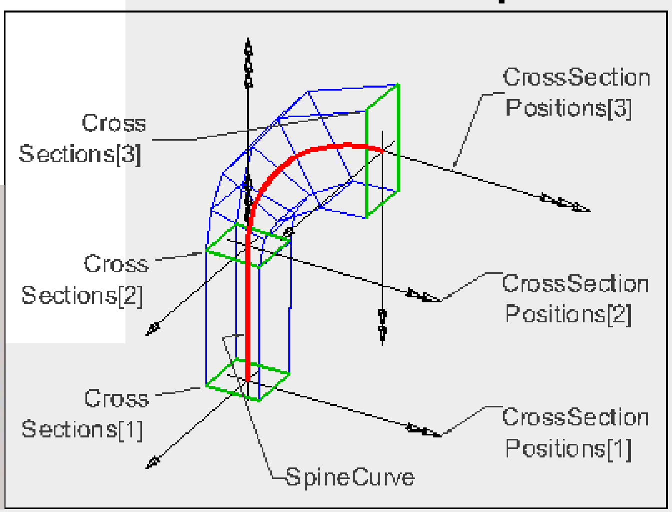


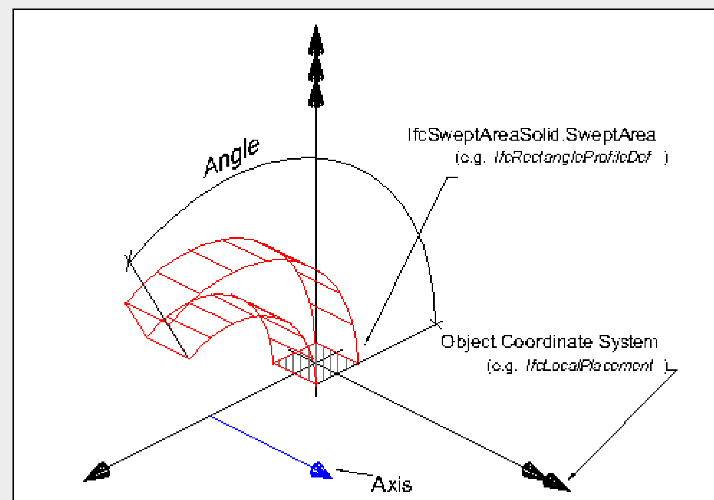
Swept disk geometry rectangular pyramid Sphere

Sectioned spine Revolved area





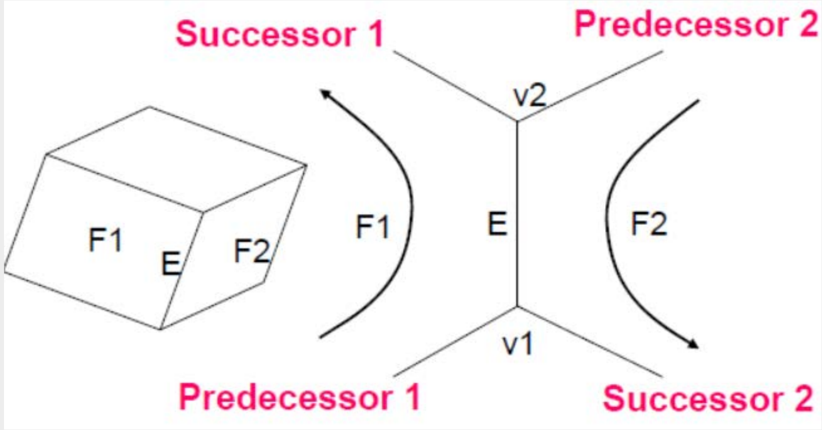




Winged Edge Data structure

All the adjacency relations of each edge are described explicitly. An edge is adjacent to exactly two faces and hence it is component in two loops, one for each face.

As each face is orientable, edges of the loops are traversed in a given direction. The winged edge data structure is efficient in object modifications (addition, deletion of edges, Euler operations).



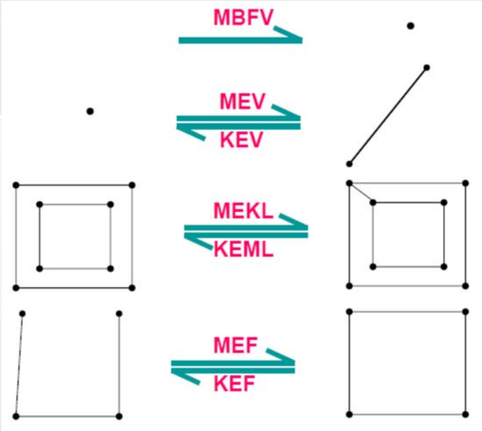
Building Operations

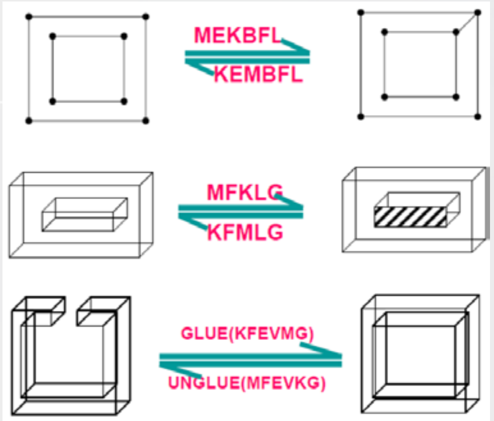
$$F - E + V - L = 2(B - G) \qquad \text{General}$$

The basis of the Euler operations is the above equation. M and K stand for Make and Kill respectively.

Operation	Operator	Complement	Description
Initiate Database and begin creation	MBFV	KBFV	Make Body Face Vertex
Create edges and vertices Create edges and faces	MEV MEKL MEF MEKBFL MFKLG	KEV KEML KEF KEMBFL KFMLG	Make Edge Vertex Make Edge Kill Loop Make Edge Face Make Edge Kill Body, Face Loop Make Edge Kill Loop Genus
Glue	KFEVMG KFEVB	MFEVKG MFEVB	Kill Face Edge Vertex Make Genus Kill Face Edge Vertex Body
Composite Operations	MME ESPLIT KVE	KME ESQUEEZE	Make Multiple Edges Edge Split Kill Vertex Edge

Transition States of Euler Operations





$$F - E + V - L = 2(B - G) \quad \text{General}$$

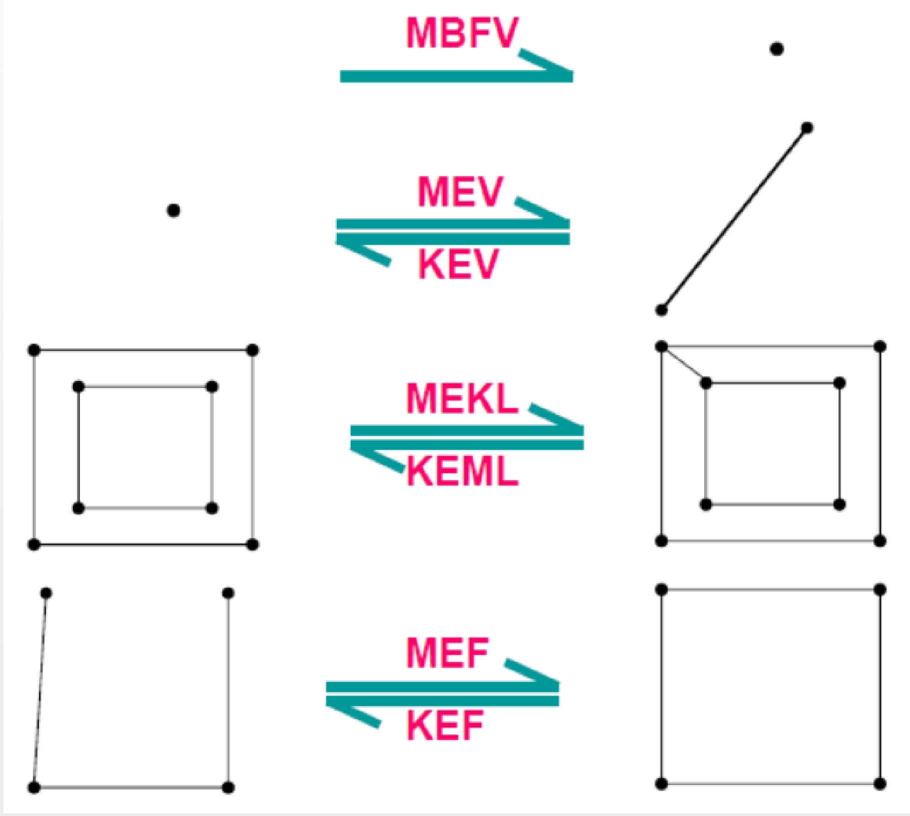
While creating B-rep models at each stage we use Euler operators and ensure the validity.

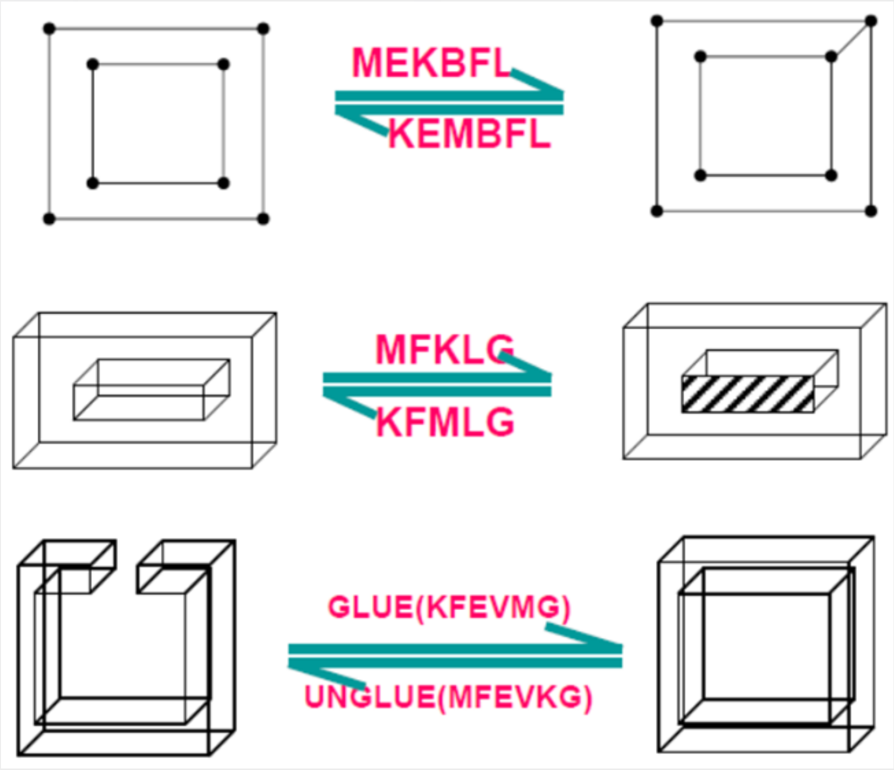
Operator	F	E	V	L	B	G
MBFV	1	0	1	0	1	0
MEV	0	1	1	0	0	0
MEKL	0	1	0	-1	0	0
MEF	1	1	0	0	0	0
MEKBFL	-1	1	0	-1	-1	0
MFKLG	1	0	0	-1	0	-1
KFEVMG	-2	-n	-n	0	0	1
KFEVB	-2	-n	-n	0	-1	0
MME	0	n	n	0	0	9
ESPLIT	0	1	1	0	0	9
KVE	-(n-1)	-n	-1	0	0	9

Operator	Complement	Description
MBFV	KBFV	Make Body Face Vertex
MEV	KEV	Make Edge Vertex
MEKL	KEML	Make Edge Kill Loop
MEF	KEF	Make Edge Face
MEKBFL	KEMBFL	Make Edge Kill Body, Face Loop
MFKLG	KFMLG	Make Edge Kill Loop Genus
KFEVMG	MFEVKG	Kill Face Edge Vertex Make Genus
KFEVB	MFEVB	Kill Face Edge Vertex Body
MME	KME	Make Multiple Edges
ESPLIT	ESQUEEZE	Edge Split
KVE		Kill Vertex Edge

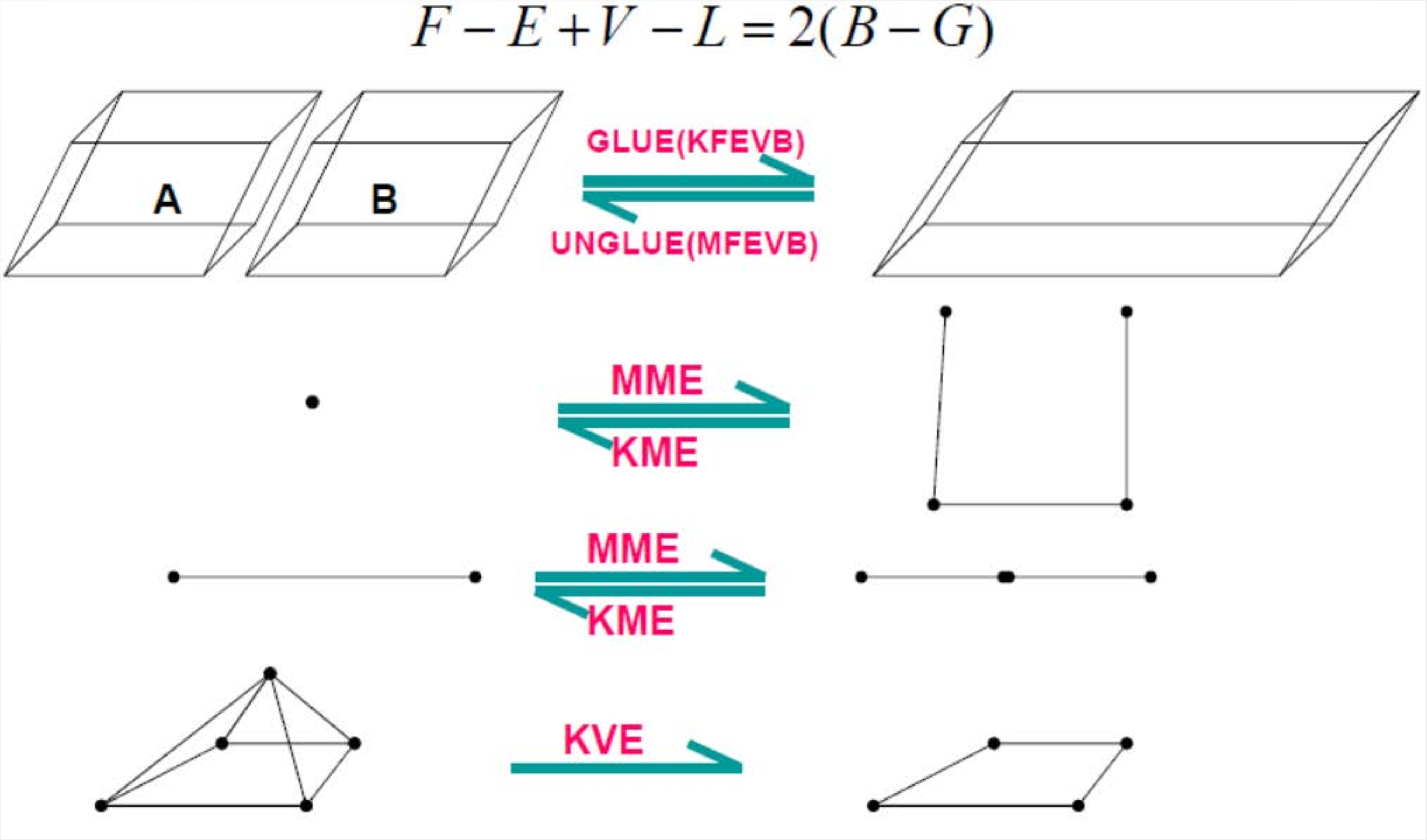
Euler Operations

$$F - E + V - L = 2(B - G)$$

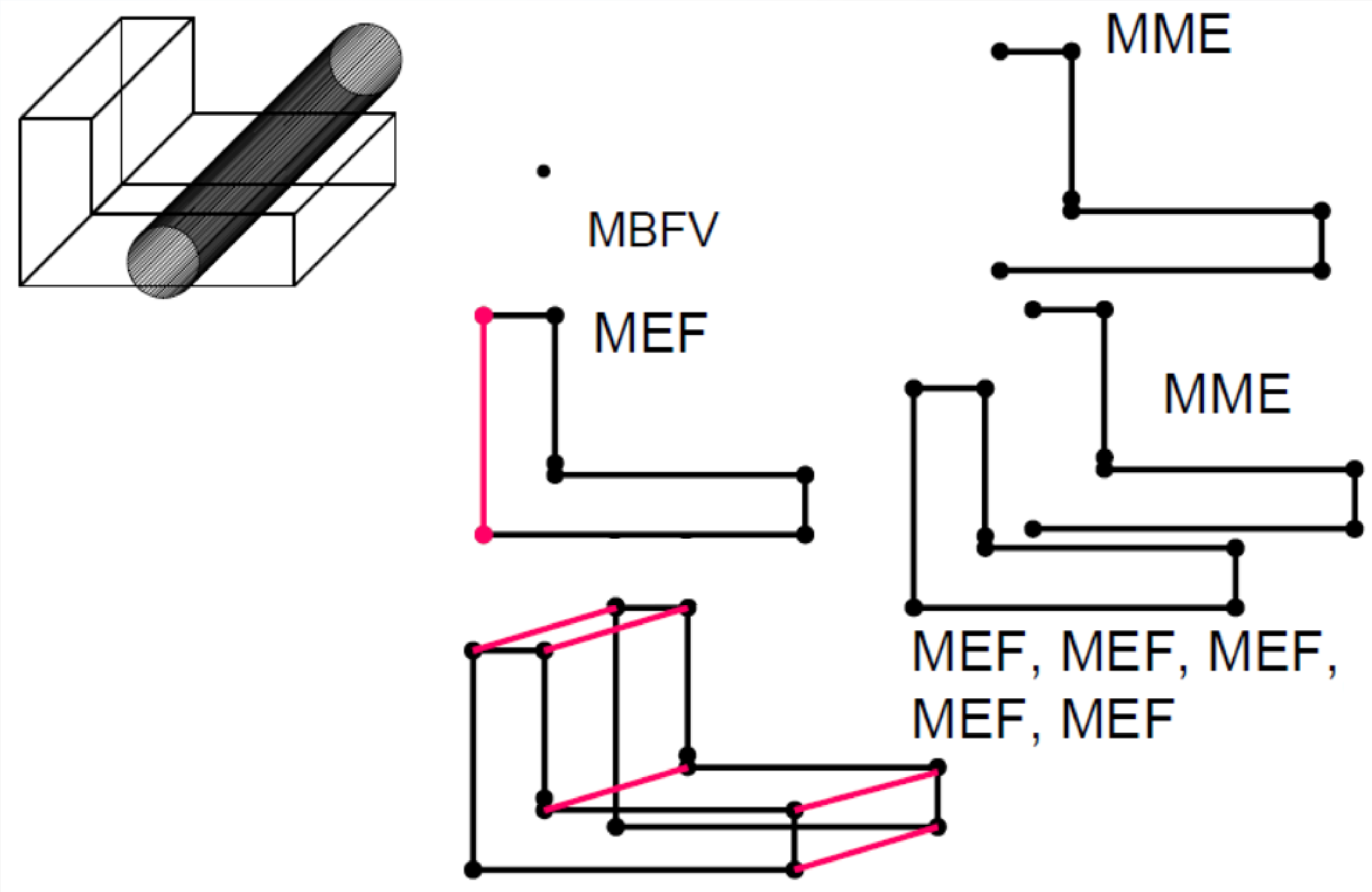




Euler Operations



Building operations



Merits and Demerits of Euler Operations

If the operator acts on a valid topology and the state transition it generates is valid, then the resulting topology is a valid solid. Therefore, Euler's law is never verified explicitly by the modeling system.

Merits:

- They ensure creating valid topology
- They provide full generality and reasonable simplicity
- They achieve a higher semantic level than that of manipulating faces, edges and vertices directly

Demerits :

- They do not provide any geometrical information to define a solid polyhedron
- They do not impose any restriction

Advantages and Disadvantages of B-rep

Advantages :

- It is historically a popular modeling scheme related closely to traditional drafting
- It is very appropriate tool to construct quite unusual shapes like aircraft fuselage and automobile bodies that are difficult to build using primitives
- It is relatively simple to convert a B-rep model into a wireframe model because its boundary definition is similar to the wireframe definitions
- In applications B-rep algorithms are reliable and competitive to CSG based algorithms

Disadvantages :

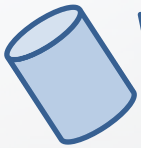
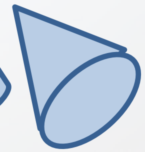
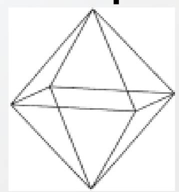
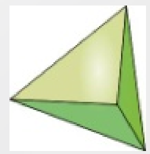
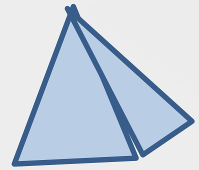
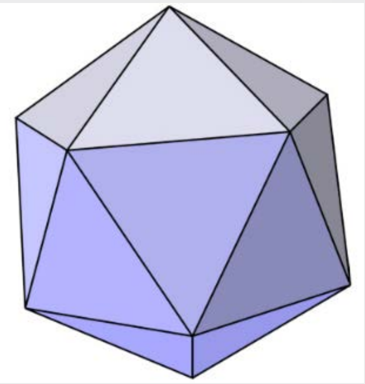
- It requires large storage space as it stores the explicit definitions of the model boundaries
- It is more verbose than CSG
- Faceted B-rep is not suitable for manufacturing applications

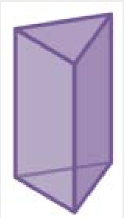
Boundary Representation (B-Rep)

- Euler's rule applies of a simple polyhedron:

  $V - E + F = 2$

  where
  V = number of vertices,
  E = number of edges,
  F = number of faces.

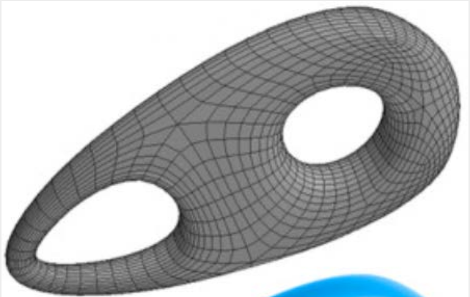







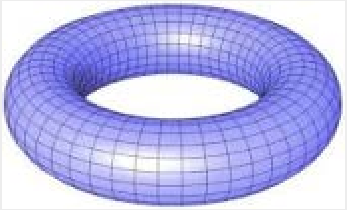



- Euler-Poincare topological equation for solid with hole:

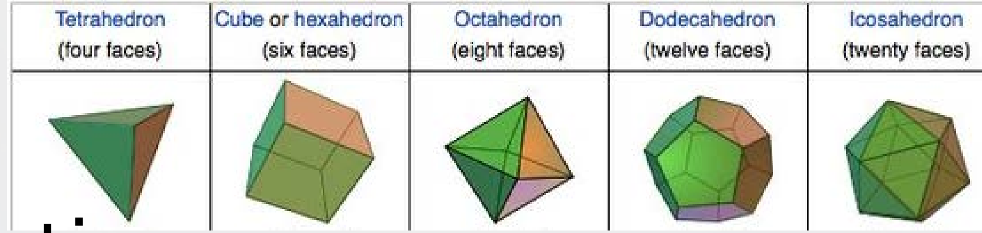
  $V - E + F - (L - F) - 2(S - G) = 0$

  where L = number of edge loops,
  S = number of shells,
  G = genus of solid (holes).





- Surface must be **closed**



Boundary Representation

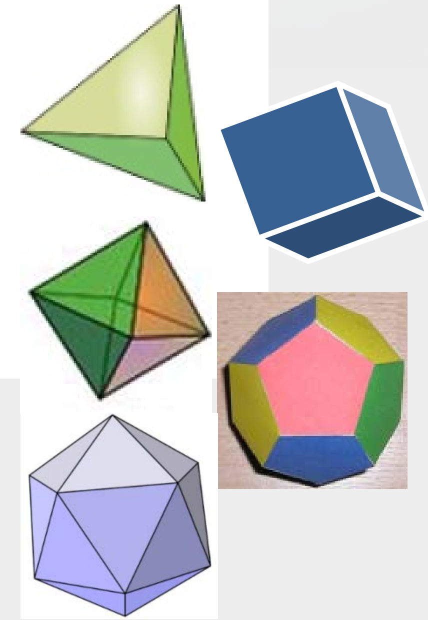
Boundary/surface contains
0D vertices, 1D edges, 2D faces

There are 5 regular polyhedrons.

Euler's Formula for regular polyhedrons

$V - E + F = 2$

p	v	(p-2)(v-2)	Name	Description
3	3	1	Tetrahedron	3 triangles at each vertex
4	3	2	Cube	3 squares at each vertex
3	4	2	Octahedron	4 triangles at each vertex
5	3	3	Dodecahedron	3 pentagons at each vertex
3	5	3	Icosahedron	5 triangles at each vertex



	Face polygons	vertices	Edges	Faces	Faces at a vertex
Tetrahedron	Triangles	4	6	4	3
Cube	Squares	8	12	6	3
Octahedron	Triangles	6	12	8	4
Dodecahedron	Pentagons	20	30	12	3
Icosahedron	Triangles	12	30	20	5

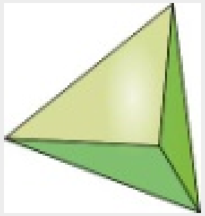
Euler’s formula for regular polyhedra

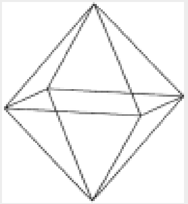
We can determine all possible regular polyhedra; that is, those polyhedra with every **face** having the same number of edges, say, **h**; with every **vertex** having the same number of edges emanating from it, say, **k**; and every **edge** having the same length. Since every **edge has two vertices** and belongs to exactly two faces, it follows that **Fh=2E=Vk** . Substitute this into Euler’s formula: (page.294, Geometric modeling, Mortenson, 1996)

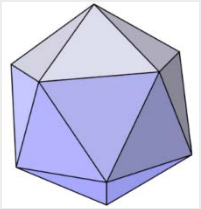
$$V - E + F = 2$$

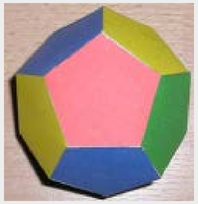
$$\frac{2E}{k} - E + \frac{2E}{h} = 2$$

$$\frac{1}{E} = \frac{1}{h} + \frac{1}{k} - \frac{1}{2}$$







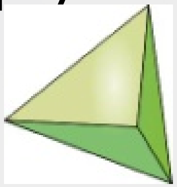


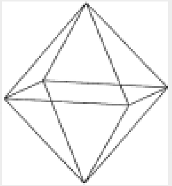
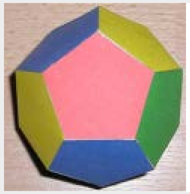
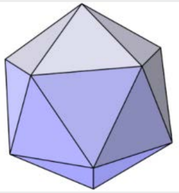
$$V - E + F = 2 \qquad \frac{2E}{k} - E + \frac{2E}{h} = 2 \qquad \frac{1}{E} = \frac{1}{h} + \frac{1}{k} - \frac{1}{2}$$

Euler's formula for regular polyhedra

For a polyhedron, we safely assume that **h, k ≥ 3**. On the other hand, both **h** and **k** were larger than 3, then the above equation would imply that

$$0 < \frac{1}{E} = \frac{1}{h} + \frac{1}{k} - \frac{1}{2} \leq \frac{1}{4} + \frac{1}{4} - \frac{1}{2} = 0$$

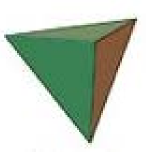
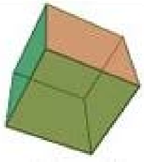
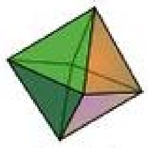
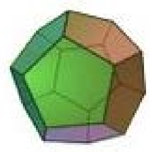
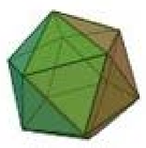


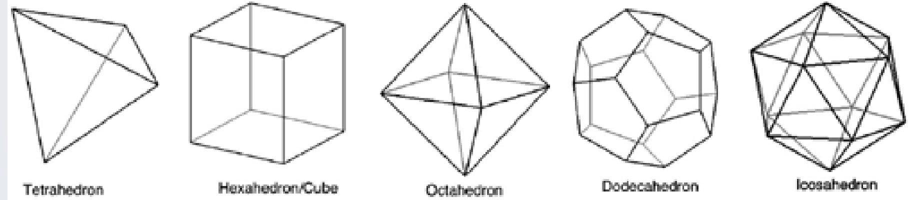
which is obviously impossible. Therefore, either h or k equals 3. If h=3, then $0 < \frac{1}{E} = \frac{1}{3} + \frac{1}{k} - \frac{1}{2}$

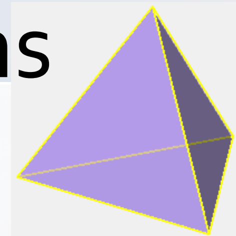
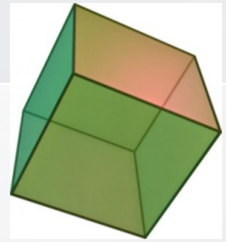
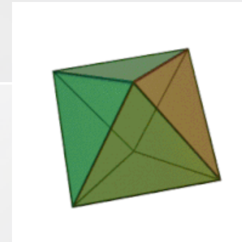
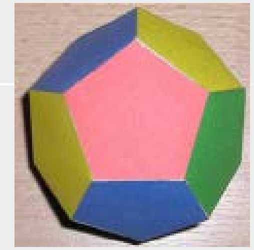
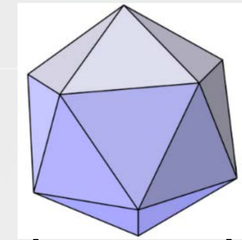
implies that **3≤k≤ 5** . By symmetry, if **k=3**, then **3≤h≤5** . Thus, (h,k,E) =(3,3,6), (4,3,12), (3,4,12), (5,3,30), (3,5,30) are only possibilities.

	Face polygons	vertices	Edges	Faces	Faces at a vertex
Tetrahedron	Triangles	4	6	4	3
Cube	Squares	8	12	6	3
Octahedron	Triangles	6	12	8	4
Dodecahedron	Pentagons	20	30	12	3
Icosahedron	Triangles	12	30	20	5

Tetrahedron (four faces)	Cube or hexahedron (six faces)	Octahedron (eight faces)	Dodecahedron (twelve faces)	Icosahedron (twenty faces)
				

Regular polyhedrons



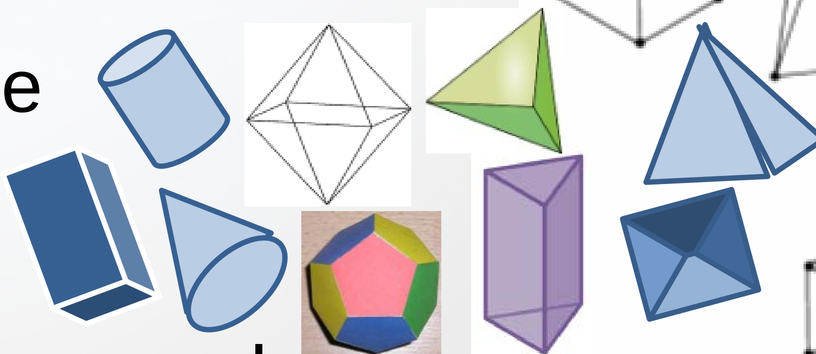
Thus, (h,k,E) =(3,3,6), (4,3,12), (3,4,12), (5,3,30), (3,5,30) are only possibilities. They are, in fact, realized by the tetrahedron, the cube (hexahedron), the octahedron, the dodecahedron, and the icosahedron, respectively.

Observe that we did not really use the fact that the edges of the polyhedron all have the same length. As long as the numbers h and k are constant, we still have only five possibilities (up to stretching or contracting).

Boundary Representation (B-Rep)

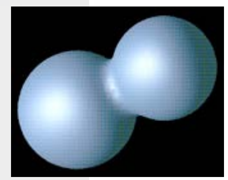
- Euler's rule

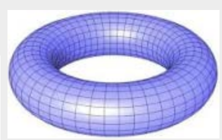
  V-E+F=2



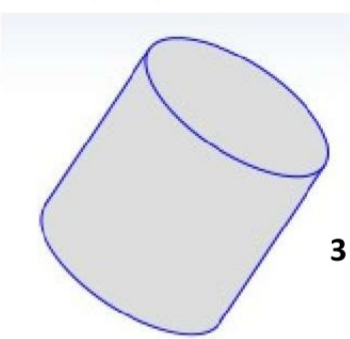
- Euler-Poincare rule

  V-E+F-(L-F)-2(S-G)=0



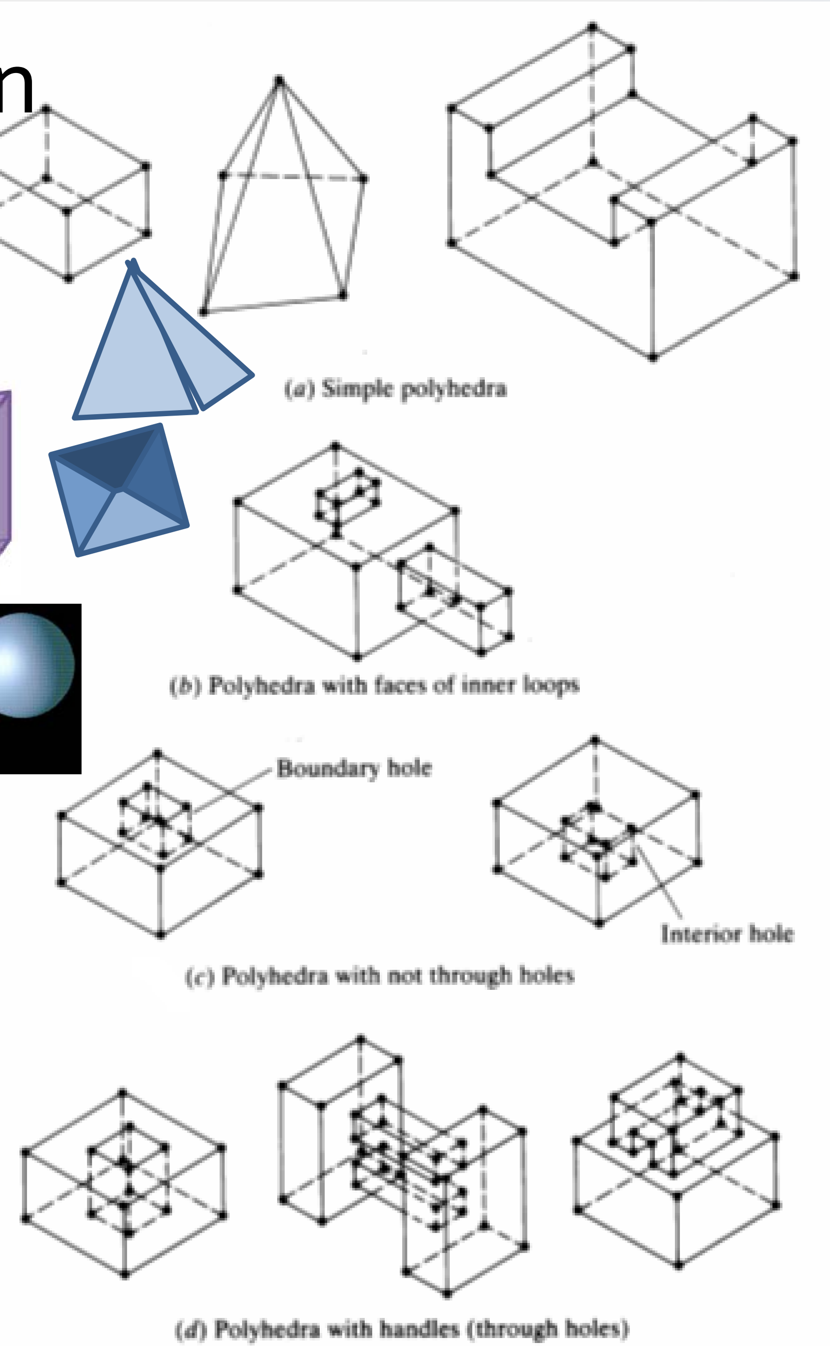


- The extended Euler-Poincarré formula allow test the topology for polyhedral solids:

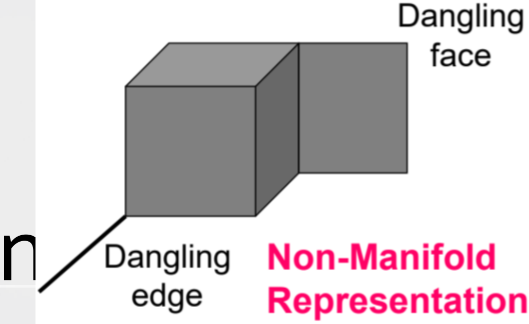


Faces = 3
Vertices = 0
Edges = 2

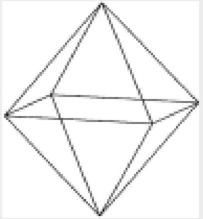
3 + 0 – 2 – 0 ... 2 (1 – 0)

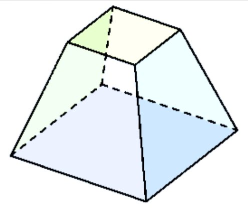


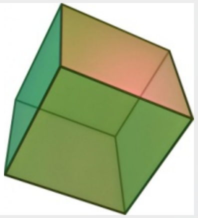
Euler’s rule for simple polyhedron



Euler’s rule $V - E + F = 2$ for simple polyhedron.
Applying this formula to a cube yields 8 - 12 + 6 = 2 and to an octahedron yields 6 - 12 + 8 = 2

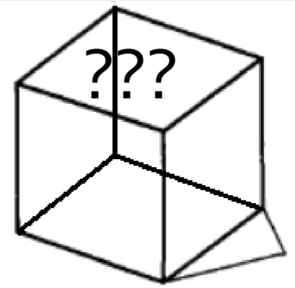






To apply Euler's formula,
other conditions must also be met:

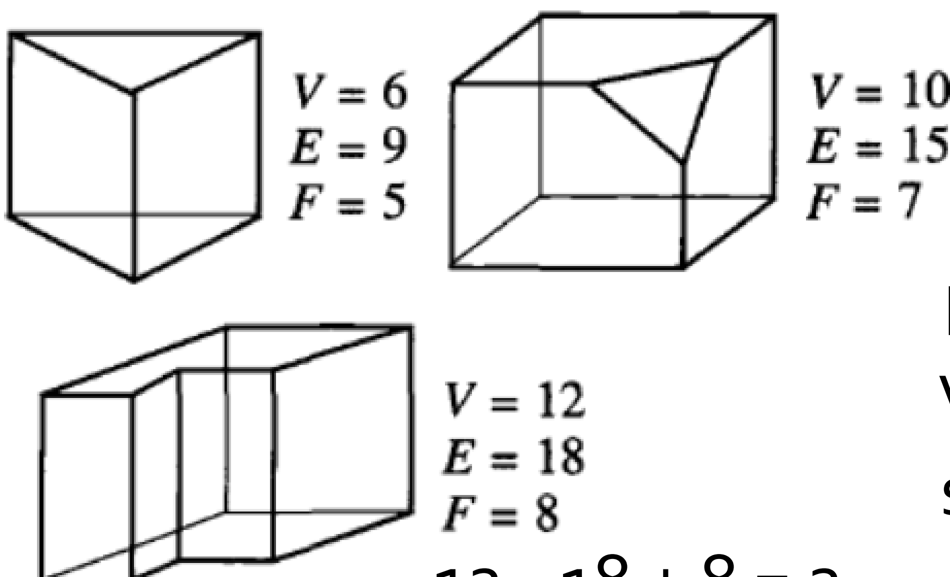
1. All faces must be bounded by a single ring of edges, with no holes in the faces.
2. The polyhedron must have no holes through it.
3. Each edge is shared by exactly two faces and is terminated by a vertex at each end.
4. At least three edges must meet at each vertex.



Euler’s rule

The polyhedra in Figure satisfy the four conditions and, therefore, Euler's formula applies.

6 - 9 + 5 = 2 , 10 - 15 + 7 = 2

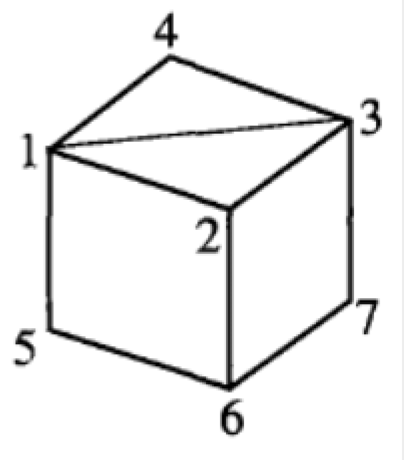


12 - 18 + 8 = 2

Figure.
Vertices, edges, and faces satisfying Euler's formula.

Euler’s rule

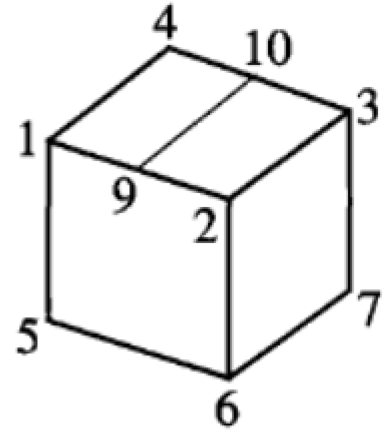
If we add vertices, edges, or faces to a polyhedron, we must do so in a way that satisfies Euler's formula and the four conditions. In Figure (a) we add an edge, joining vertex 1 to vertex 3 and dividing face 1, 2, 3, 4 into two separate faces. We have added one face and one edge. These additions produce no net change to Euler's formula (since 0 - 1 + 1 = 0).



(a)

Euler’s rule

In Figure (b) we add vertices 9 and 10 and join them with an edge. The new vertices divide edges 1, 2, 3, 4, and the new edge 9, 10 divides face 1, 2, 3, 4. These changes, too, produce no net change to Euler's formula (since 2 - 3 + 1 = 0).

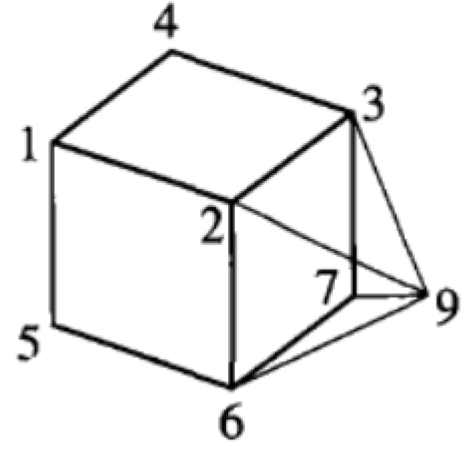


(b)

Euler’s rule

In Figure (c) we add one vertex, four edges, and four faces, but we delete the existing Face 2, 6, 7, 3.

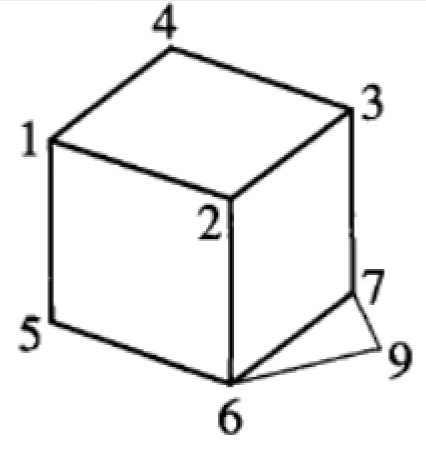
Again, this action produces no net change to Euler's formula (since 1- 4 + 3 = 0).



(c)

Euler's rule

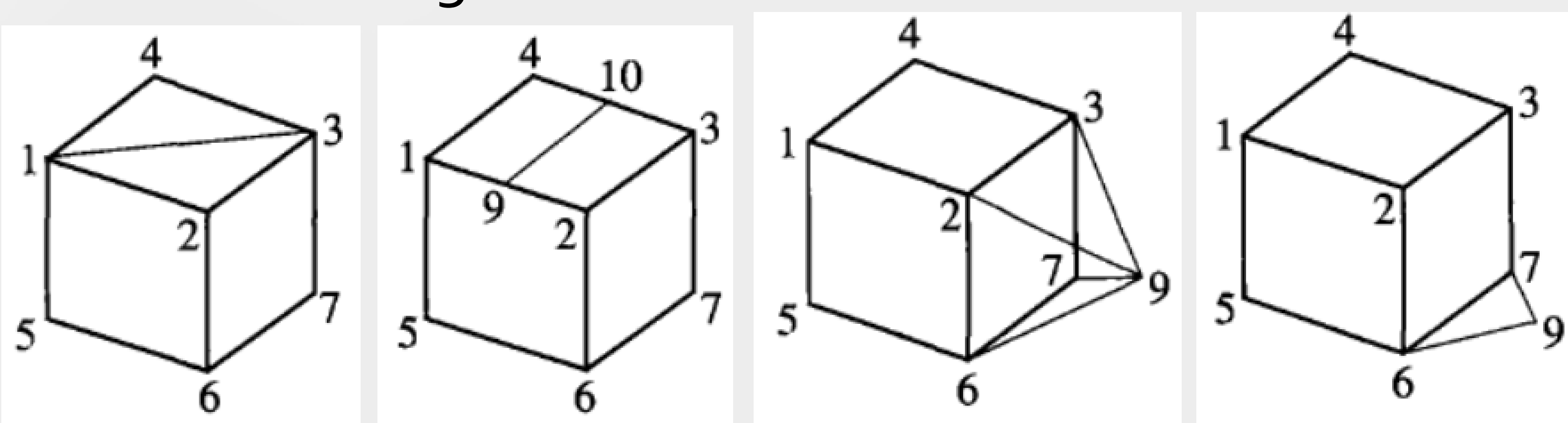
In Figure (d), where we attempt to add one vertex, two edges, and one face, the change is not acceptable. Although this change preserves Euler's formula (since 1- 2 + 1 = 0), it does not satisfy the conditions requiring each edge to adjoin exactly two faces and at least three edges to meet at each vertex.



(d)

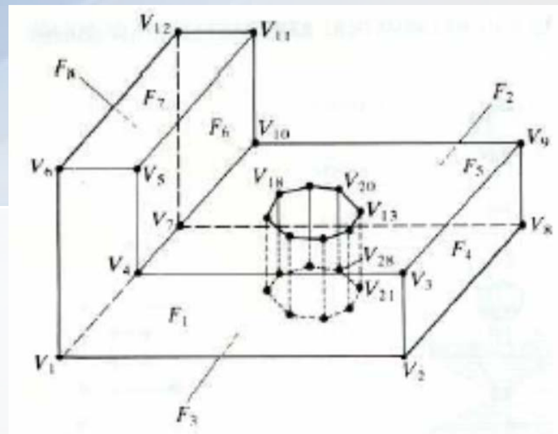
Euler’s rule

Two kinds of changes are illustrated in the figure. In Figures (a) and (b) the solid shape of the polyhedron (in this case a cube) is preserved, and only the network of vertices, edges, and faces is changed. In Figure (c) the solid shape itself is modified by the change in the network defining it.



(a) (b) (c) (d)

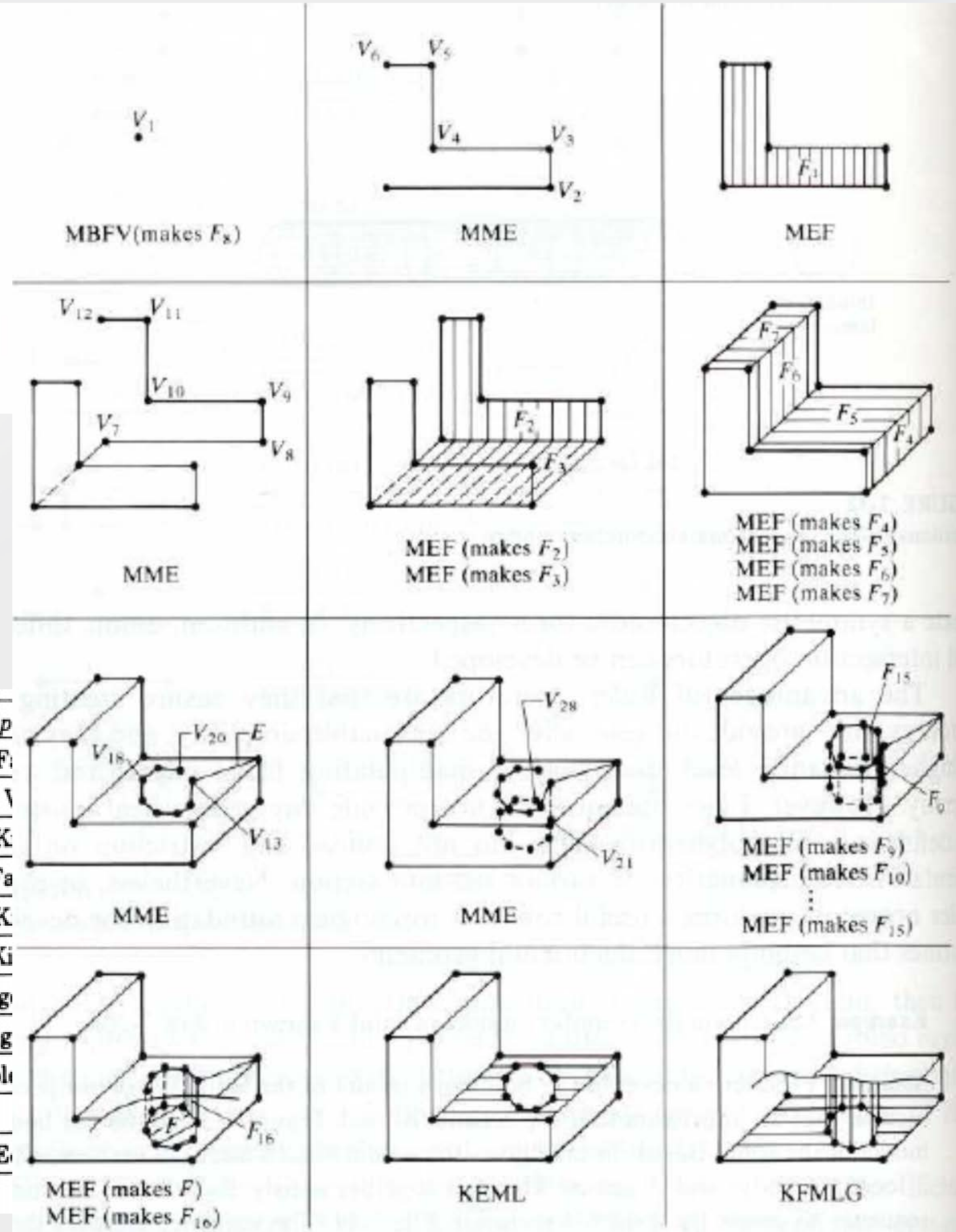
B-rep



Body, Face, Polygon
(Edge Loop), Edge, Vertex
Euler Operators

Operation	*Operator*	*Complement*	*Descrip*
Initialize database and begin creation	MBFV	KBFV	Make Body, F
Create edges and vertices	MEV	KEV	Make Edges, V
Create edges and faces	MEKL	KEML	Make Edge, K
	MEF	KEF	Make edge, Fa
	MEKBFL	KEMBFL	Make Edge, K
	MFKLG	KFMLG	Make Face, Ki
Glue	KFEVMG	MFEVKG	Kill Face, Edg
	KFEVB	MFEVB	Kill Face, Edg
Composite operations	MME	KME	Make Multipl
	ESPLIT	ESQUEEZE	Edge-Split
	KVE		Kill Vertex, E

M : Make, K : Kill

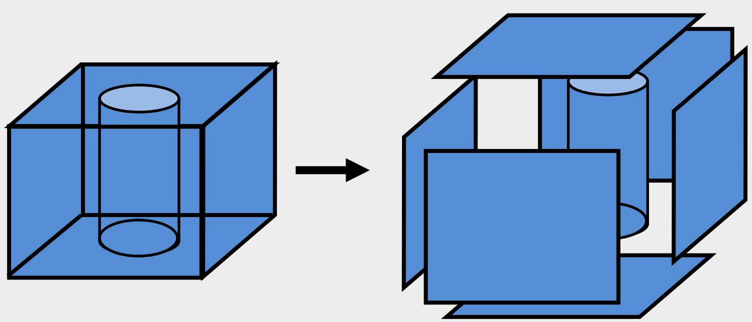


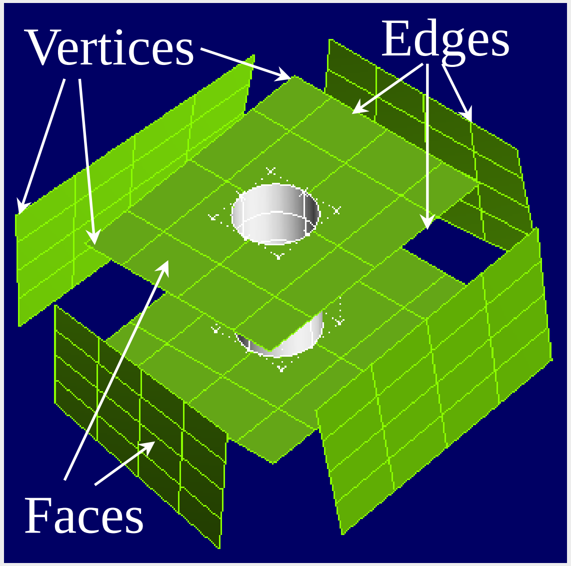
Solid B-Rep Example

Complete part representation including topological and geometrical data

Geometry: shape and dimensions

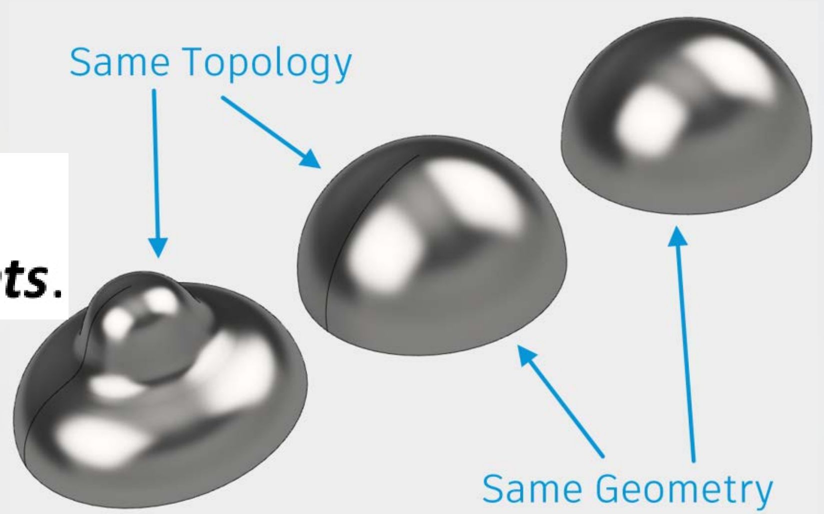
Topology: the connectivity and associativity of the object entities; it determines the relational information between object entities



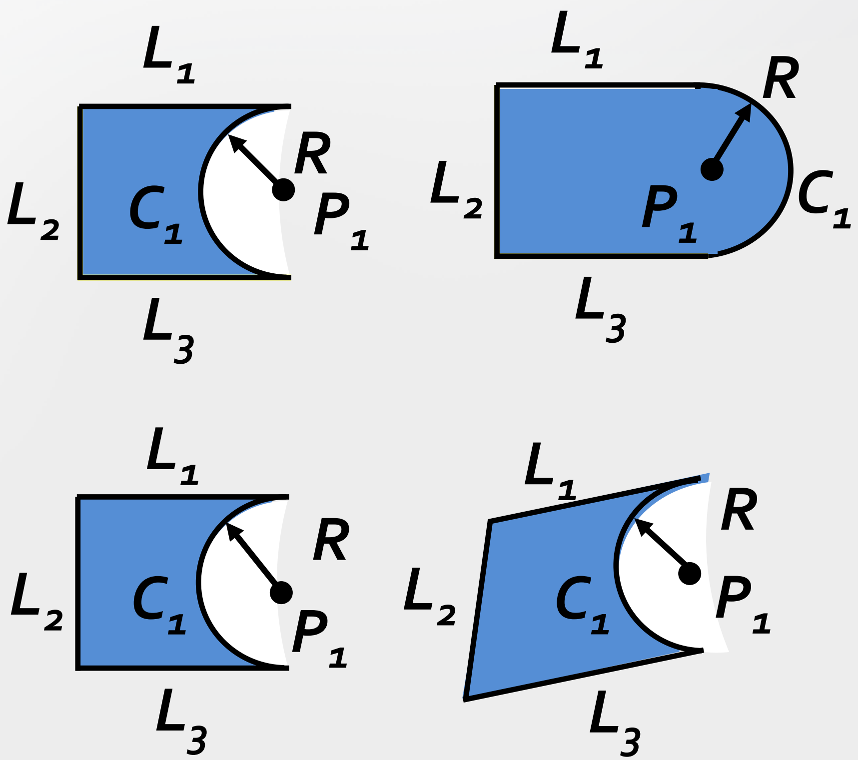


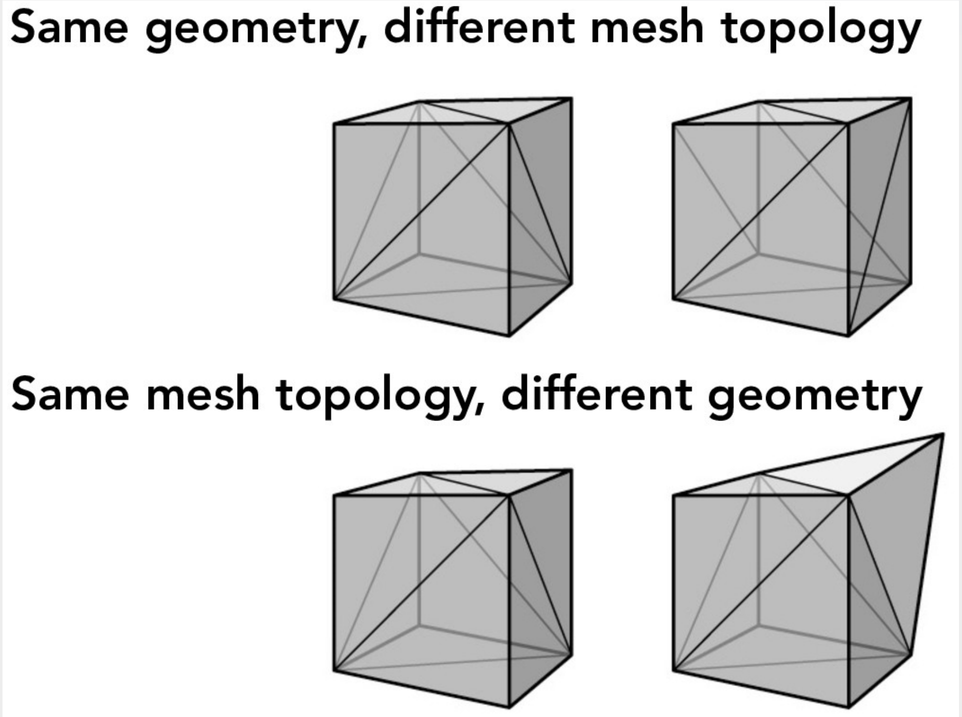
Topology vs Geometry

- Topology: ***faces***, ***edges*** and ***vertices***.
- Geometry: ***surfaces***, ***curves*** and ***points***.



Topology

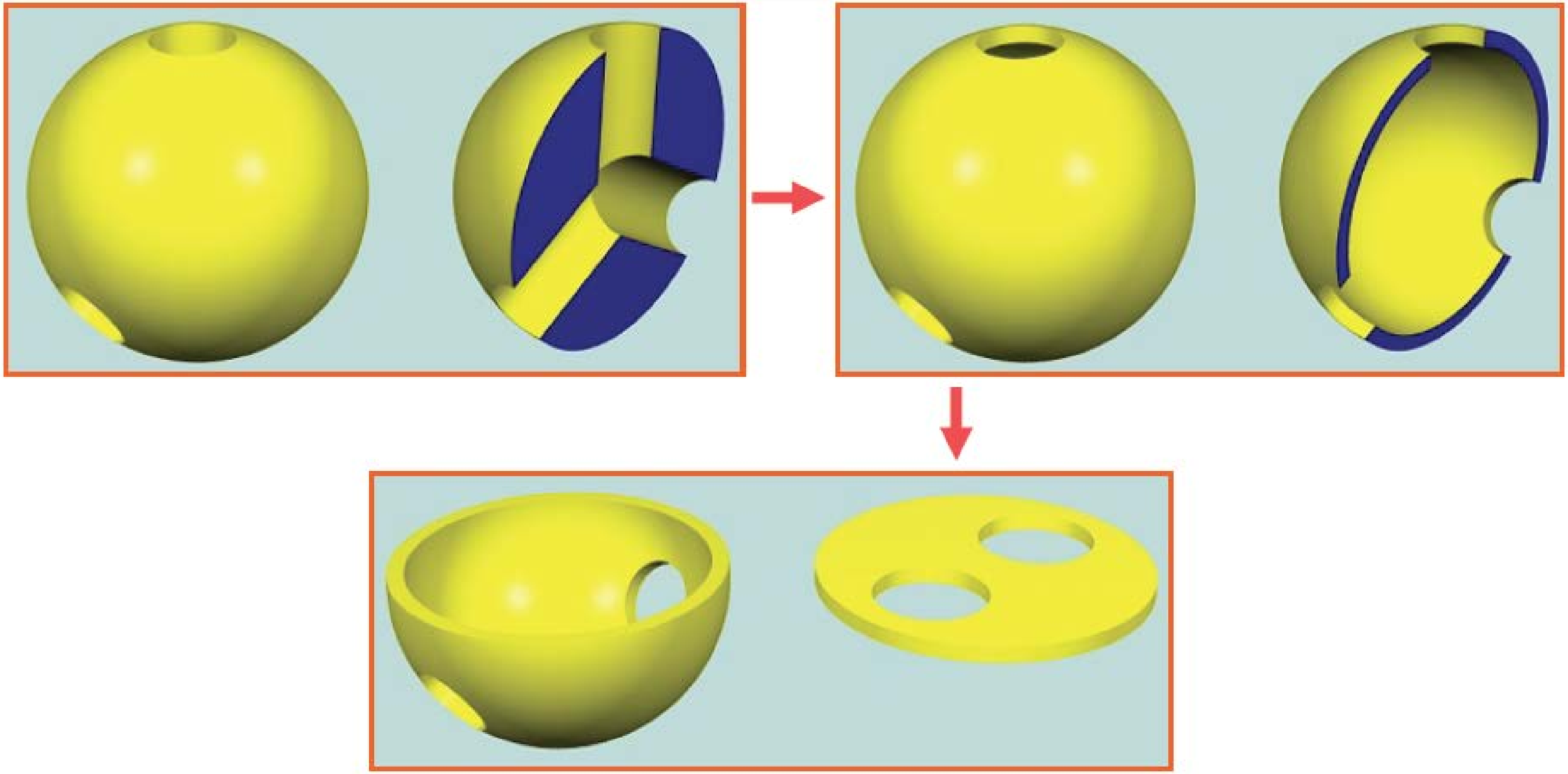




Solid B-Rep

- Complete part representation including topological and geometrical data
- Able to transfer data directly from CAD to CAE and CAM.
- Support various engineering applications, such as mass properties, mechanism analysis, FEA/FEM and tool path creation for CNC, and so on.

Sphere Punched by Three Tunnels

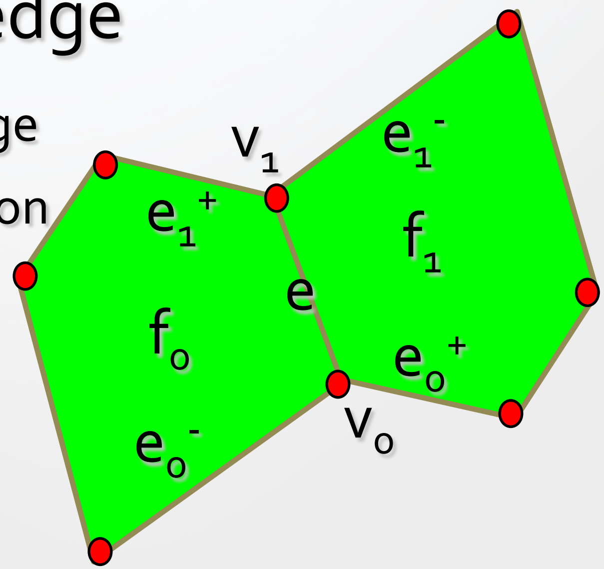


The Genus is 2

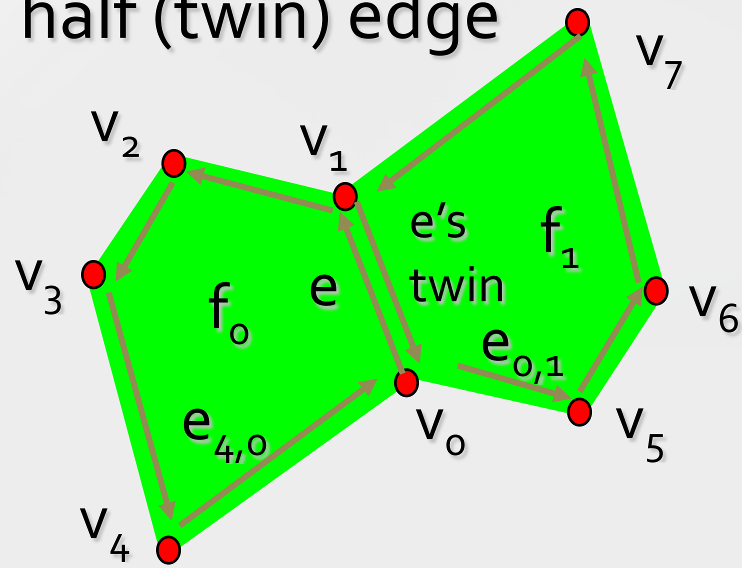
Polyhedral Boundary Representations

winged edge

- focus is on edge
- edge orientation is arbitrary



half (twin) edge

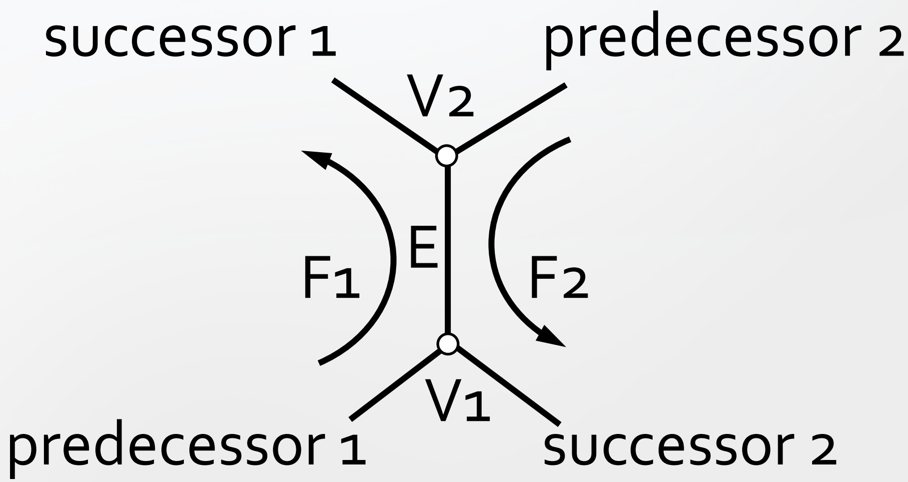


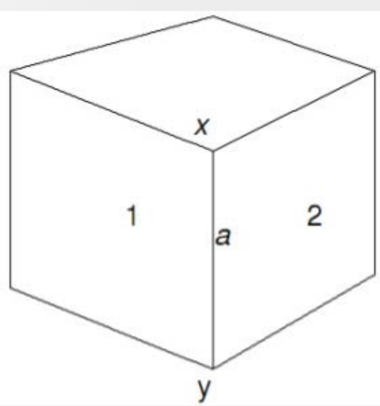
twin edge [doubly connected edge list]

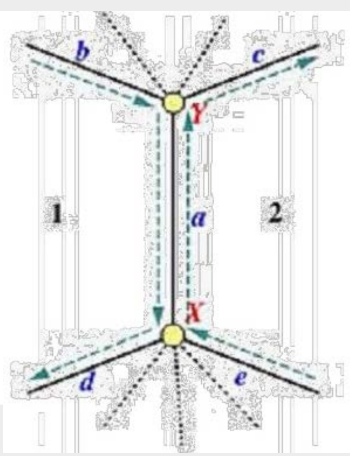
- represent edge as 2 halves
- lists: vertex, face, edge/twin
- more storage space
- facilitates face traversal
- can represent holes with face inner/outer edge pointer

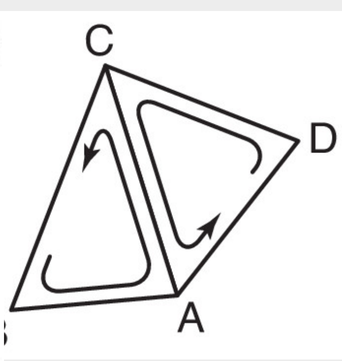
- Topology: ***faces***, ***edges*** and ***vertices***.
- Geometry: ***surfaces***, ***curves*** and ***points***.

Winged Edge Data Structure

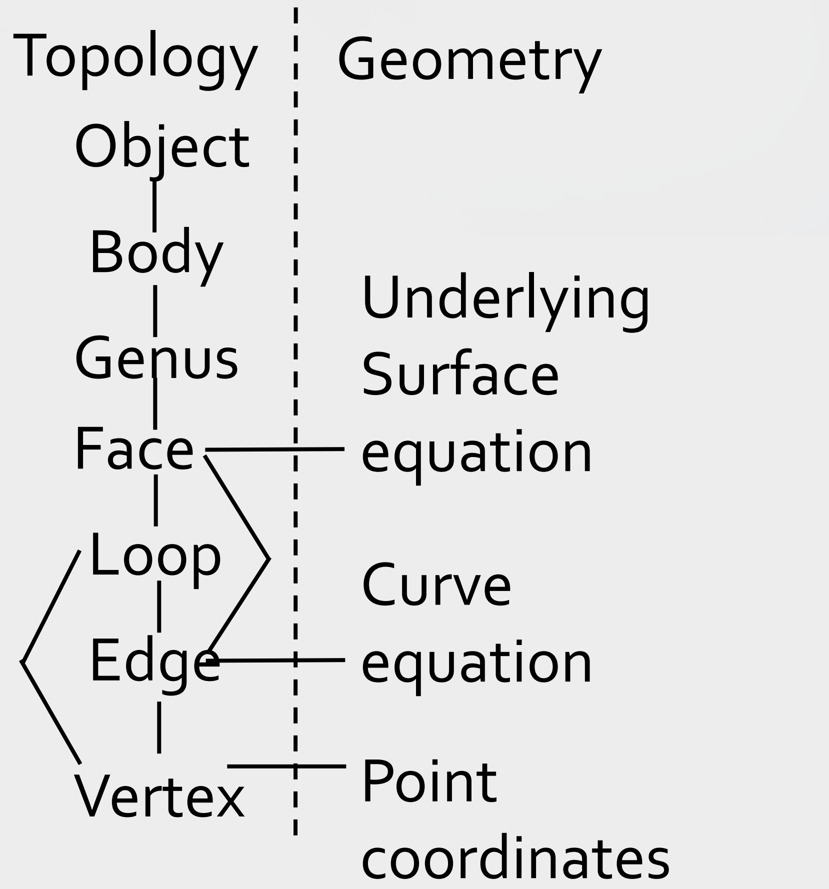








Topology and geometry

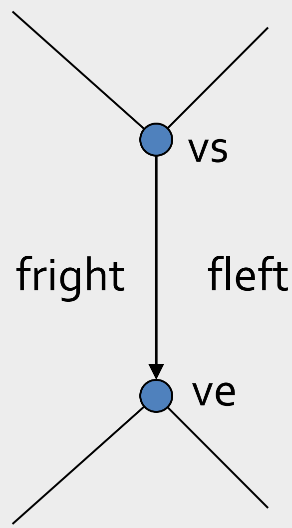


Winged Edge Data Structure

```
class Vertex {
    Vec3 pos;
public:
    Vertex() {pos = vl_0;}
    Vertex (double x, double y, double z) {pos = Vec3(x,y,z);}
    void setpos (double x, double y, double z) {pos = Vec3(x,y,z);}
    void printpos() {cout << pos << endl;}
};
```

```
class Model {
public:
    vector<Vertex*> vs;
    vector<Edge*>   es;
    vector<Face*>   fs;
};
```

```
class Edge {
    Vertex *vs, *ve;
    Face *fleft, *fright;
public:
    Edge() {fleft = fright = NULL;};
    Edge (Vertex *v1, Vertex *v2) {vs = v1, ve = v2, fleft = fright = NULL;}
    void setLface(Face* f) {fleft = f;}
    void setRface(Face* f) {fright = f;}
    Vertex* startV() {return vs;}
    Vertex* endV() {return ve;}
    bool vertexInE (Vertex* v) {return (v == vs) || (v == ve);}
    void printedge();
};
```



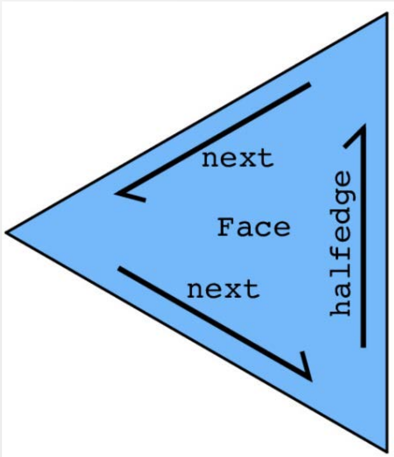

```
class Face {
    Edge* edges[3];
public:
    Face () {};
    void setEdge(int i, Edge* edge) {edges[i] = edge;}
    Edge* findPreE (Edge *e);
};
```

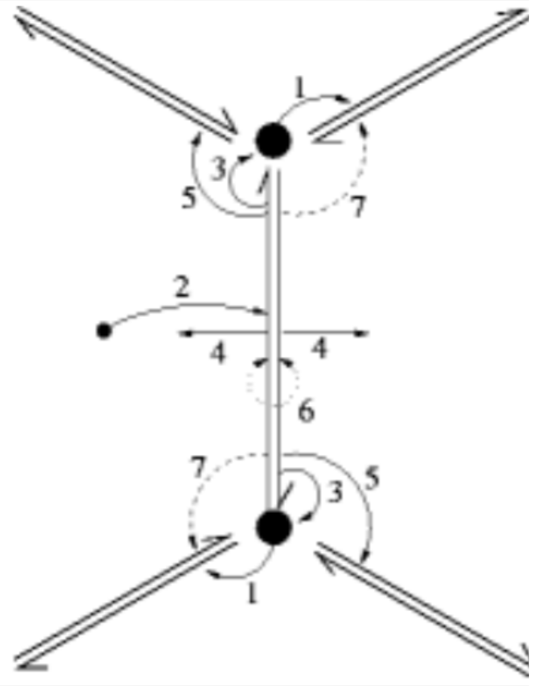
face-based, half-edge based, edge-based structure

There are many popular data structures used to represent polygonal meshes.
While **face-based structures** store their connectivity in faces referencing their vertices and neighbors,
edge-based structures put the connectivity information into the edges. Each edge references its two vertices, the faces it belongs to and the two next edges in these faces.
If one now splits the edges (i.e. an edge connecting vertex A and vertex B becomes two directed **halfedges** from A to B and vice versa) one gets a **halfedge-based** data structure. The following figure illustrates the way connectivity is stored in this structure:

Half-Edge Data Structure

used to represent polygonal meshes connectivity information in computer graphics.

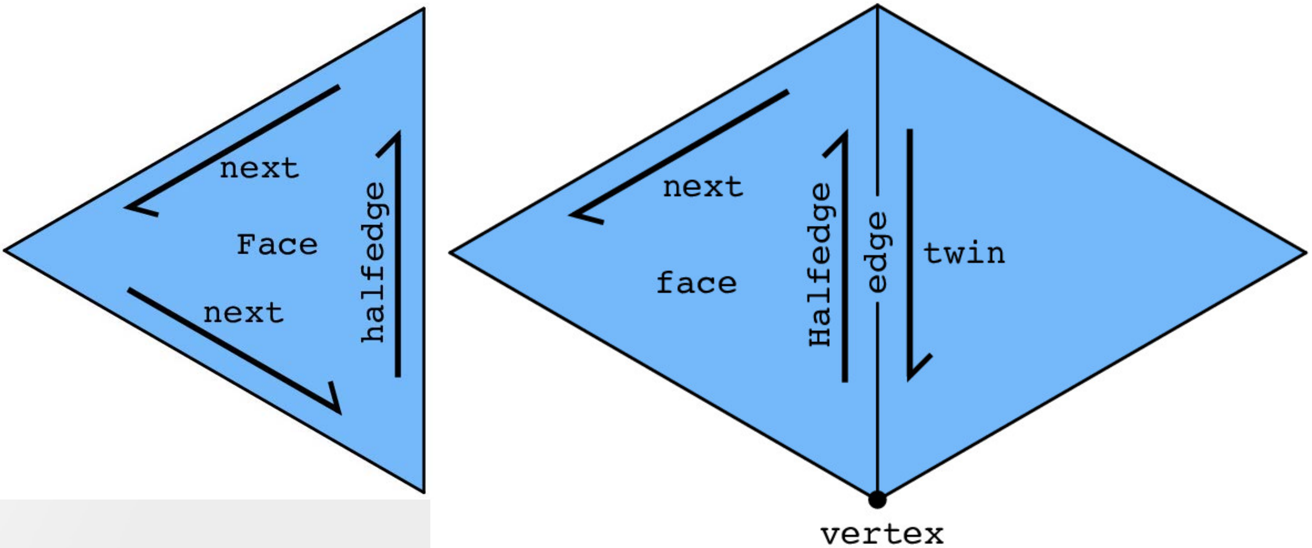




- Each **vertex** references one outgoing halfedge, i.e. a halfedge that starts at this vertex (1).
- Each **face** references one of the halfedges bounding it (2).
- Each **halfedge** provides a handle to
  - the vertex it points to (3),
  - the face it belongs to (4)
  - the next halfedge inside the face (ordered counter-clockwise) (5),
  - the opposite halfedge (6),
  - (optionally: the previous halfedge in the face (7)).

Half-Edge Data Structure

Used in Computer Graphics programs.



Key idea: two half-edges act as "glue" between mesh elements

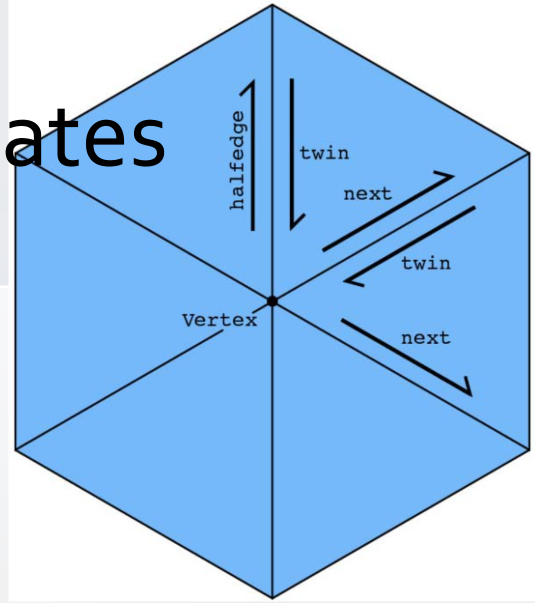
Each vertex, edge and face points to one of its half edges

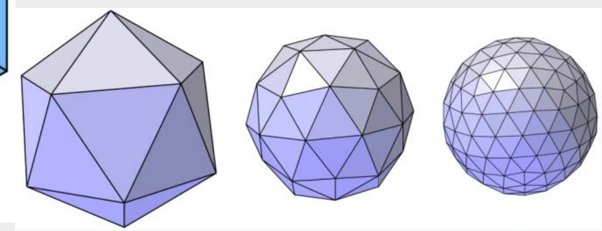
Use twin and next pointers to move around mesh

Process vertex, edge and/or face pointers

```
struct Halfedge {
    Halfedge *twin,
    Halfedge *next;
    Vertex *vertex;
    Edge *edge;
    Face *face;
}
struct Vertex {
    Point pt;
    Halfedge *halfedge;
}
struct Edge {
    Halfedge *halfedge;
}
struct Face {
    Halfedge *halfedge;
}
```

Half-Edge Facilitates Mesh Traversal





process all vertices of a face

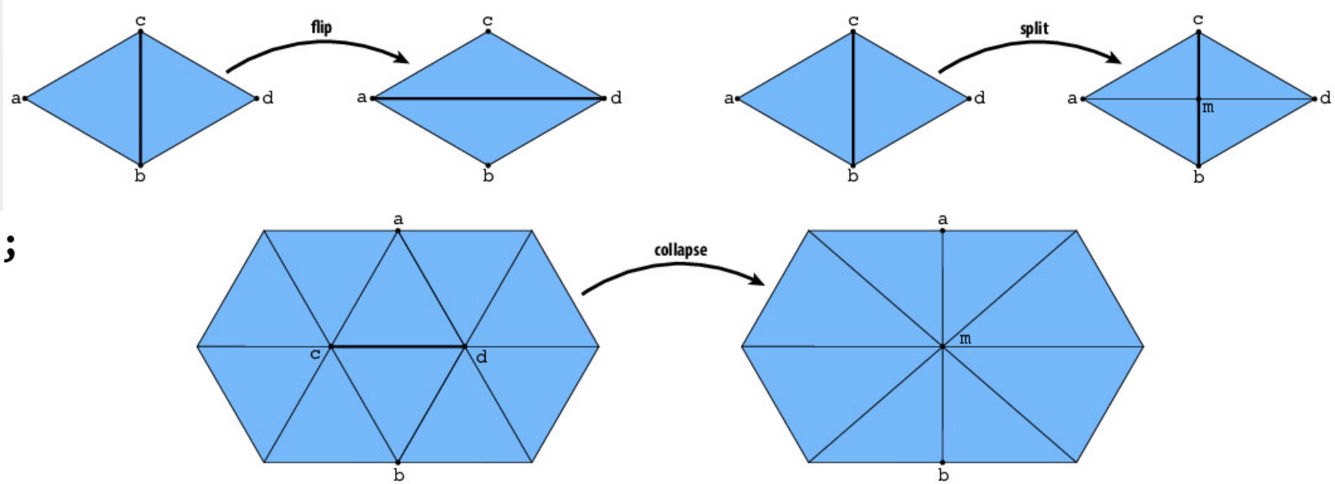
```
Halfedge* h = f->halfedge;
do {
   process(h->vertex);
   h = h->next;
}
while( h != f->halfedge );
```

Basic operations for linked list: insert, delete

Basic ops for half-edge mesh: flip, split, collapse edges

process all edges around a vertex

```
Halfedge* h = v->halfedge;
do {
   process(h->edge);
   h = h->twin->next;
}
while( h != v->halfedge );
```

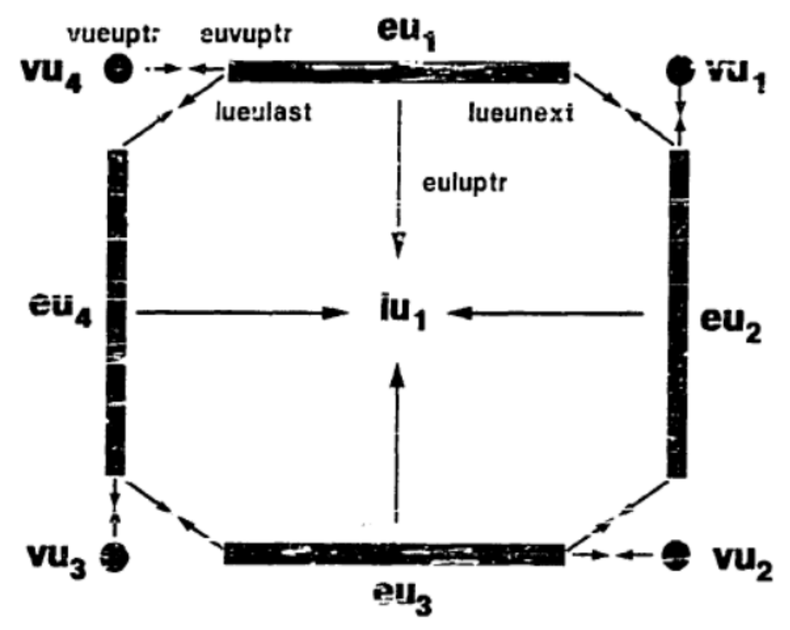


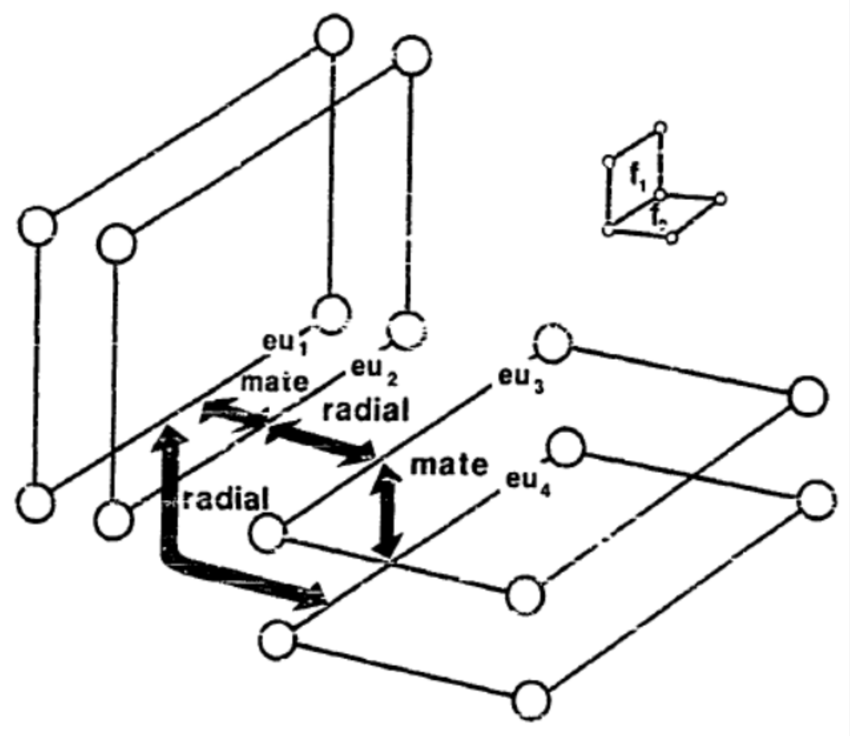
Allocate / delete elements; reassign pointers

(Care needed to preserve mesh manifold property)

Radial Edge non-manifold data structure

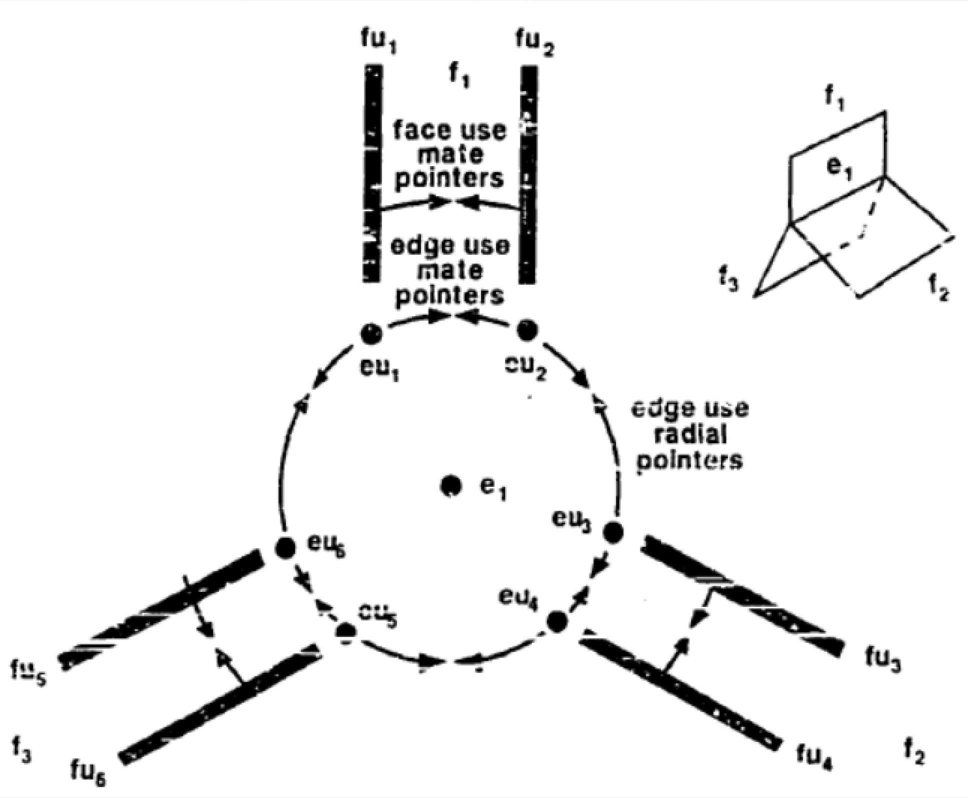
Radial Edge representation of two faces joining along a common edge showing how the four edge uses of the common edge (each side of each face uses the edge) are connected

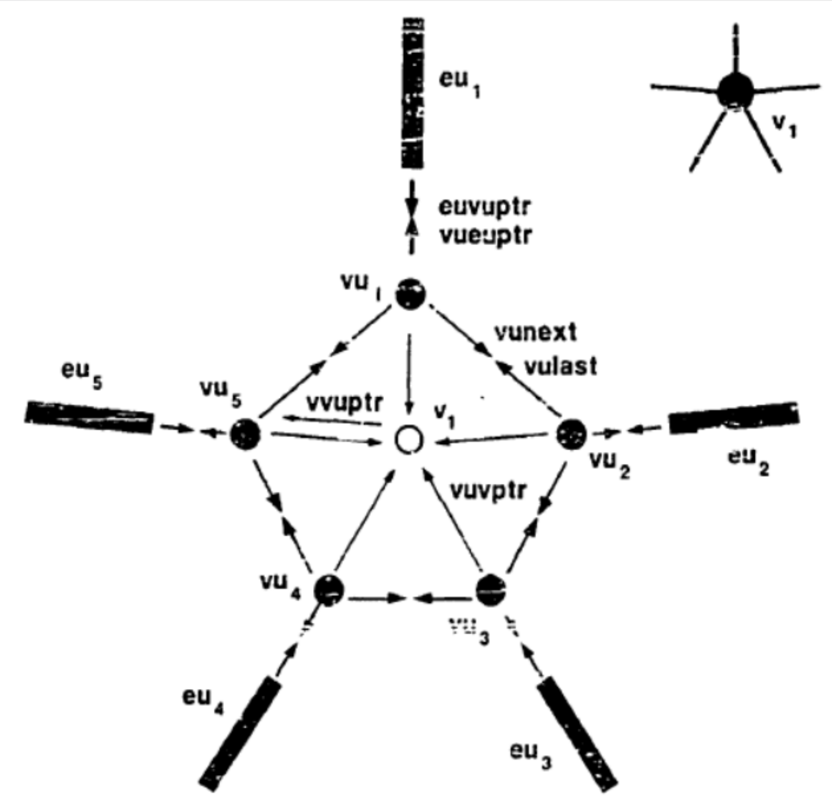




Radial Edge non-manifold data structure

Cross-sections of three and five faces sharing
a common edge in the Radial Edge representation.





CSG vs. B-Rep

B-Rep is appropriate to construct solid models of unusual shapes.

CSG	**B-Rep**
• Simple representation • Limited to simple objects • Stored as binary tree • Difficult to calculate • **Used** in CAD systems **as hybrid modeler**	• Flexible and powerful representation • Stored explicitly • Can be generated from CSG representation • **Used** in CAD systems **as hybrid modeler**

CSG vs. B-Rep

- Solid modeling systems

Comparison between CSG and B-rep representations.

	Storage of Model	Detail Level
CSG	Implicit	Low
B-rep	Explicit	High

Advantages (A) and Disadvantages (D) comparisons.

	Complexity	Uniqueness	History of Construction	Use in Interactive Environment	Local Operations
CSG	A	D	A	D	D
B-rep	D	A	D	A	A

CSG vs. B-Rep

1. B-rep uses Euler operators in modeling.
2. CSG needs low storage due to the simple tree structure and primitives.
3. CSG primitives are constructed from the half-space concept.
4. Directed surfaces, Euler operations and Euler's law fundamentally distinguish the B-rep from wireframe modeling.
5. Traditionally, CSG cannot model sculptured objects and thus is limited in modeling capability. (This is no longer true for Adv. CAD systems, such as Pro/E)

CSG vs. B-Rep

6. It is easier to convert a CSG model to a wireframe model than to convert a B-rep model to a wireframe model.
7. Because both CSG and B-rep use face direction (half-space or surface normal), they can have a full "body knowledge."
8. Generally speaking, most high-end CAD tools have the B-rep (or hybrid) method.
9. B-Rep requires more storage.
10. B-Rep manipulation is slow with respect to CSG.

New Challenges to Geometric Modeling

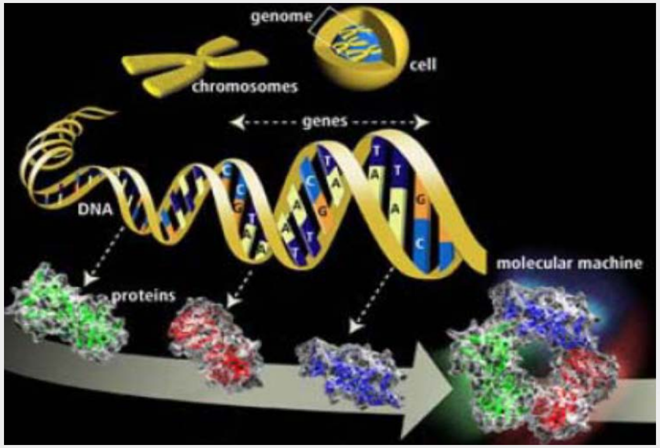
Modeling Porous Medium
Modeling Non-homogeneous Materials

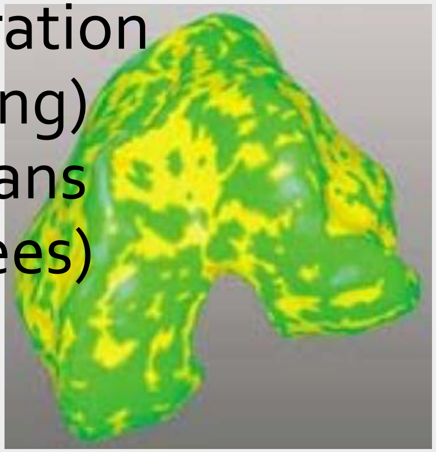
- varying density
- changing composition
- multiple phases (solid, liquid)

…

Biomedical Applications (geometry, materials, motion and mechanics)

- Medical Images (surgical operation
- simulator, training and planning)
- Computer models from CT scans
- (quantify motion in actual knees)





Solid Modeling Ref. Mantyla

Introduction

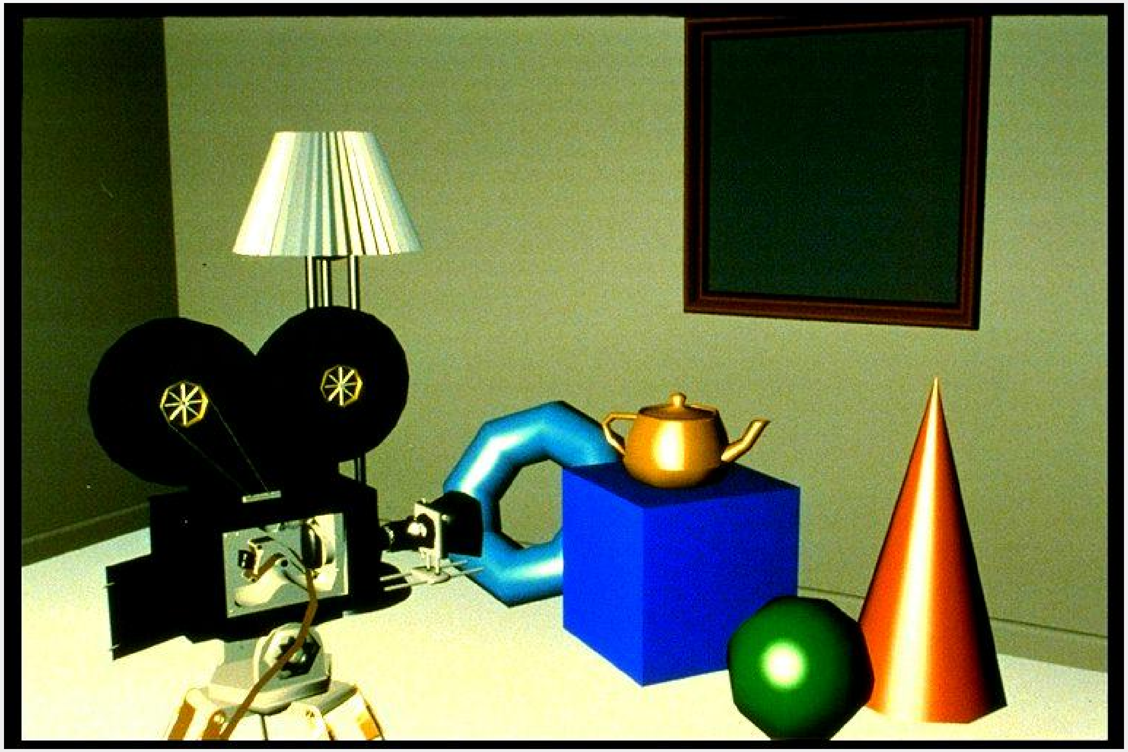
Aim of modeling:

- The search of a media of communication

Introduction (cont)

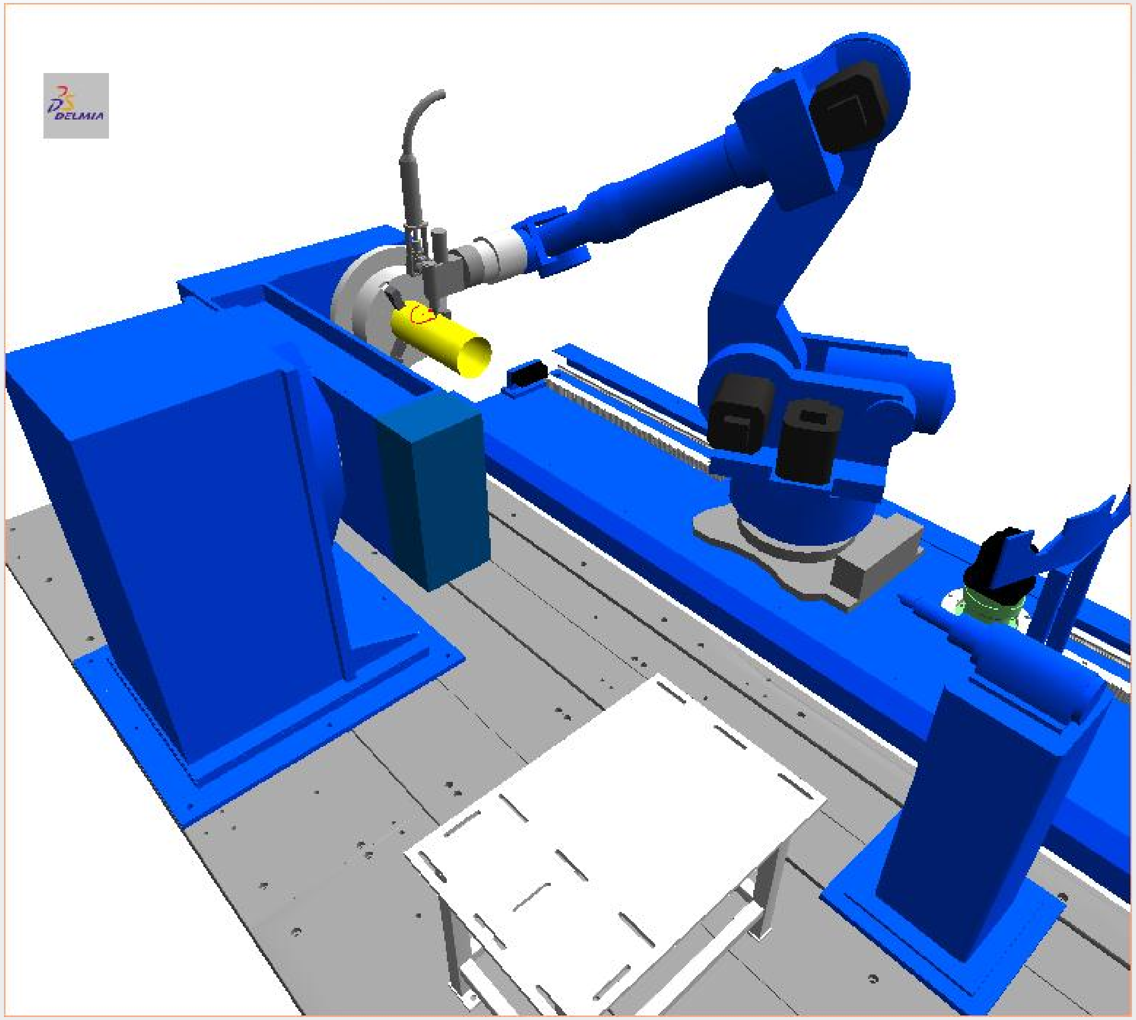
Geometric modeling

- Which parts of the objects are visible to the viewer? Colors?

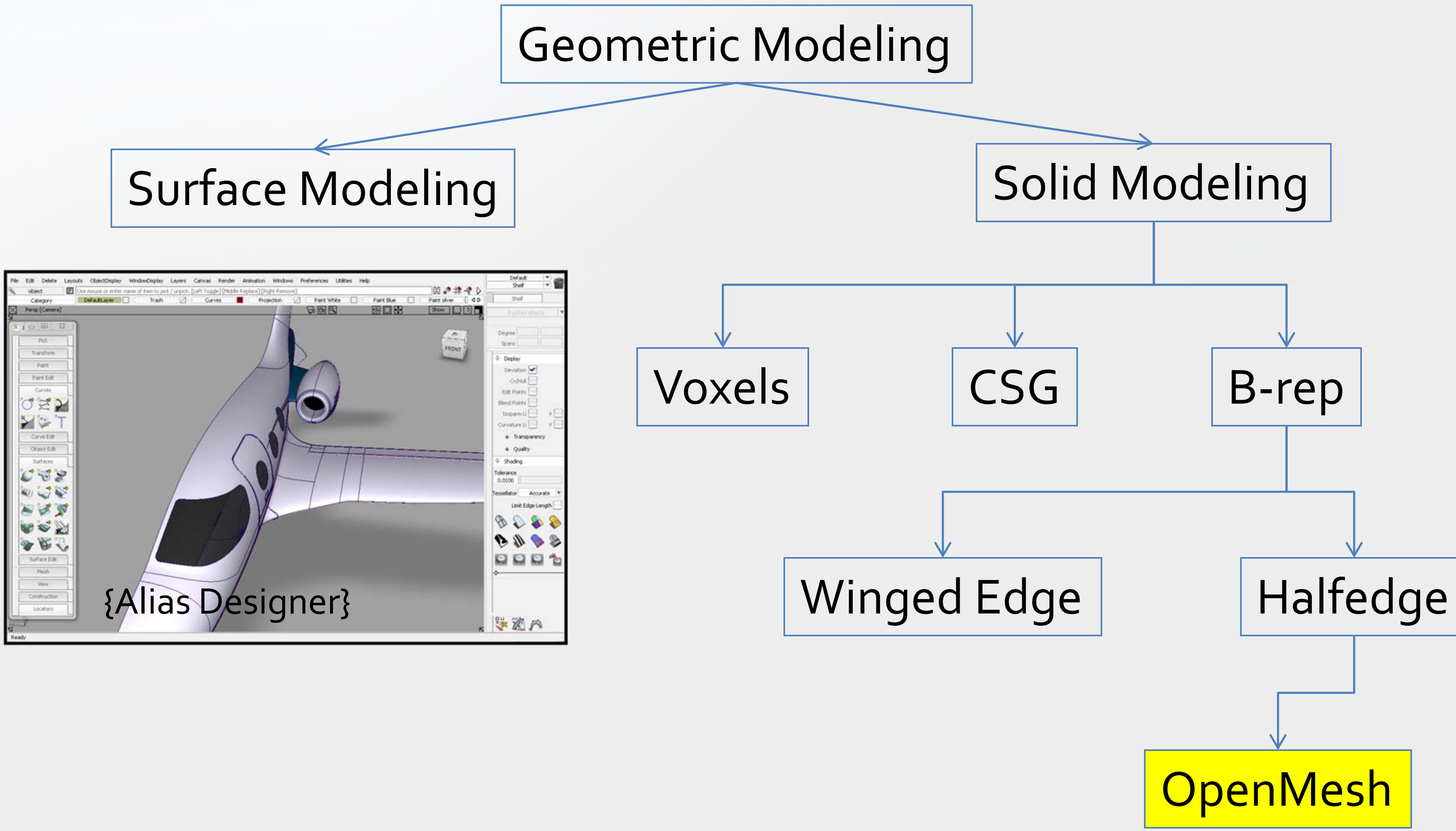


Introduction

- Solid modeling

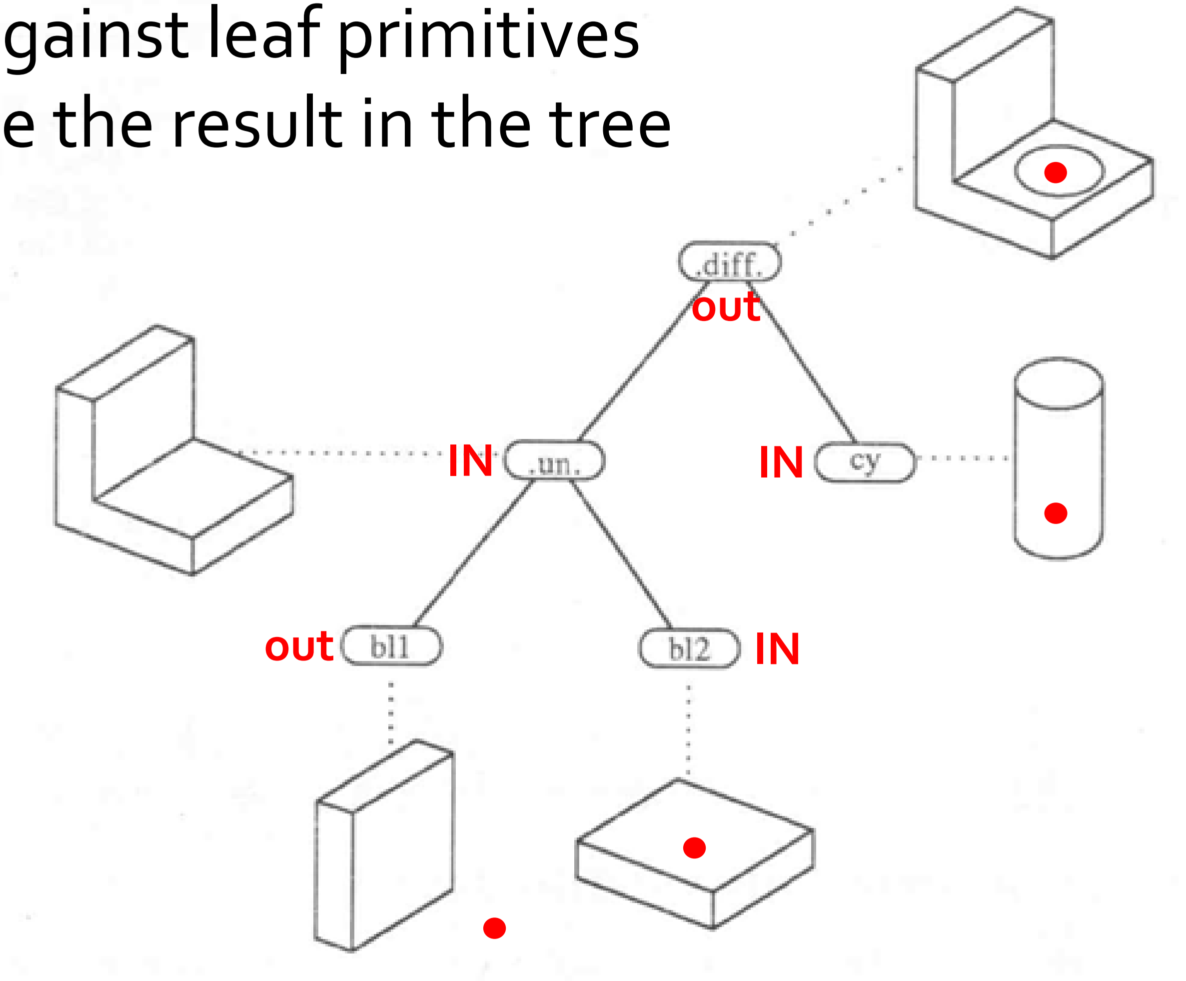


Taxonomy

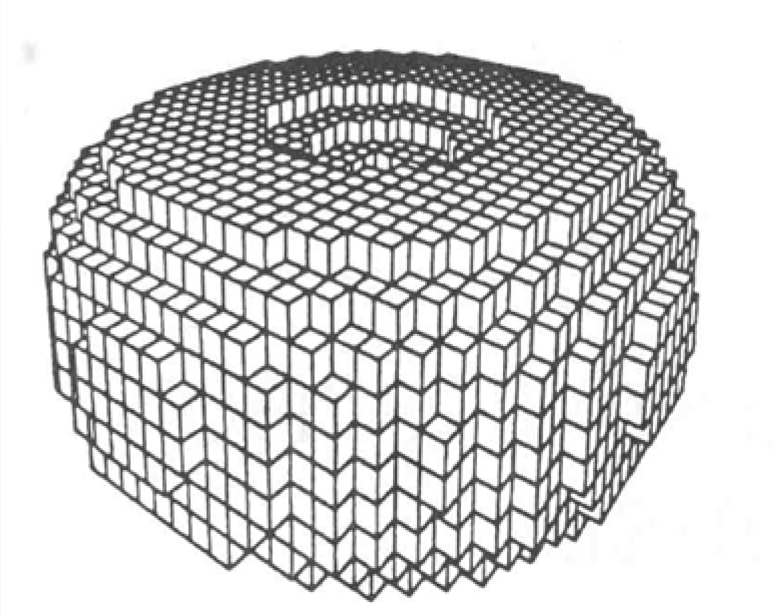

Geometric Modeling
Surface Modeling
Solid Modeling
Voxels
CSG
B-rep
Winged Edge
Halfedge
OpenMesh
{Alias Designer}

Point Inclusion Test for CSG

1. Classify against leaf primitives
2. Propagate the result in the tree



Volumetric Representation



$$v_{ijk} = \begin{cases} 1 & \text{solid} \\ 0 & \text{otherwise} \end{cases}$$

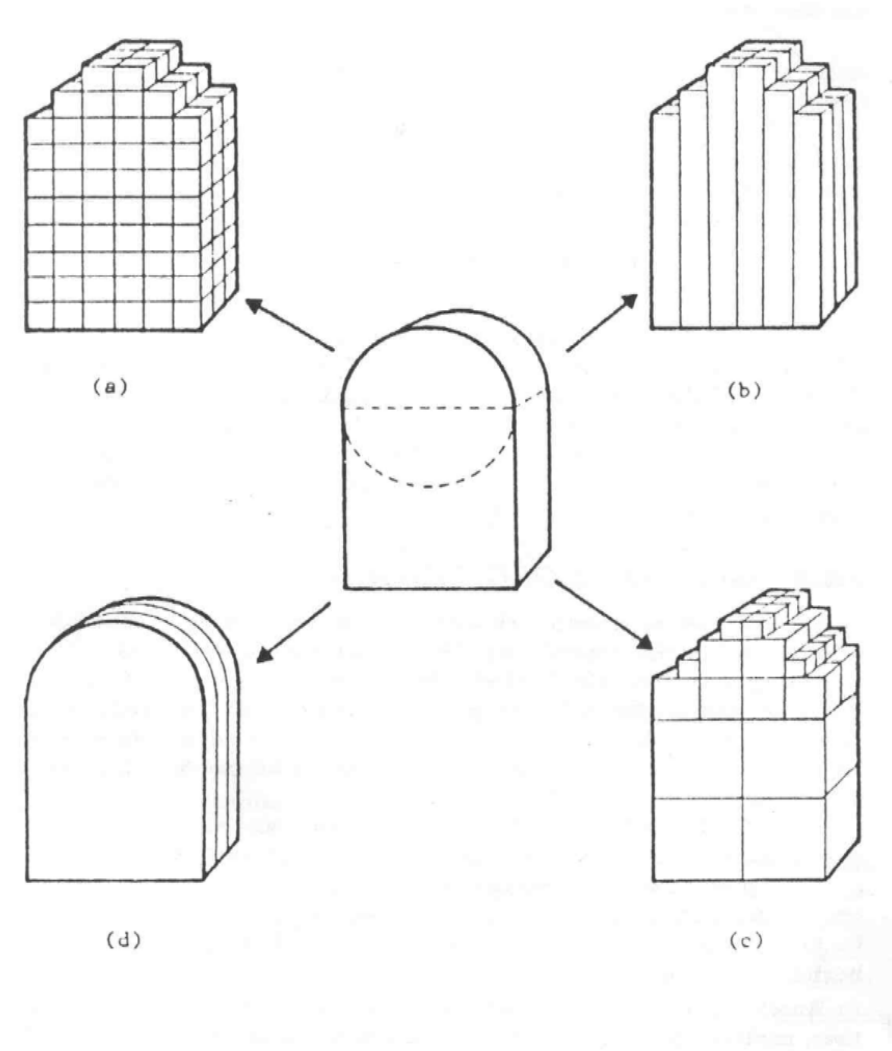
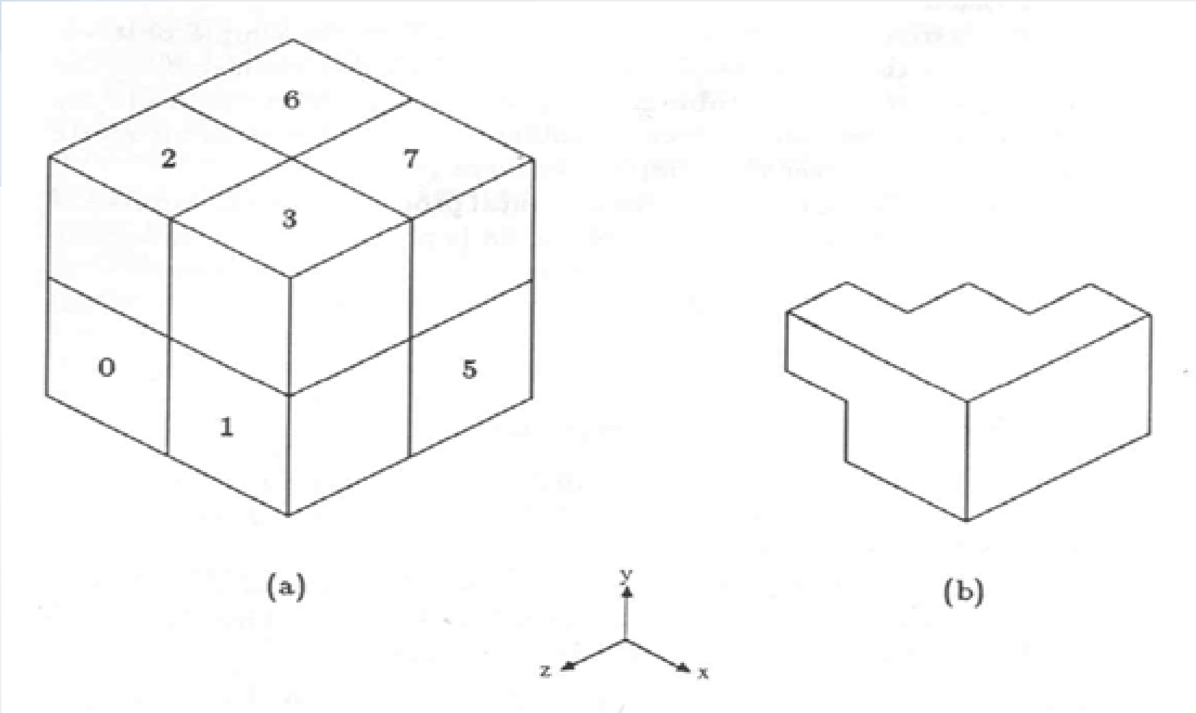
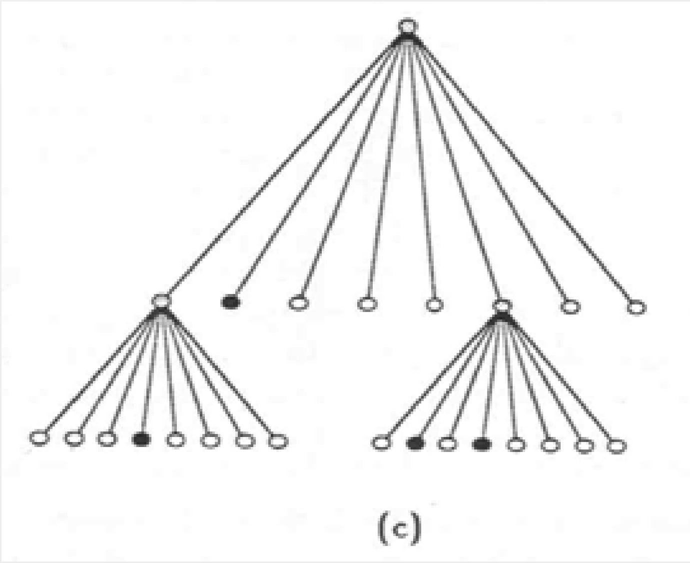


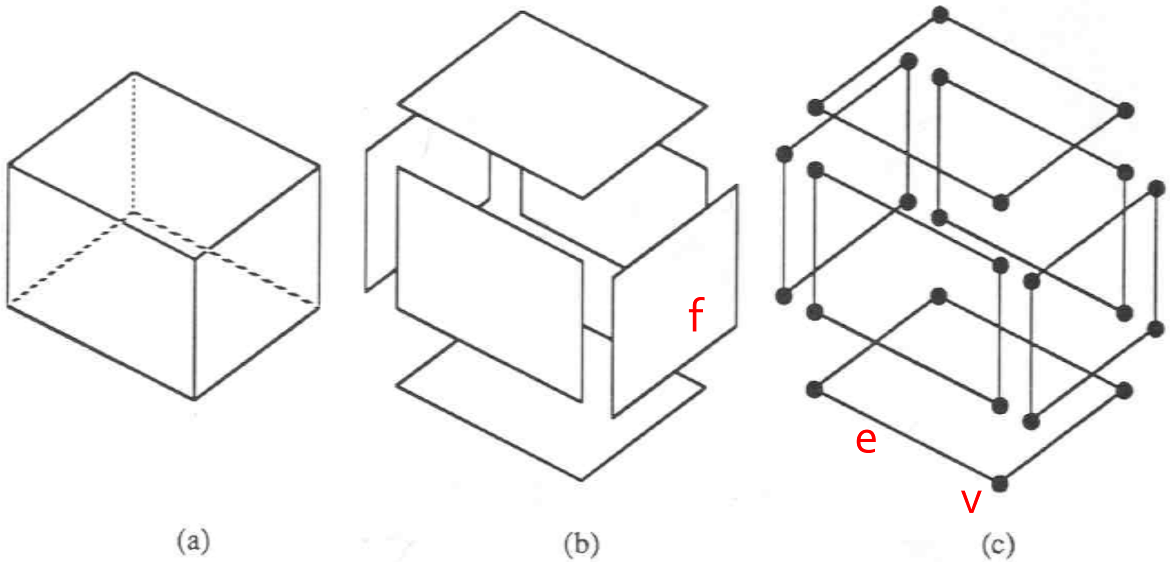
Figure 5.9 CSG-to-cell conversions.

Octree



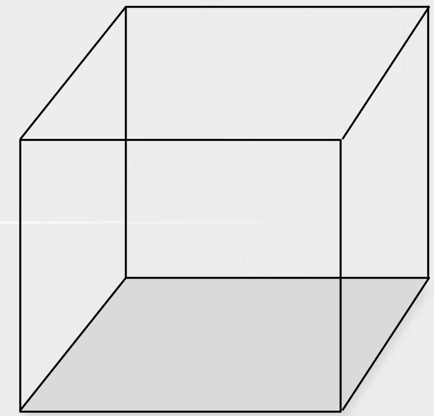


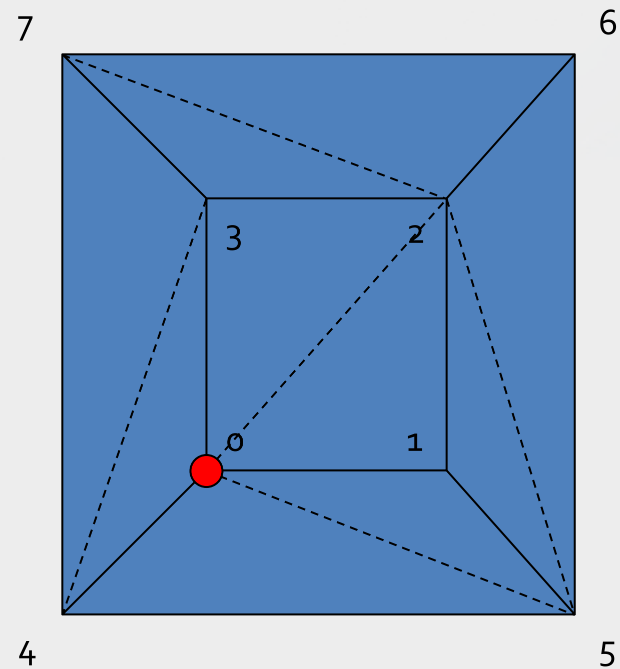
Boundary Model



Face, Edge, Vertex

Figure 6.2 Basic constituents of boundary models.

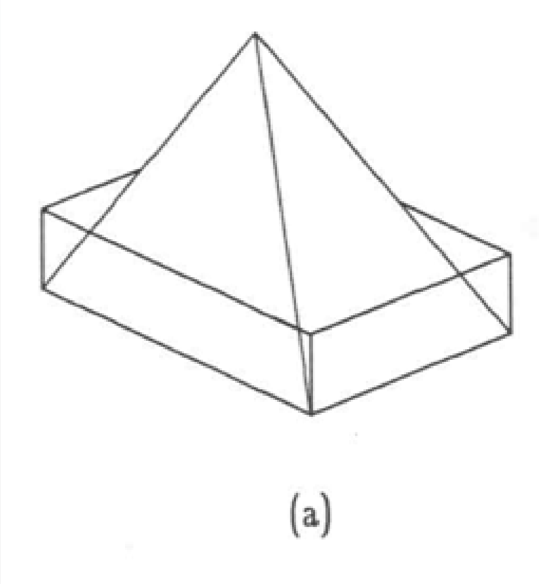


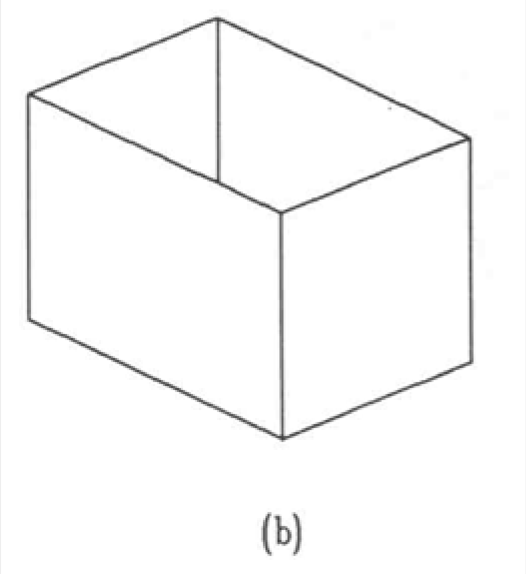


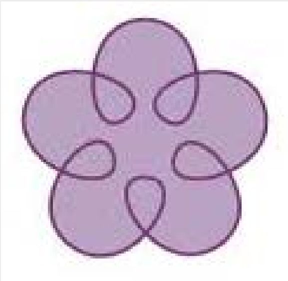
Validity of Boundary Model

Self-intersecting

non-manifold (next page)



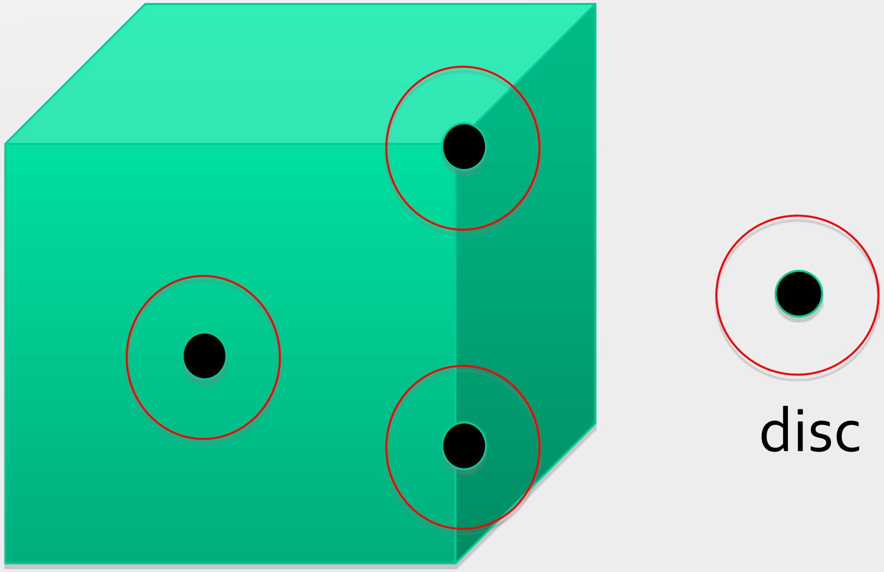




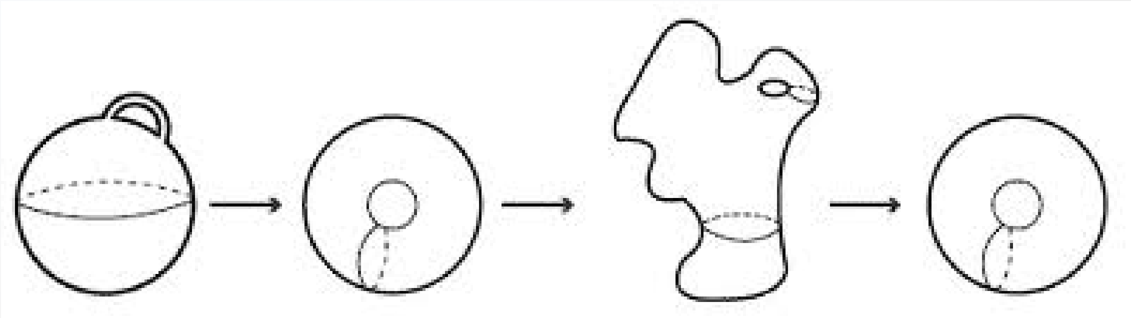
Elements of the model
- should not **self-intersect**
- should not intersect each other unless at their boundary.

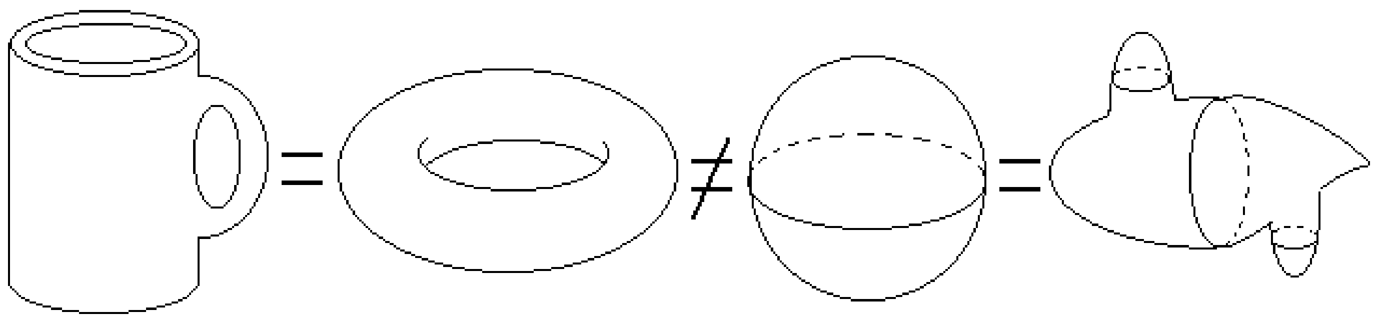
Definition of Manifold

For every point on the boundary,
its **neighborhood on the boundary** is homeomorphic (topologically equivalent) to an open disc.

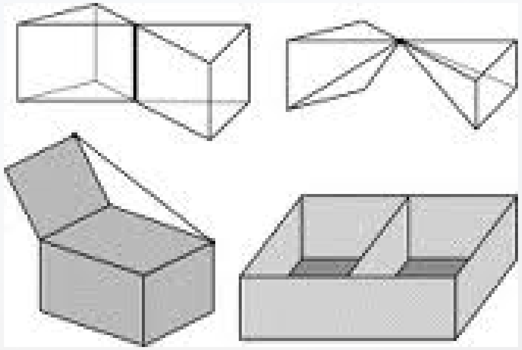


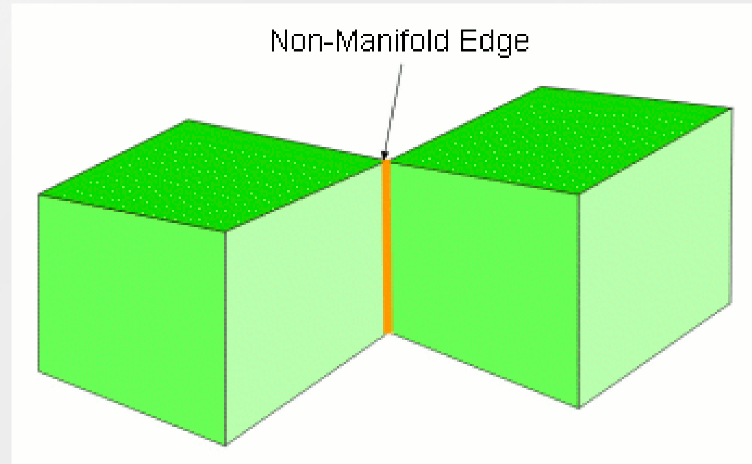
Topologically Equivalent





Examples of Non-Manifold Models





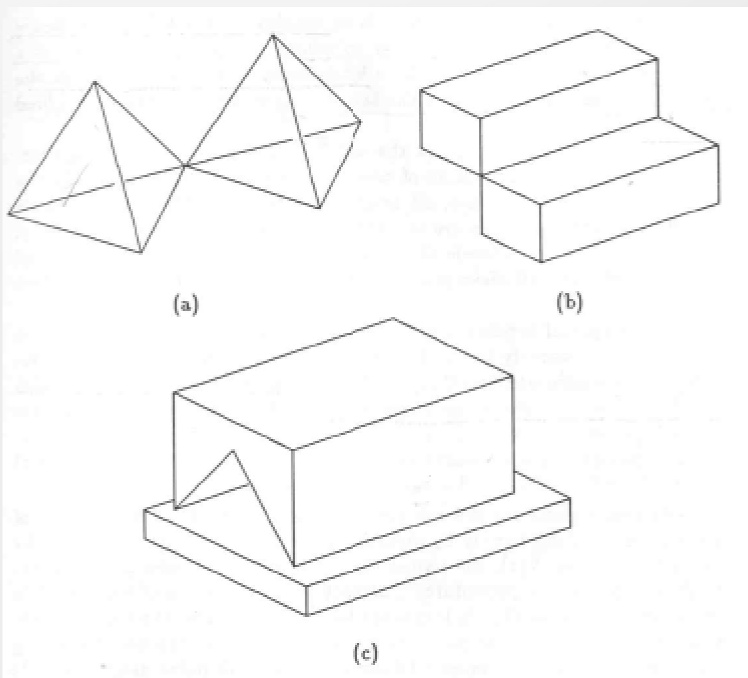
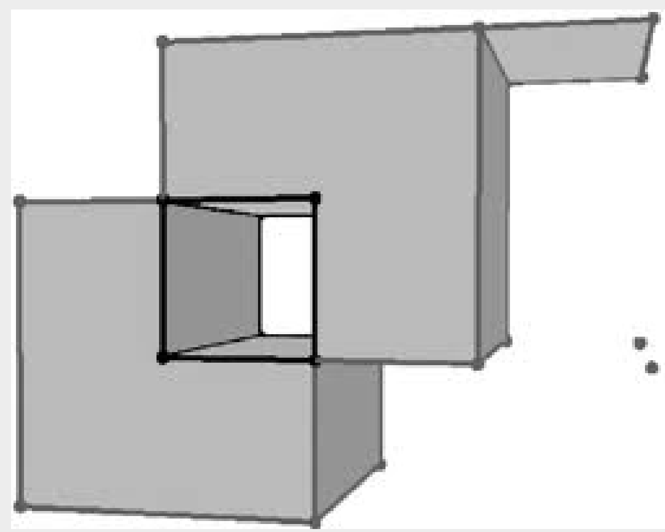
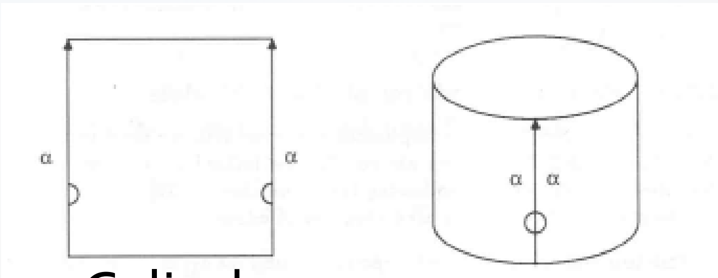


Figure 3.4 Solids with nonmanifold surfaces.

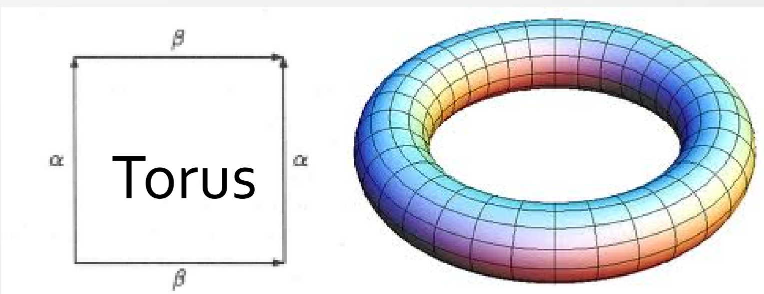


Plane Models

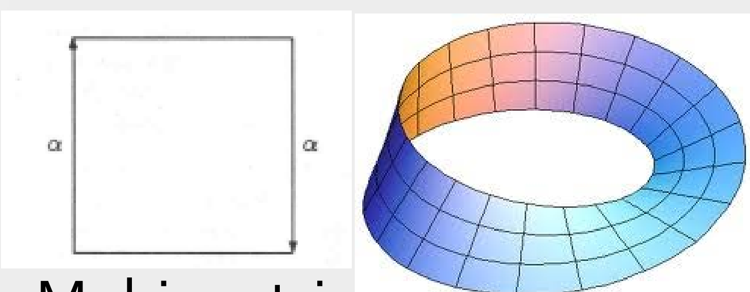
Edge identification



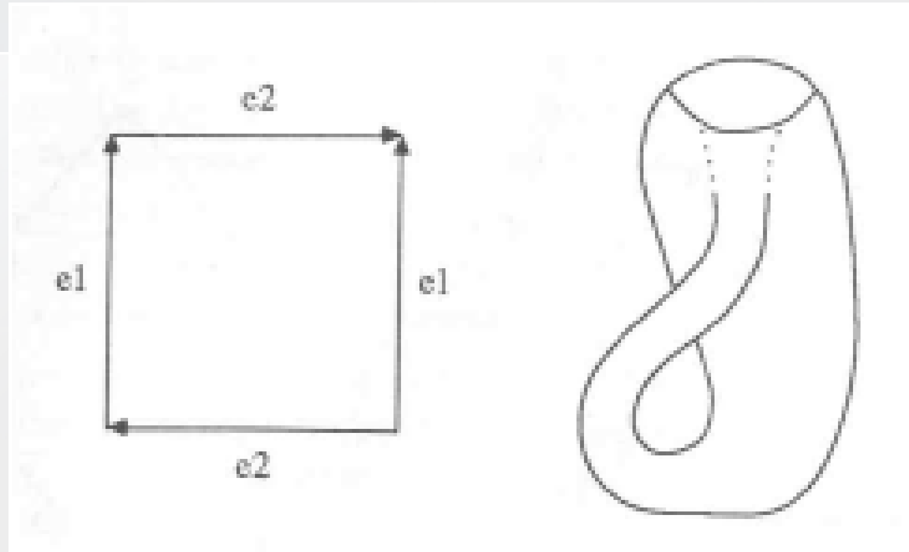
Cylinder



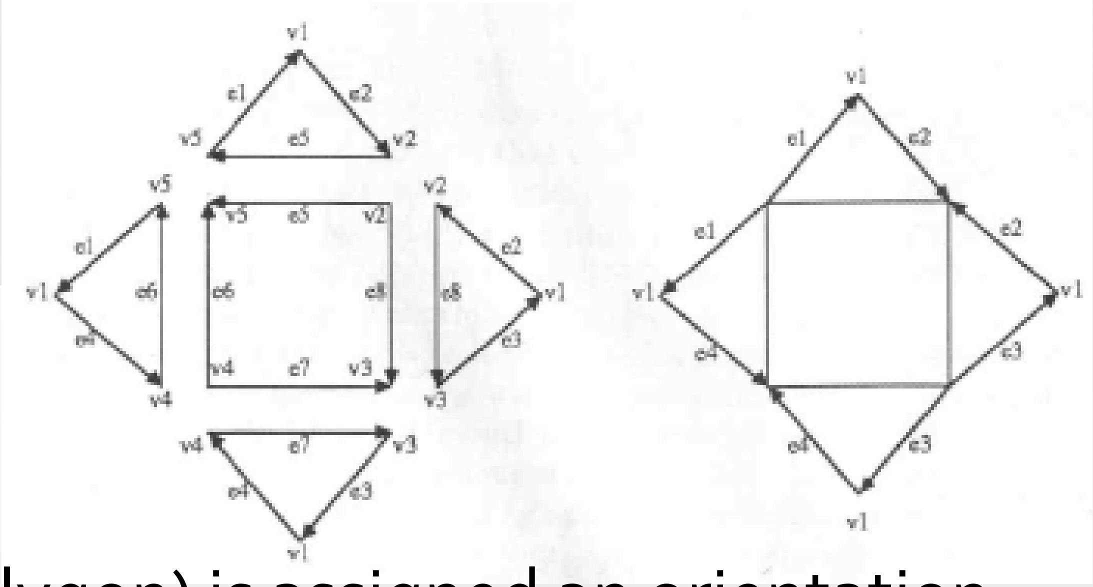
Torus



Mobius strip



Plane Model



Each edge (of a polygon) is assigned an orientation from one endpoint to the other

Every edge is identified with exactly to one other edge

For each collection of identified vertices, the polygons identified at that collection can be arranged in a cycle such that each consecutive pair of polygons in a cycle is identified at an edge adjacent to a vertex from the collection.

Orientable Solids

A plane model is orientable if the directions of its polygons can be chosen so that for each pair of identified edges, one edge occurs in its positive orientation, and the other one in its negative orientation

Euler-Poincaré Formula (ref)

$$V - E + F - (L - F) = 2(S - G)$$

V: the number of vertices
E: the number of edges
F: the number of faces
G: the number of holes that penetrate the solid, usually referred to as *genus* in topology
S: the number of *shells*. A shell is an internal void of a solid. A shell is bounded by a 2-manifold surface. Note that the solid itself is counted as a shell. Therefore, the value for **S** is at least 1.
L: the number of loops, all outer and inner loops of faces are counted.

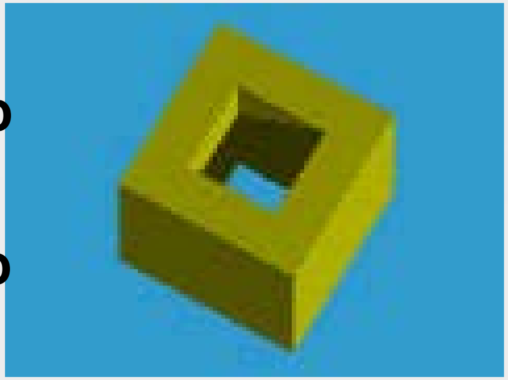
Examples

Box: V-E+F-(L-F)-2(S-G) = 8-12+6-(6-6)-2(1-0)=0

Open Box: V-E+F-(L-F)-2(S-G) = 8-12+5-(5-5)-2(0-0)=1

Box w/ *through* hole:

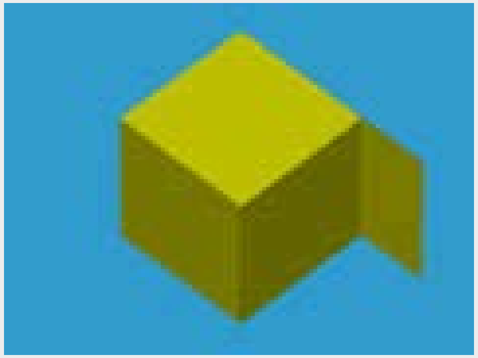
V-E+F-(L-F)-2(S-G) = 16-24+10-(12-10)-2(1-1)=0



Box w/ *blind* hole:

V-E+F-(L-F)-2(S-G) = 16-24+11-(12-11)-2(1-0)=0

V-E+F-(L-F)-2(S-G) = 10-15+7-(7-7)-2(1-0)=0



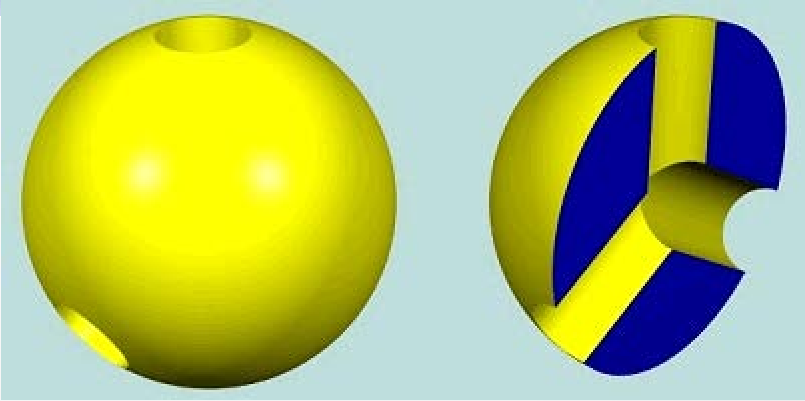
Invalid nonmanifold solid yet still yields ZERO!

The equation for Open objects is

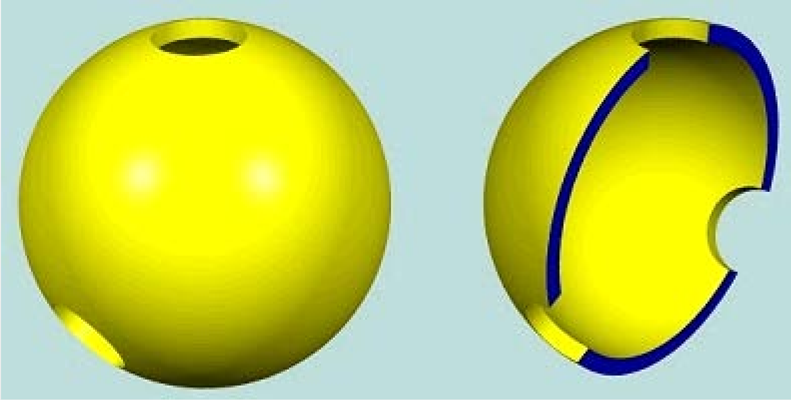
V-E+F-(L-F)-(S-G) = 10-15+7-0-(1-0)=1

$$F - E + V - L = B - G$$

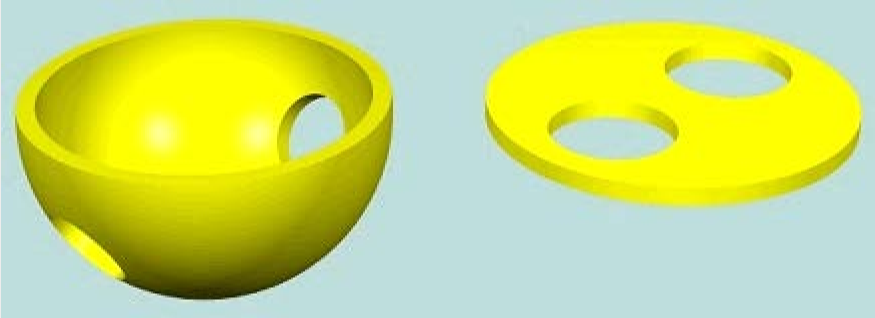
Count Genus Correctly



G = ?

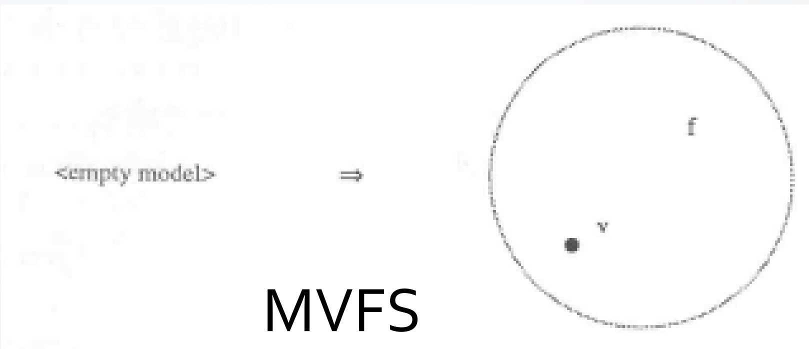


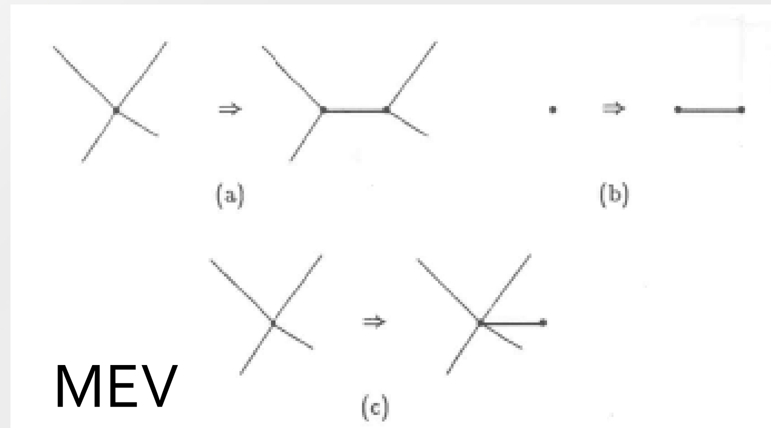
G = 3?

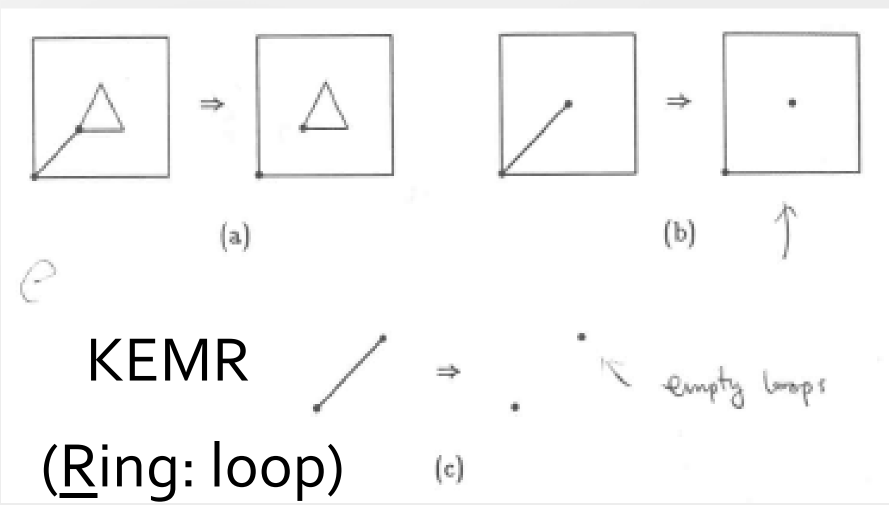


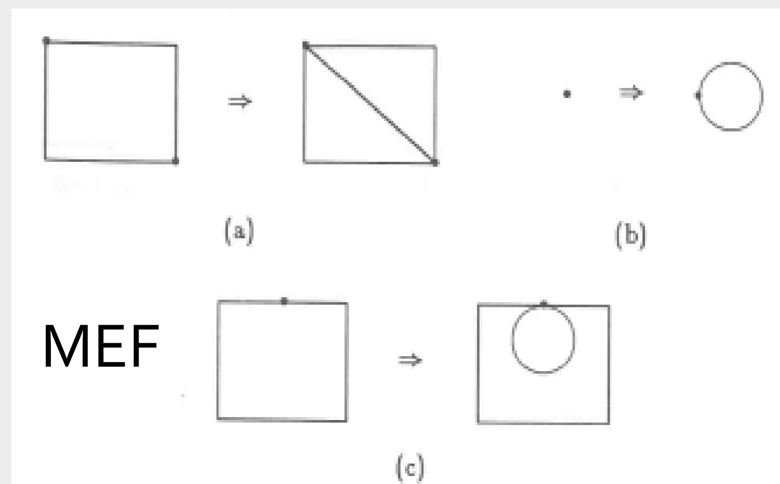
G = 2!

Euler Operators

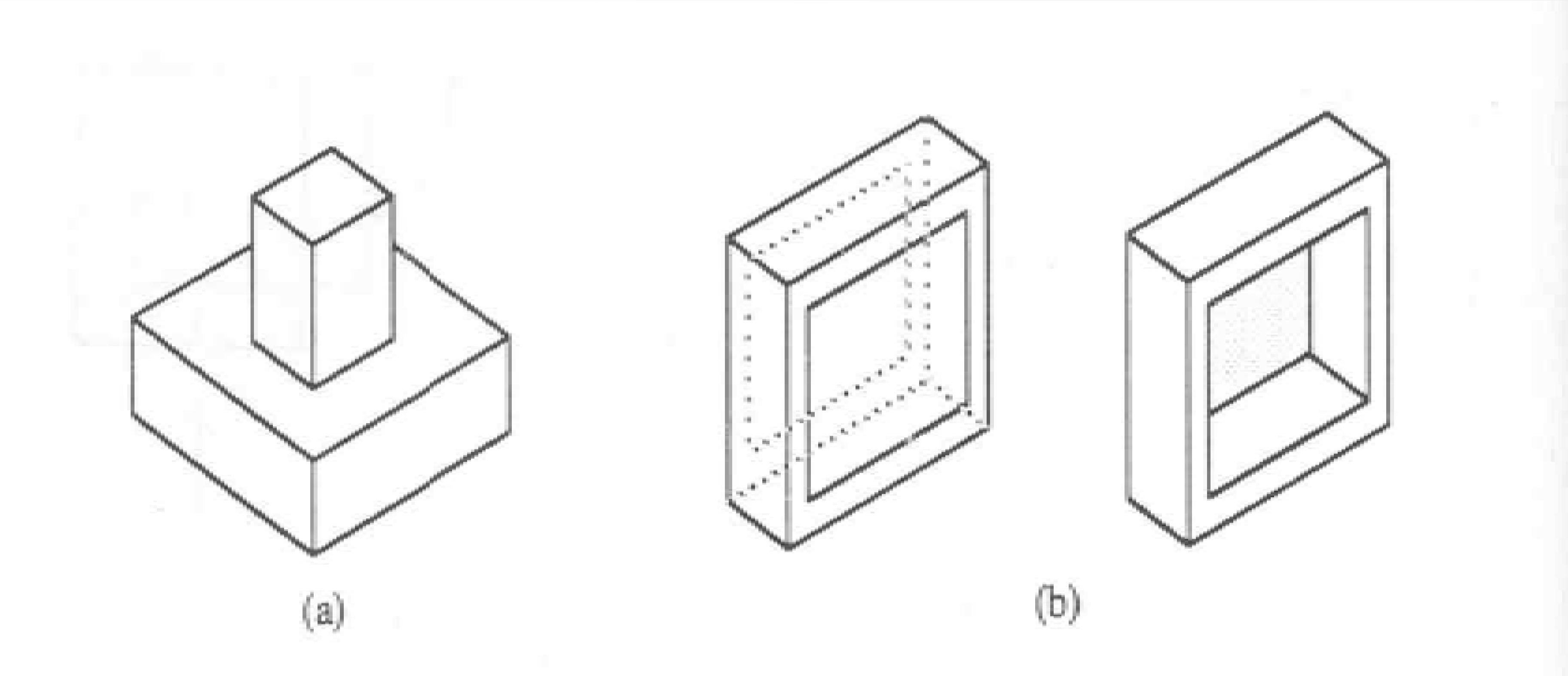








Global Operators



Example:
Euler Operators

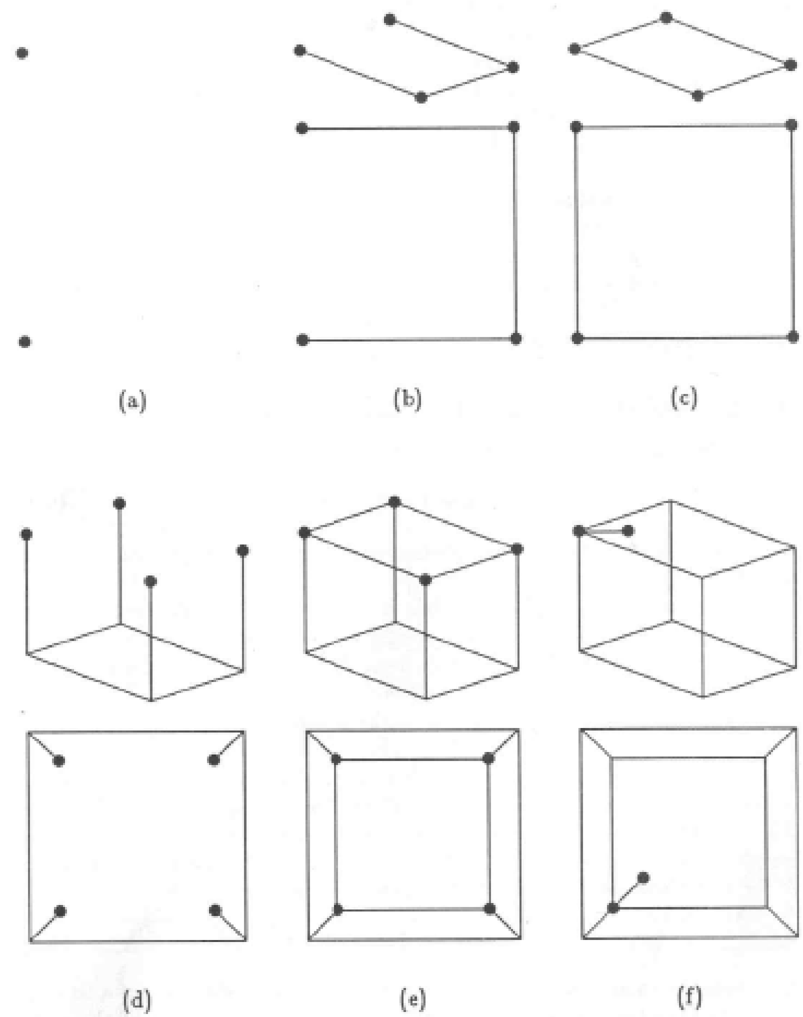


Figure 9.11 Example of Euler operators.

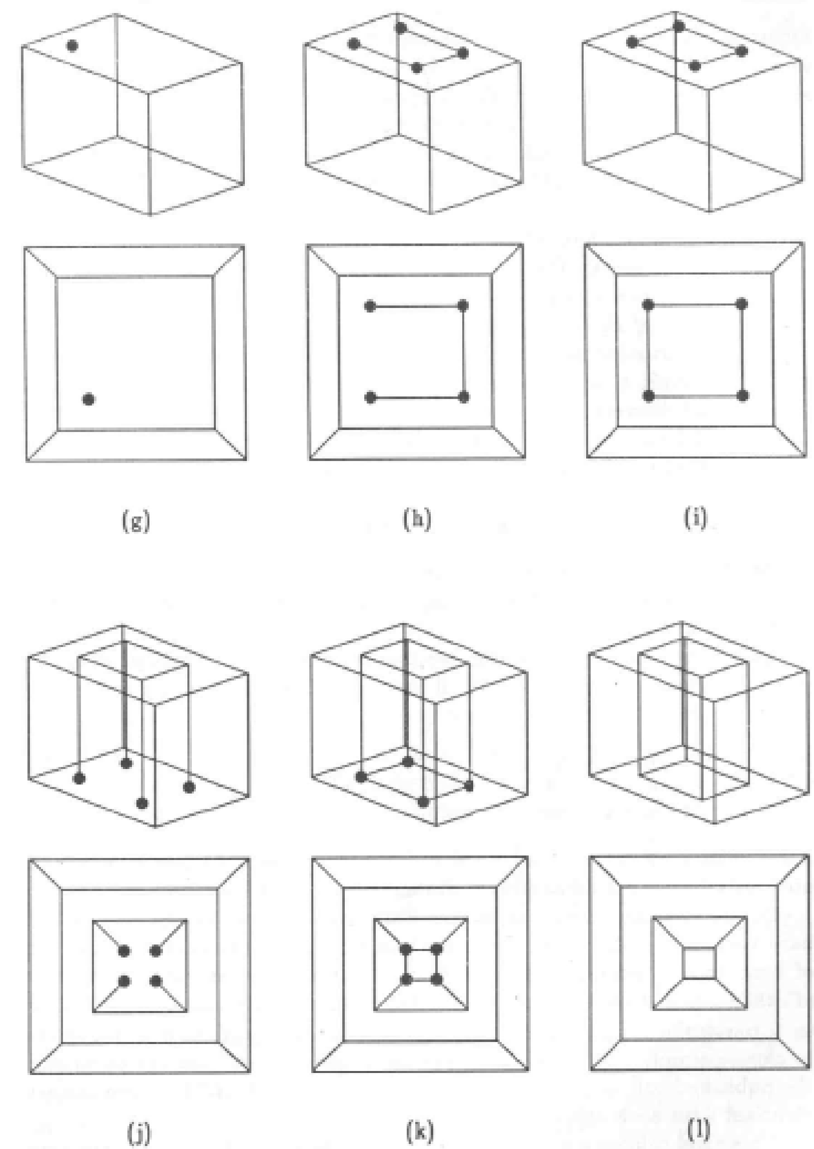
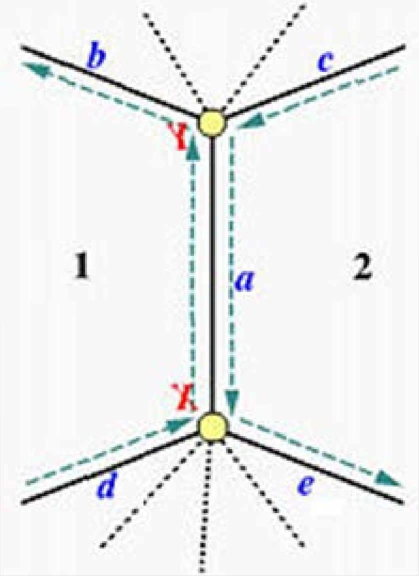


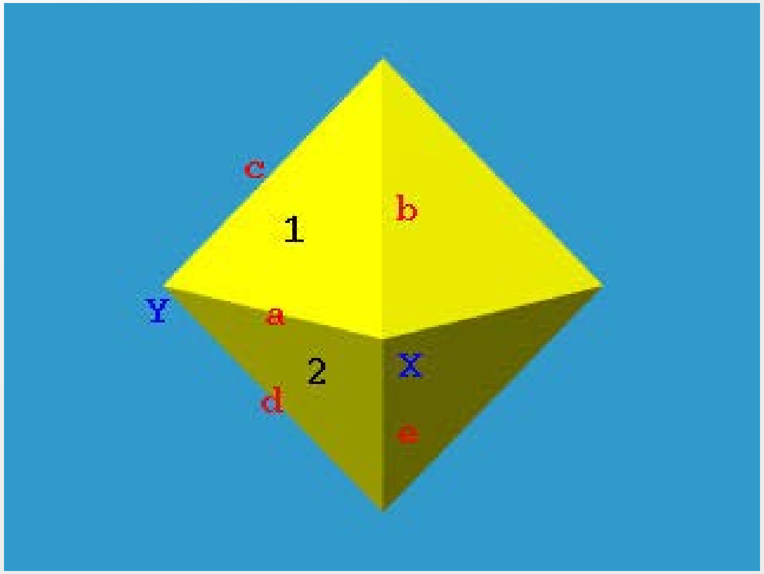
Figure 9.11 Example of Euler operators (cont.).

Winged-Edge Data Structure

- Commonly used to describe polygon models
- Quick traversal between faces, edges, vertices
- Linked structure of the network
- Assume there is no holes in each face

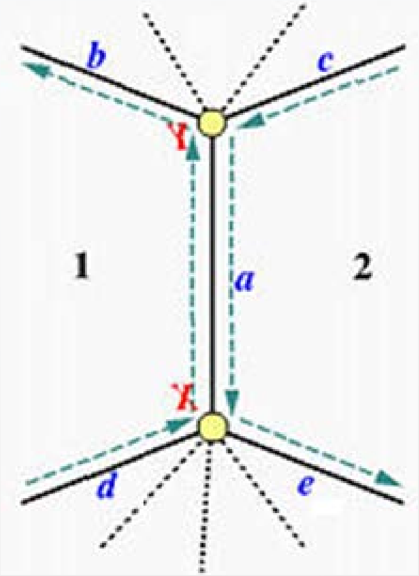
Winged-Edge Data Structure

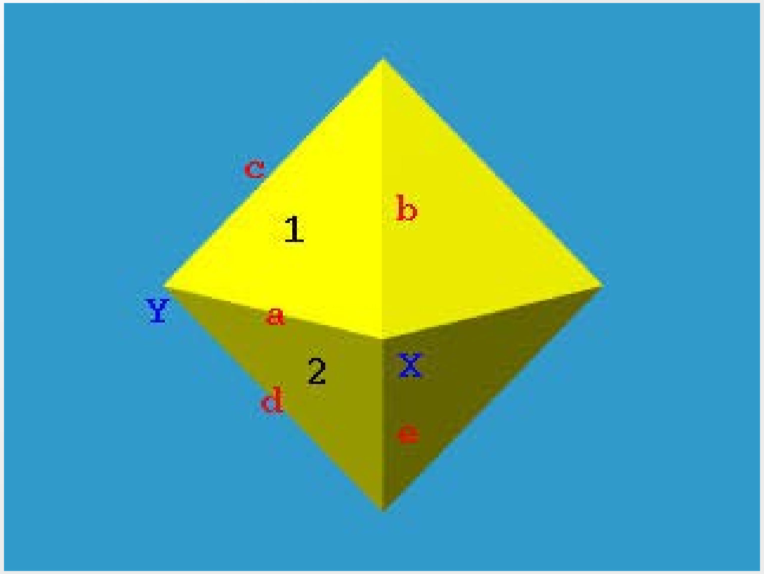




- vertices of this edge
- its *left* and *right* faces
- the predecessor and successor when traversing its left face
- the predecessor and successor when traversing its right face.

Winged-Edge Data Structure

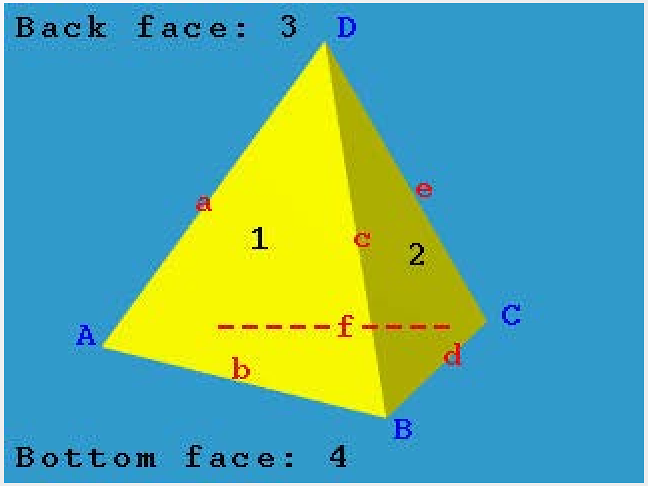




Edge	Vertices		Faces		Left Traverse		Right Traverse	
Name	Start	End	Left	Right	Pred	Succ	Pred	Succ
a	X	Y	1	2	d	b	c	e

Edge Table

Winged-Edge Data Structure



Edge	Vertices		Faces		Left Traverse		Right Traverse	
Name	Start	End	Left	Right	Pred	Succ	Pred	Succ
a	A	D	3	1	f	e	c	b
b	A	B	1	4	a	c	d	f
c	B	D	1	2	b	a	e	d
d	B	C	2	4	c	e	f	b
e	C	D	2	3	d	c	a	f
f	A	C	4	3	b	d	e	a

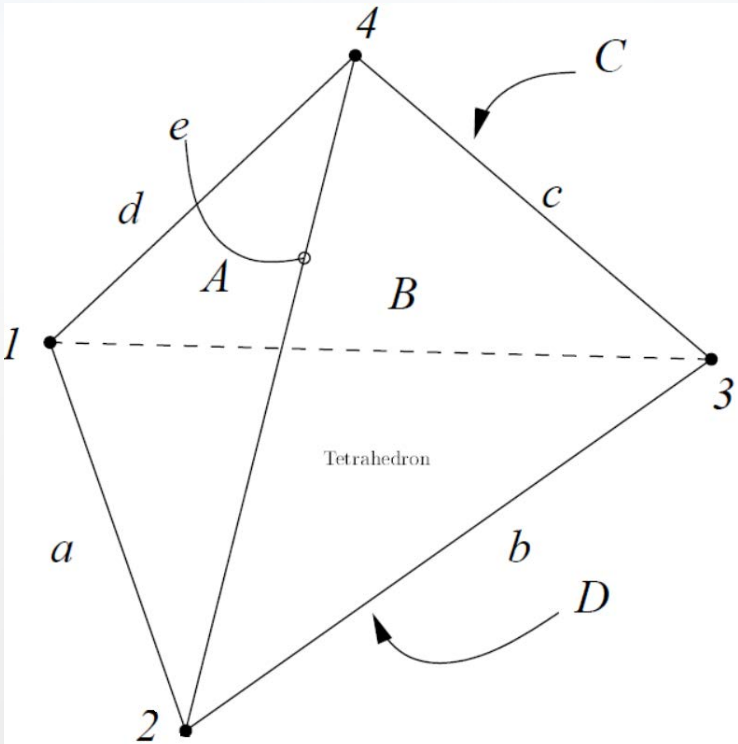
Winged-Edge Data Structure

- the *vertex table* and the *face table*

Vertex Name	Incident Edge
A	a
B	b
C	d
D	c

Face Name	Incident Edge
1	a
2	c
3	a
4	b

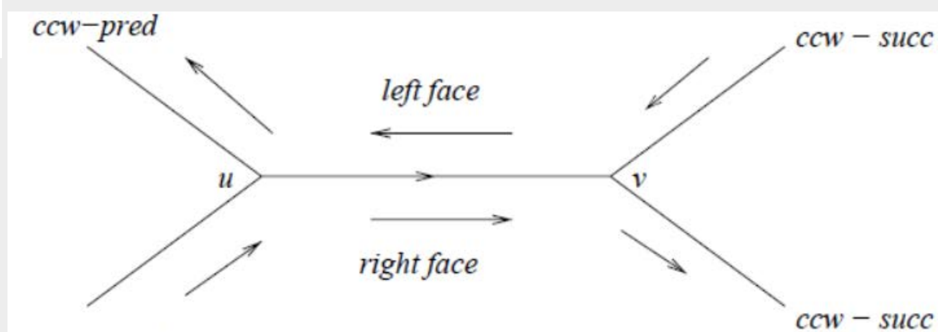
Winged-Edge Data Structure



Tetrahedron

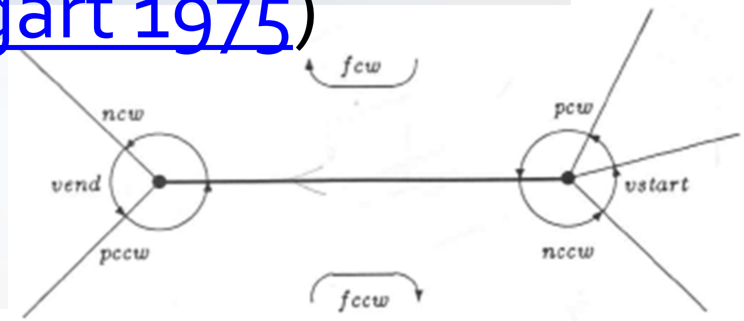
Edge	Vertices		Faces		Clockwise		Counter-clockwise	
Name	from	to	left	right	pred	succ	pred	succ
a	1	2	A	D	d	e	f	b
b	2	3	B	D	e	c	a	f
f	3	1	C	D	c	d	b	a
c	3	4	B	C	b	e	f	d
d	1	4	C	A	f	c	a	e
e	2	4	A	B	a	d	b	c

Edge Table of the Tetrahedron, Winged-Edge Methodology



Winged-Edge Data Structure

Winged Edge Data Structure
(Baumgart 1975)



(a)

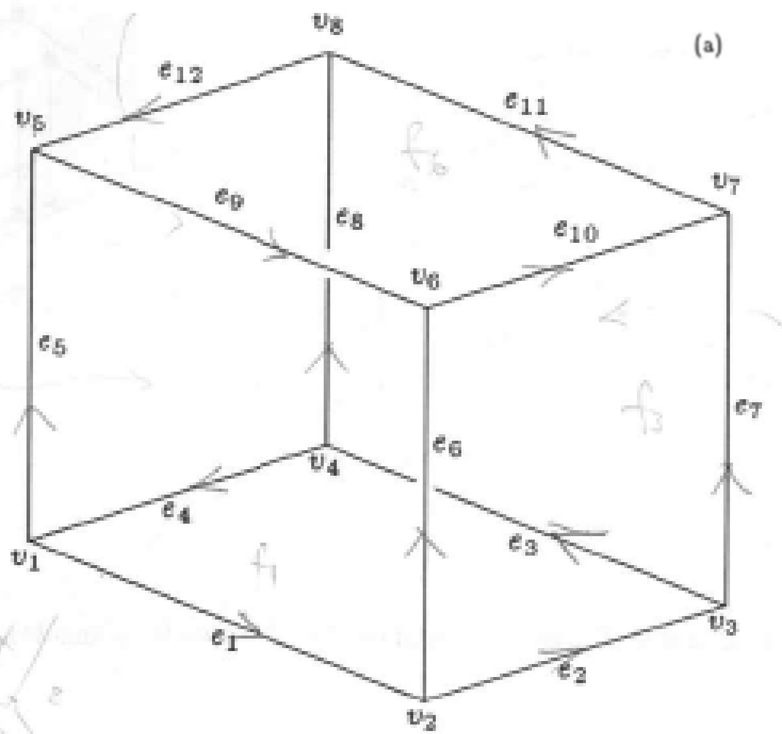


Figure 6.3 A sample object.

edge	*vstart*	*vend*	*ncw*	*nccw*
e_1	v_1	v_2	e_2	e_5
e_2	v_2	v_3	e_3	e_6
e_3	v_3	v_4	e_4	e_7
e_4	v_4	v_1	e_1	e_8
e_5	v_1	v_5	e_9	e_4
e_6	v_2	v_6	e_{10}	e_1
e_7	v_3	v_7	e_{11}	e_2
e_8	v_4	v_8	e_{12}	e_3
e_9	v_5	v_6	e_6	e_{12}
e_{10}	v_6	v_7	e_7	e_9
e_{11}	v_7	v_8	e_8	e_{10}
e_{12}	v_8	v_5	e_5	e_{11}

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	$x_6\ y_6\ z_6$
v_7	$x_7\ y_7\ z_7$
v_8	$x_8\ y_8\ z_8$

face	*first edge*	*sign*
f_1	e_1	+
f_2	e_9	+
f_3	e_6	+
f_4	e_7	+
f_5	e_{12}	+
f_6	e_9	−

Figure 6.6 The winged-edge data structure.

Winged Edge

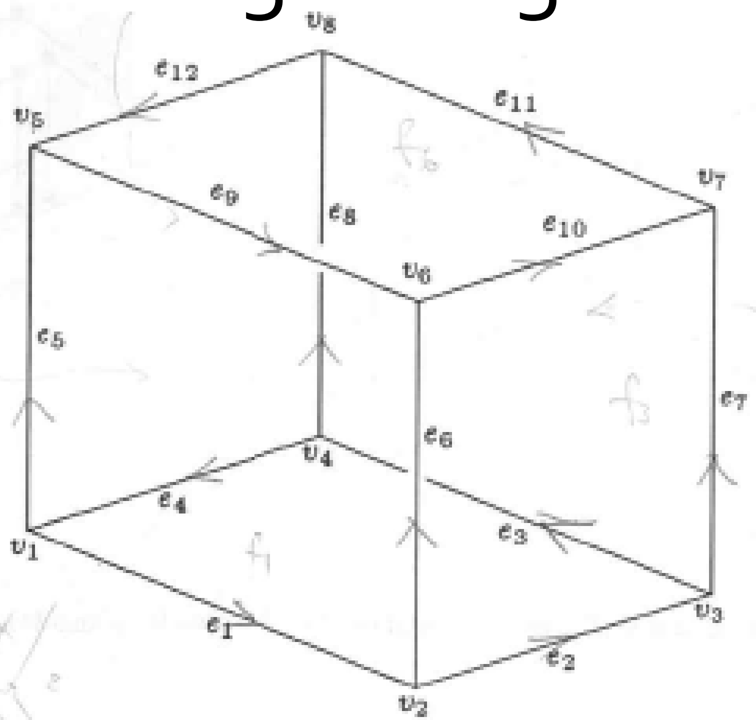
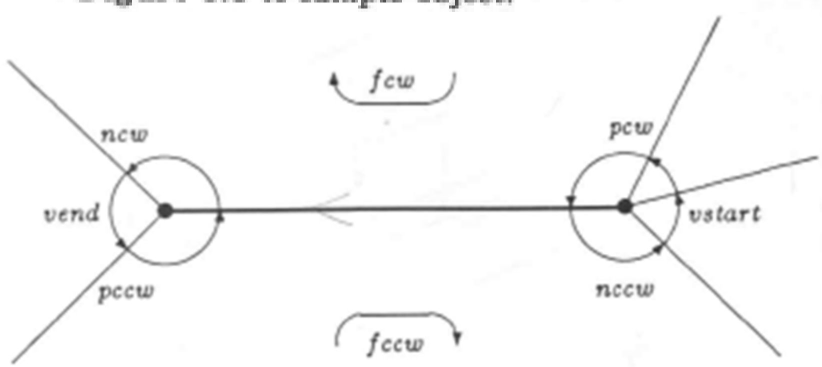


Figure 6.3 A sample object.



(a)

edge	vstart	vend	fcw	fccw	ncw	pcw	nccw	pccw
e_1	v_1	v_2	f_1	f_2	e_2	e_4	e_5	e_6
e_2	v_2	v_3	f_1	f_3	e_3	e_1	e_6	e_7
e_3	v_3	v_4	f_1	f_4	e_4	e_2	e_7	e_8
e_4	v_4	v_1	f_1	f_5	e_1	e_3	e_8	e_5
e_5	v_1	v_5	f_2	f_5	e_9	e_1	e_4	e_{12}
e_6	v_2	v_6	f_3	f_2	e_{10}	e_2	e_1	e_9
e_7	v_3	v_7	f_4	f_3	e_{11}	e_3	e_2	e_{10}
e_8	v_4	v_8	f_5	f_4	e_{12}	e_4	e_3	e_{11}
e_9	v_5	v_6	f_2	f_6	e_6	e_5	e_{12}	e_{10}
e_{10}	v_6	v_7	f_3	f_6	e_7	e_6	e_9	e_{11}
e_{11}	v_7	v_8	f_4	f_6	e_8	e_7	e_{10}	e_{12}
e_{12}	v_8	v_5	f_5	f_6	e_5	e_8	e_{11}	e_9

vertex	first edge	coordinates
v_1	e_1	$x_1\ y_1\ z_1$
v_2	e_2	$x_2\ y_2\ z_2$
v_3	e_3	$x_3\ y_3\ z_3$
v_4	e_4	$x_4\ y_4\ z_4$
v_5	e_9	$x_5\ y_5\ z_5$
v_6	e_{10}	$x_6\ y_6\ z_6$
v_7	e_{11}	$x_7\ y_7\ z_7$
v_8	e_{12}	$x_8\ y_8\ z_8$

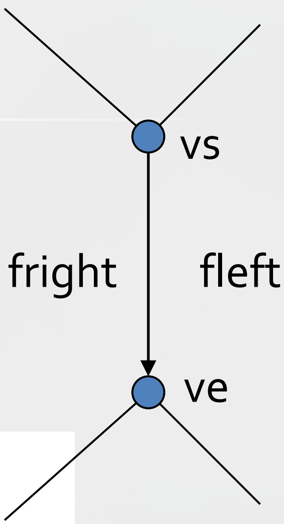
face	first edge
f_1	e_1
f_2	e_9
f_3	e_6
f_4	e_7
f_5	e_{12}
f_6	e_9

(b)

Figure 6.7 The full winged-edge data structure.

Winged Edge Data Structure

```
class Vertex {
    Vec3 pos;
public:
    Vertex() {pos = v1_0;}
    Vertex (double x, double y, double z) {pos = Vec3(x,y,z);}
    void setpos (double x, double y, double z) {pos = Vec3(x,y,z);}
    void printpos() {cout << pos << endl;}
};
```



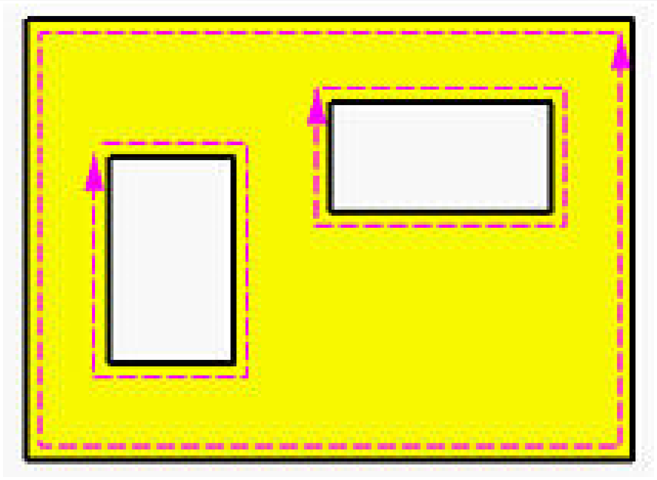

```
class Edge {
    Vertex *vs, *ve;
    Face *fleft, *fright;
public:
    Edge() {fleft = fright = NULL;};
    Edge (Vertex *v1, Vertex *v2) {vs = v1, ve = v2, fleft = fright = NULL;}
    void setLface(Face* f) {fleft = f;}
    void setRface(Face* f) {fright = f;}
    Vertex* startV() {return vs;}
    Vertex* endV() {return ve;}
    bool vertexInE (Vertex* v) {return (v == vs) || (v == ve);}
    void printedge();
};
```

```
class Face {
    Edge* edges[3];
public:
    Face () {};
    void setEdge(int i, Edge* edge) {edges[i] = edge;}
    Edge* findPreE (Edge *e);
};
```

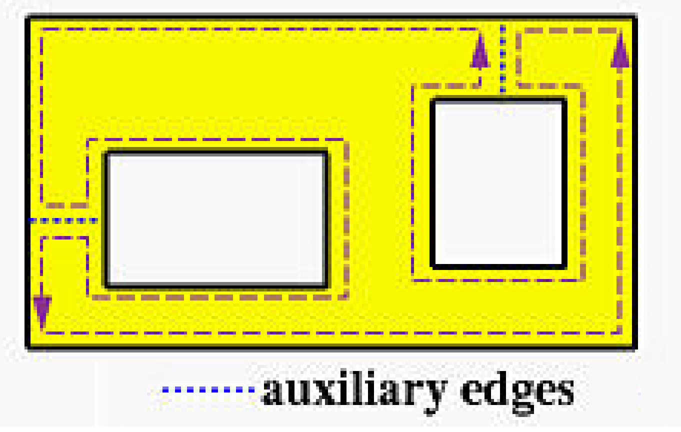
```
class Model {
public:
    vector<Vertex*> vs;
    vector<Edge*>   es;
    vector<Face*>   fs;
};
```

Winged-Edge Data Structure

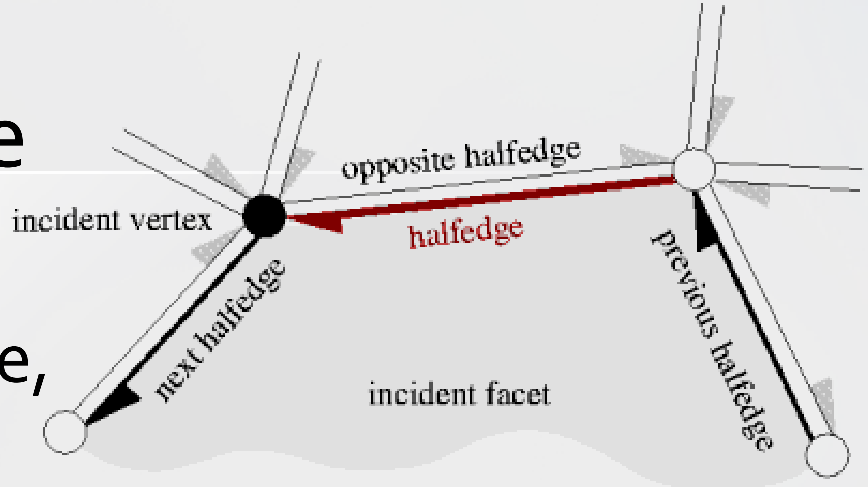
For a face with inner loops are ordered clockwise.



Adding an *auxiliary* edge between each inner loop and the outer loop



Halfedge Data Structure



- Modification of winged edge
- Since every edge is used twice, devise “halfedge” for this use
- Can have loop to account for multiply connected face (face with multiple boundaries)
- Can handle
  - Manifold models
  - Face with boundary
- OpenMesh: a specialized halfedge implementation (for triangular meshes)

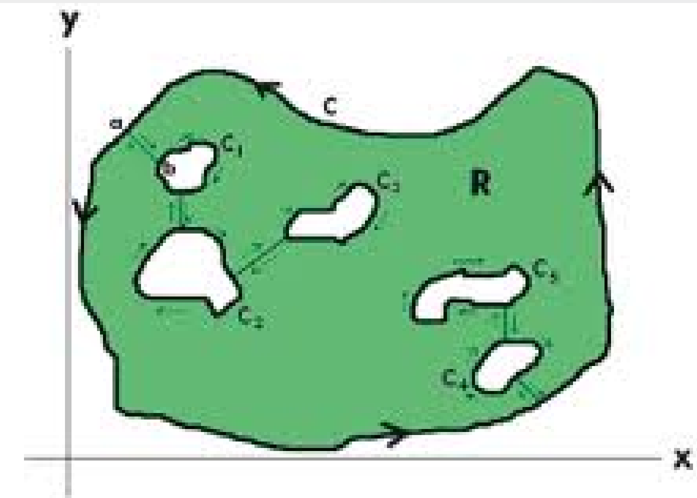
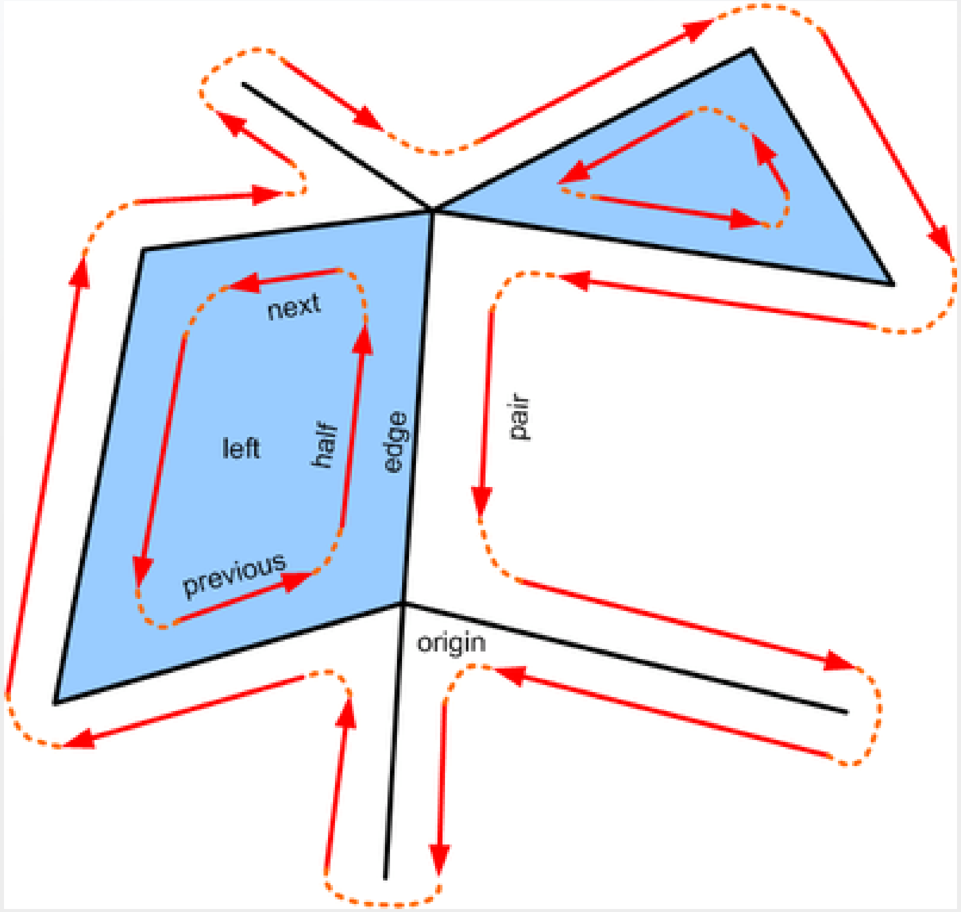


Fig. 1

Half-Edge Data Structure

- Doubly connected edge list




```
struct VertexData;
struct EdgeData;
struct PolygonData;

struct HalfData
{
    HalfData* next;
    HalfData* previous;
    HalfData* pair;

    VertexData* origin;
    PolygonData* left;
    EdgeData* edge;
};

struct VertexData
{
    HalfData* half;
};

struct EdgeData
{
    HalfData* half;
};

struct PolygonData
{
    HalfData* half;
};
```

Object File Format (OFF)

- Storing a description a 2D or 3D object
- Simple extension can handle 4D objects
  - 4D: (x,y,z,w)

- OFF File Characteristics
  - ASCII (there is also a binary version)
  - Color optional
  - 3D
  - No compression

Object File Format(OFF)

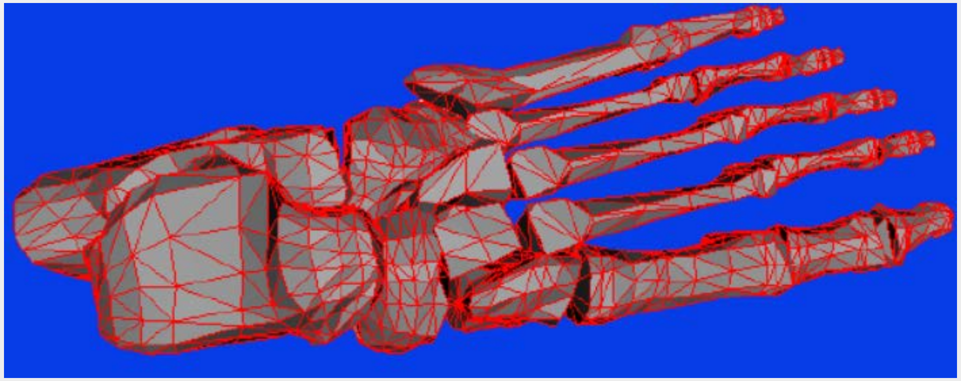
```
[ST][C][N][4][n]OFF         # Header keyword
[Ndim]                      # Space dimension of vertices, present only if nOFF
NVertices  NFaces  NEdges   # NEdges not used or checked

x[0]  y[0]  z[0]            # Vertices, possibly with normals,
                            # colors, and/or texture coordinates, in that order,
                            # if the prefixes N, C, ST
                            # are present.
                            # If 4OFF, each vertex has 4 components,
                            # including a final homogeneous component.
                            # If nOFF, each vertex has Ndim components.
                            # If 4nOFF, each vertex has Ndim+1 components.
...
x[NVertices-1]  y[NVertices-1]  z[NVertices-1]
                            # Faces
                            # Nv = # vertices on this face
                            # v[0] ... v[Nv-1]: vertex indices
                            #       in range 0..NVertices-1
Nv  v[0] v[1] ... v[Nv-1]  colorspec
...
                            # colorspec continues past v[Nv-1]
                            # to end-of-line; may be 0 to 4 numbers
                            # nothing: default
                            # integer: colormap index
                            # 3 or 4 integers: RGB[A] values 0..255
                            # 3 or 4 floats: RGB[A] values 0..1
```

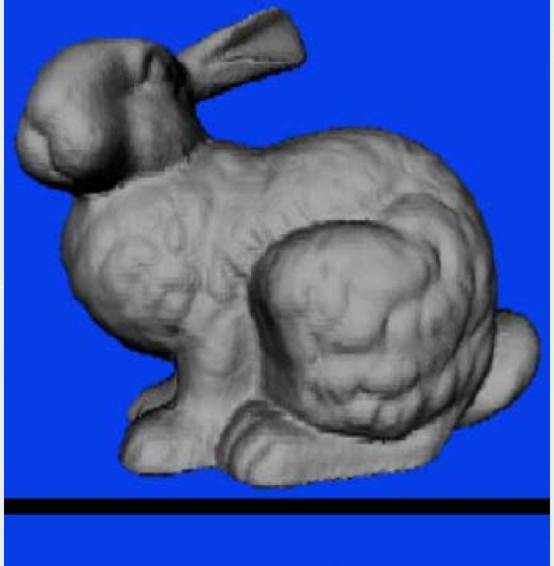
Object File Format(OFF)

```
OFF
#
#  cube.off
#  A cube.
#  There is extra RGBA color information specified for the faces.
#
8 6 12
  1.632993   0.000000   1.154701
  0.000000   1.632993   1.154701
 -1.632993   0.000000   1.154701
  0.000000  -1.632993   1.154701
  1.632993   0.000000  -1.154701
  0.000000   1.632993  -1.154701
 -1.632993   0.000000  -1.154701
  0.000000  -1.632993  -1.154701
  4  0 1 2 3  1.000 0.000 0.000 0.75
  4  7 4 0 3  0.300 0.400 0.000 0.75
  4  4 5 1 0  0.200 0.500 0.100 0.75
  4  5 6 2 1  0.100 0.600 0.200 0.75
  4  3 2 6 7  0.000 0.700 0.300 0.75
  4  6 5 4 7  0.000 1.000 0.000 0.75
```

Polygon File Format

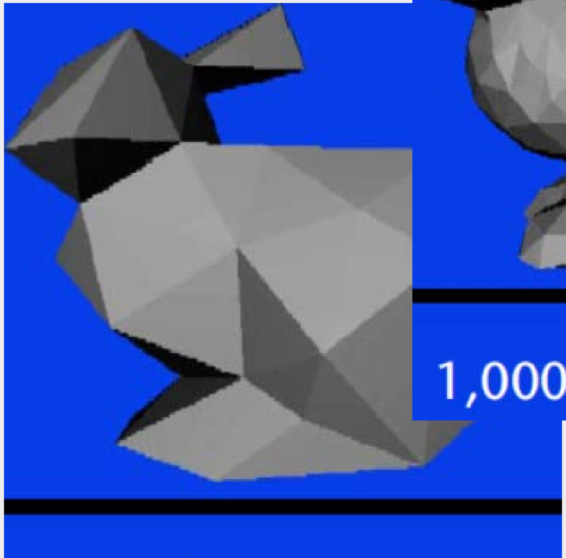
- Stanford Triangle Format
- Store 3-d data from 3D scanners
- Properties can be stored including
  - color and transparency
  - surface normals
  - texture coordinates
  - data confidence values





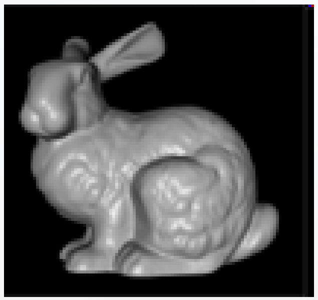
69,451 faces

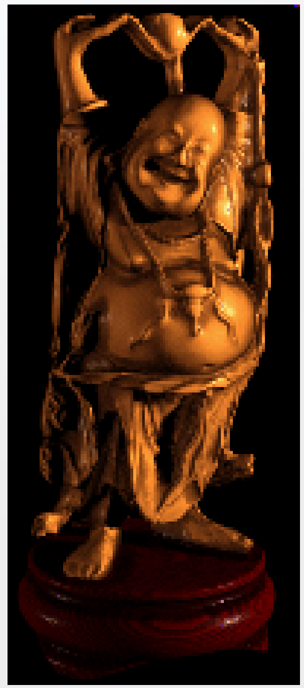
1,000 faces (30 sec)

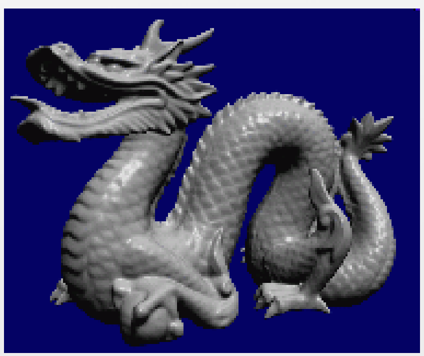


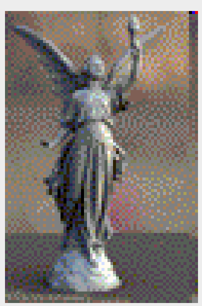
100 faces (30 sec)

Stanford 3D Scanning Repository (url)







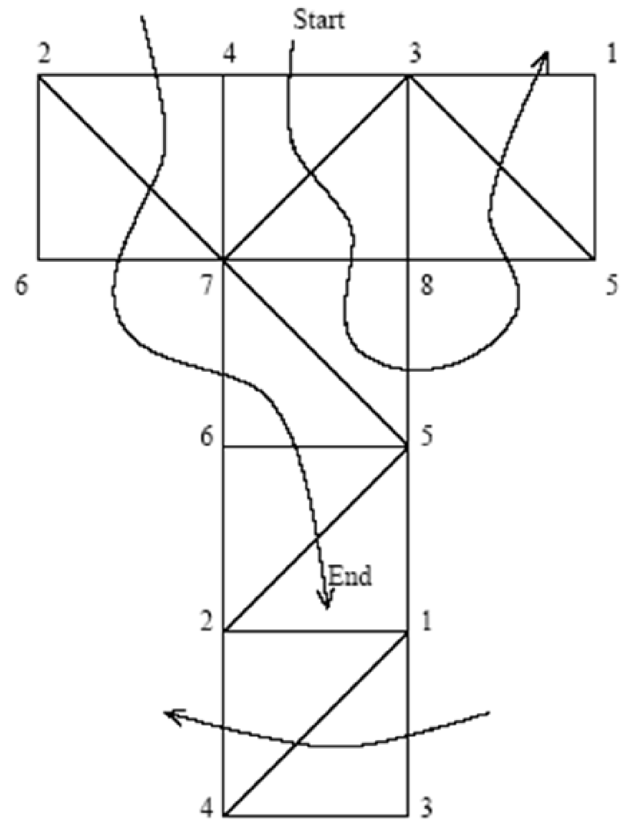


Cyberware 3D Scanners (url)

Large models also avaiable at GeogiaTech

Polygon File Format

- PLY structure
  - Header
  - Vertex List
  - Face List
  - (lists of other elements)



Strip: 4 3 7 8 5 3 1 4 2 7 6 5 2 1

Triangulating a cube
for one sequential strip.

Polygon File Format

```
ply
format ascii 1.0                                { ascii/binary, format version number }
comment made by anonymous                       { comments keyword specified, like all lines }
comment this file is a cube
element vertex 8                                { define "vertex" element, 8 of them in file }
property float32 x                              { vertex contains float "x" coordinate }
property float32 y                              { y coordinate is also a vertex property }
property float32 z                              { z coordinate, too }
element face 6                                  { there are 6 "face" elements in the file }
property list uint8 int32 vertex_index          { "vertex_indices" is a list of ints }
end_header                                      { delimits the end of the header }
0 0 0                                           { start of vertex list }
0 0 1
0 1 1
0 1 0
1 0 0
1 0 1
1 1 1
1 1 0
4 0 1 2 3                                       { start of face list }
4 7 6 5 4
4 0 4 5 1
4 1 5 6 2
4 2 6 7 3
4 3 7 4 0
```

Scaling Transformations

affect geometry but not topology of object

primitive shapes

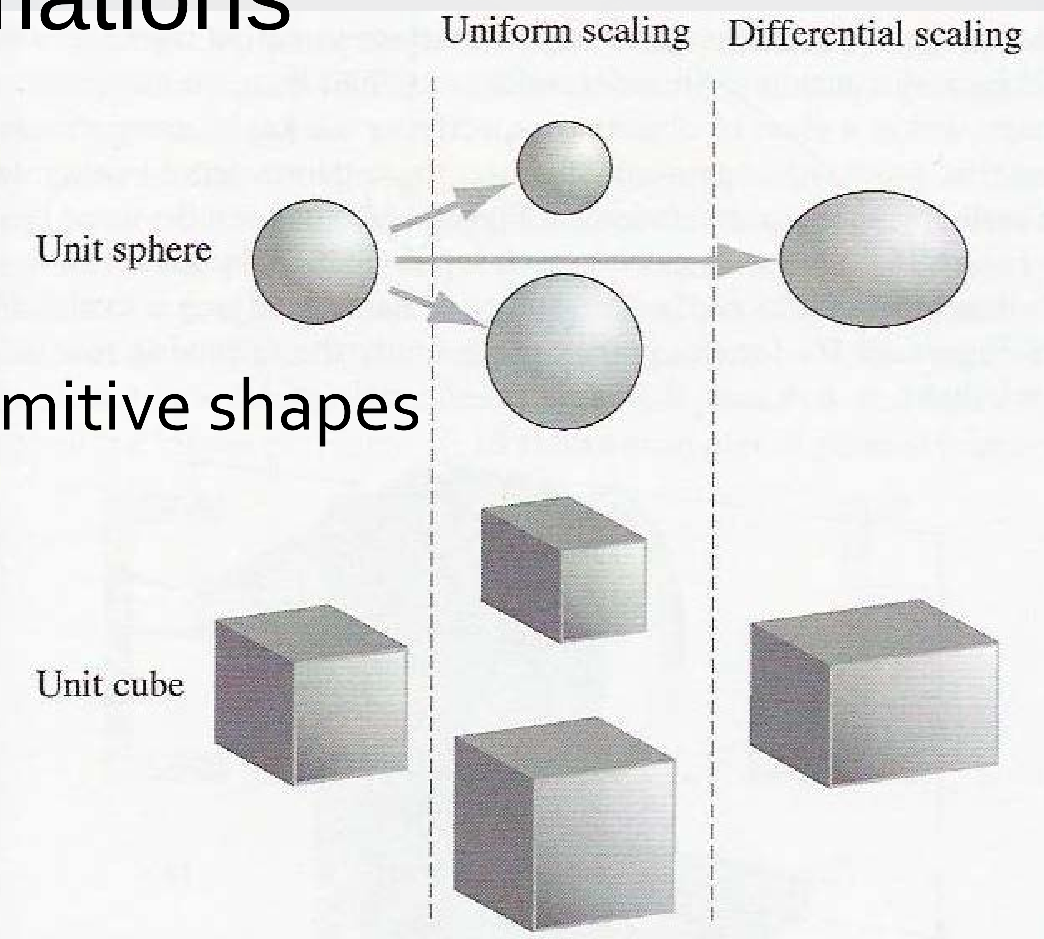
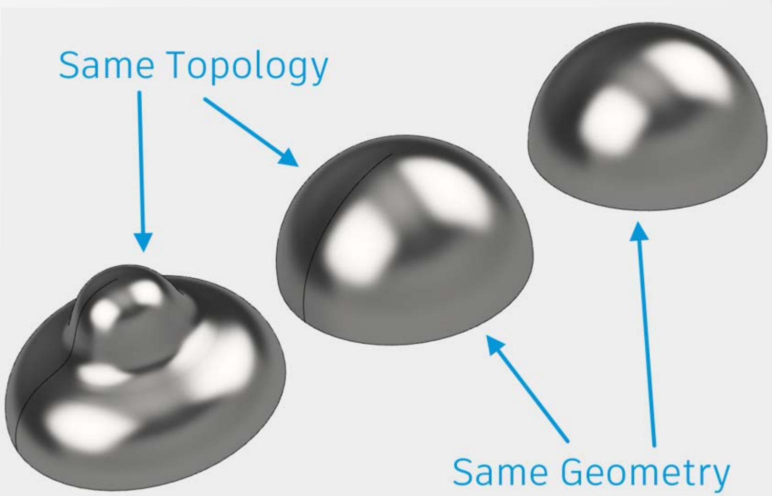


Figure 10.14 Instances



- Topology: ***faces***, ***edges*** and ***vertices***.
- Geometry: ***surfaces***, ***curves*** and ***points***.

Source: Mortenson, Ch 10

Differential Scaling Transformations

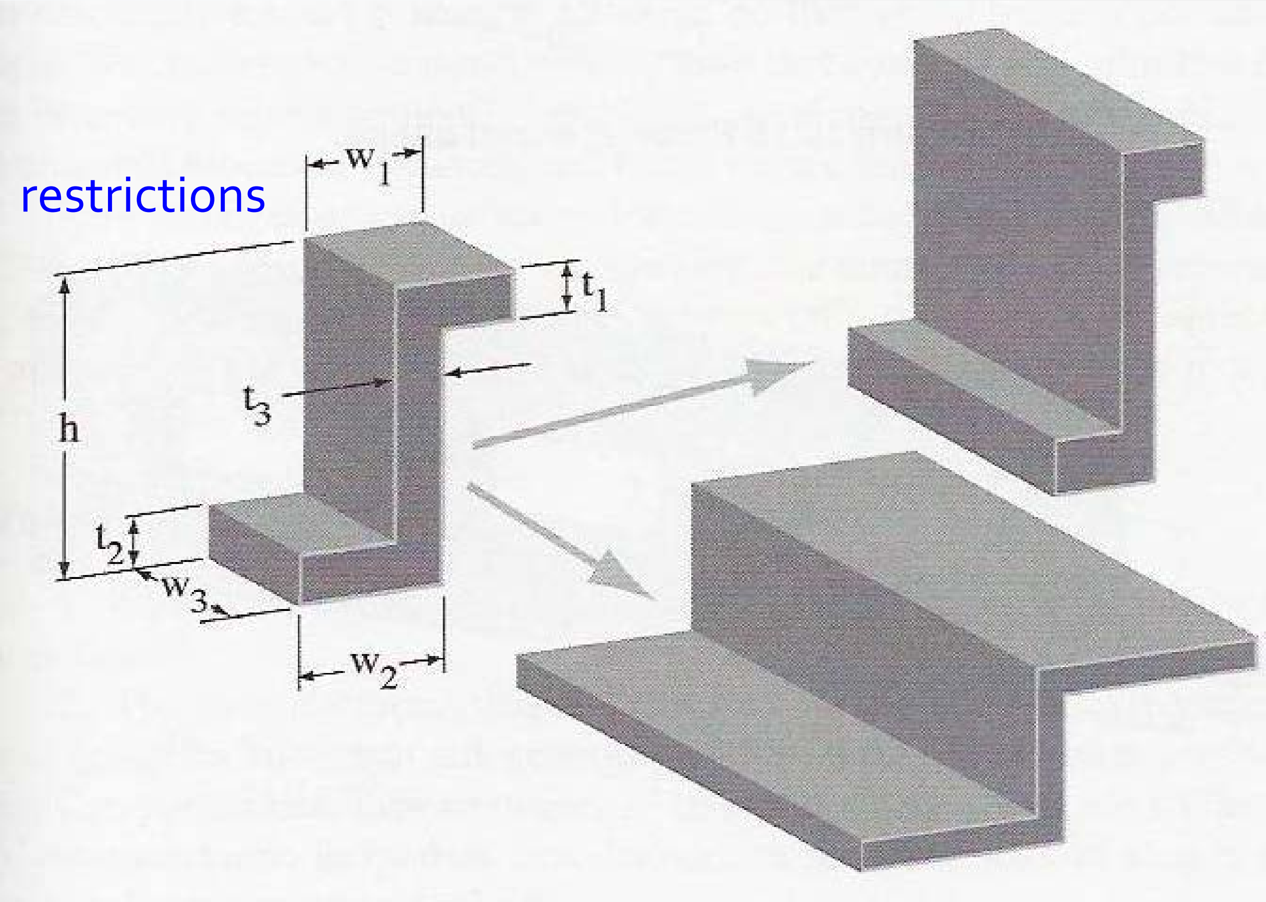


Figure 10.15 Instances of a "Z" section.

Source: Mortenson, Ch 10

Differential Scaling Transformations

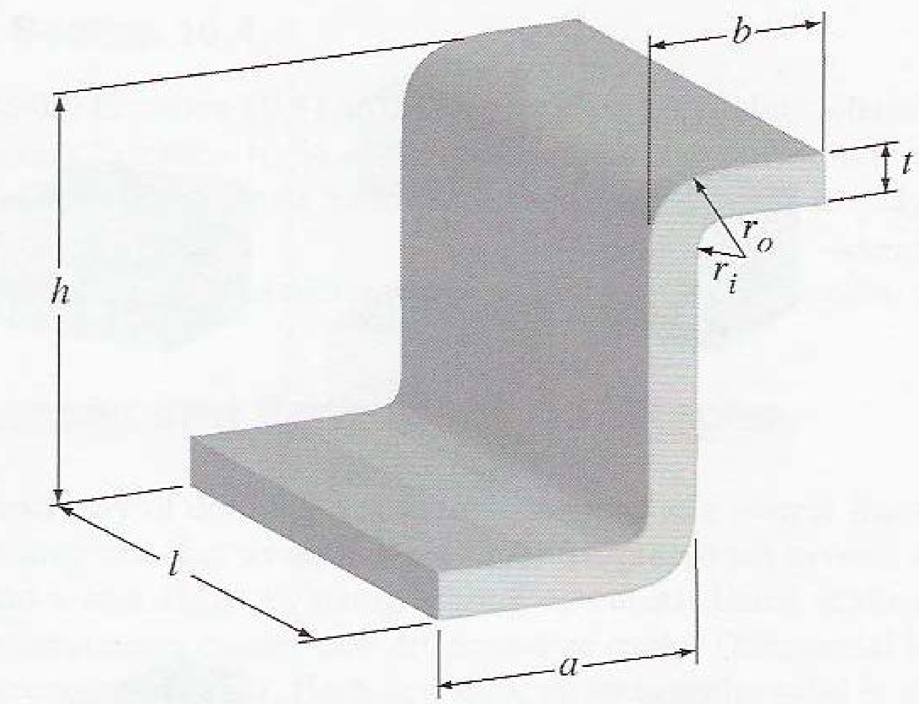


Figure 10.16 Parameterized shape.

Sample restrictions: $a, b, h, l, t > 0, \quad b \leq a, \quad a > 2t, \quad h > 4t$

Source: Mortenson, Ch 10

Parameterized Shape of Variable Topology

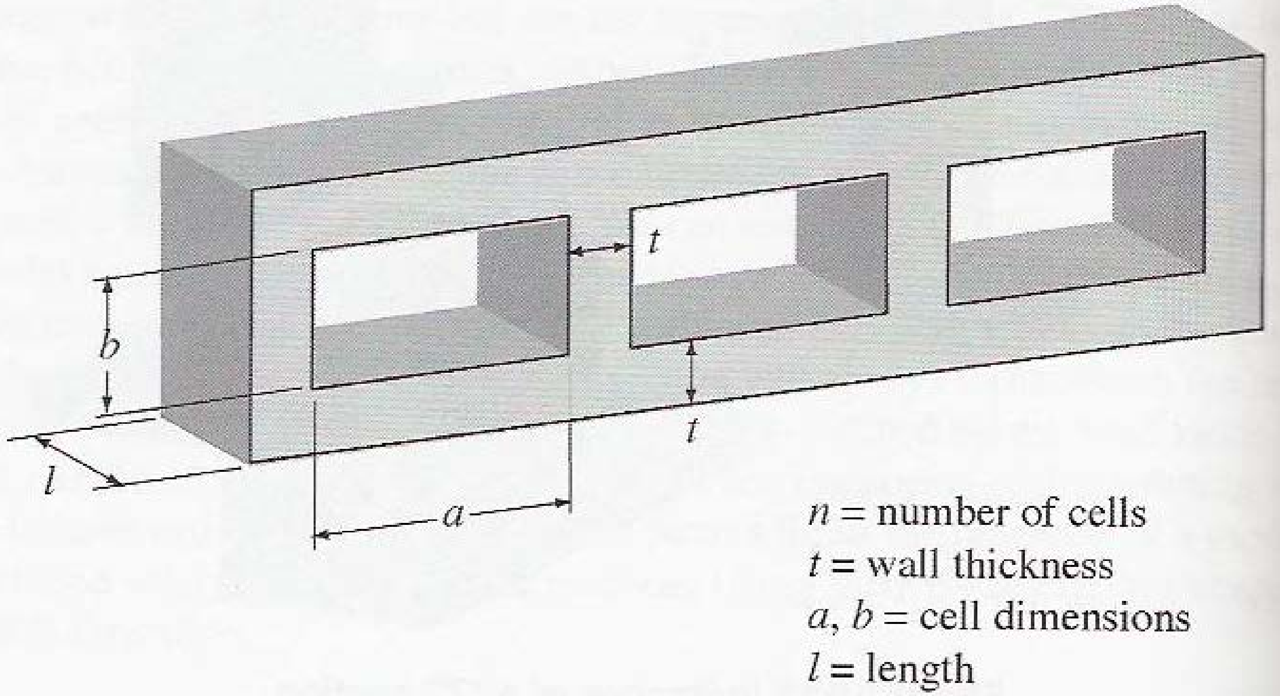


Figure 10.17 Parameterized shape of variable topology.

Source: Mortenson, Ch 10

Sweep Solids

Moving an object along a path.

- Generator = sweeping object: curve, surface, or solid
- Director = path

Common for modeling constant cross-section mechanical parts.

Translational sweep (extrusion): moving a planar curve or planar shape along a straight line normal to plane of curve.

More generally, sweep one curve along another.

Rotational sweep: rotating a planar curve or shape (with finite area) about an axis.

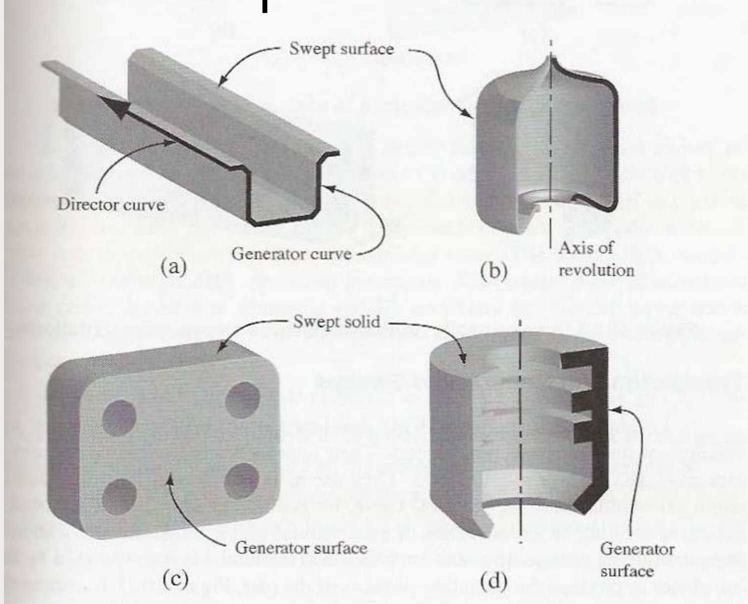


Figure 10.19 Examples of sweep shapes.

Source: Mortenson, Ch 10

Sweep Solids

some problematic situations

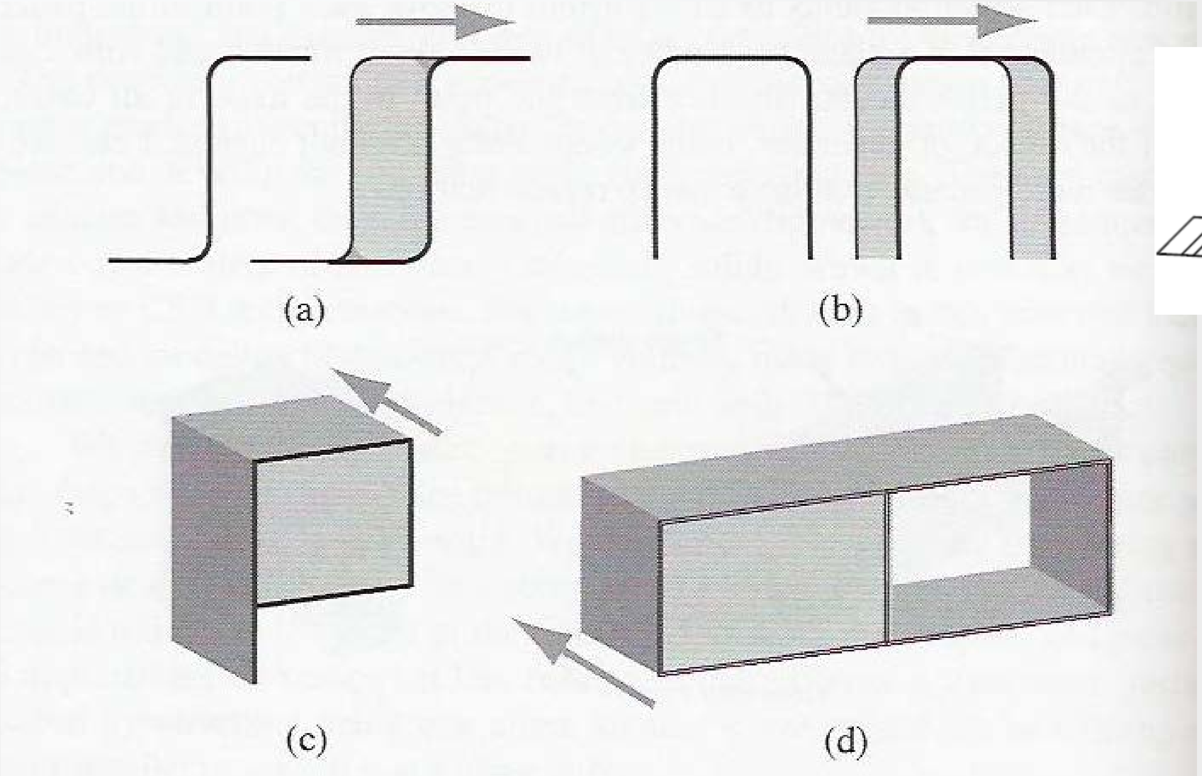
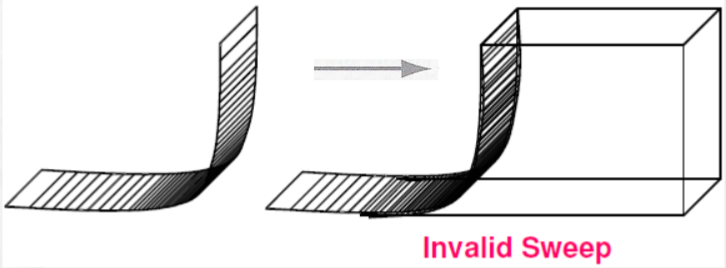


Figure 10.20 Dimensionally nonhomogeneous sweep representations.



Source: Mortenson, Ch 10

Loss and Eshleman (1974) Position and Direction Specification for Swept Solids

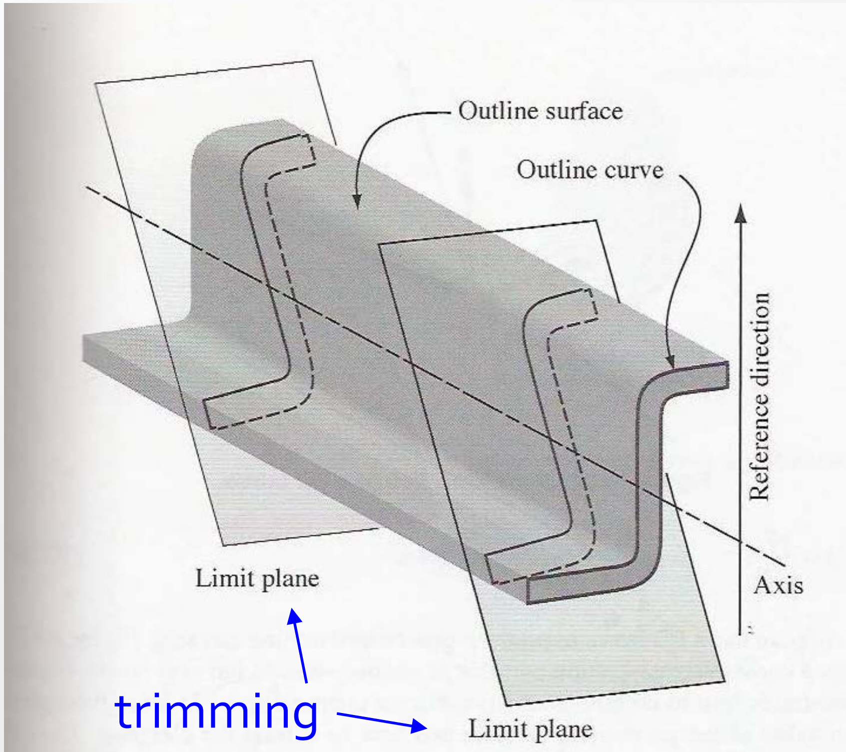


Figure 10.21 Outline surface of a constant cross-section solid.

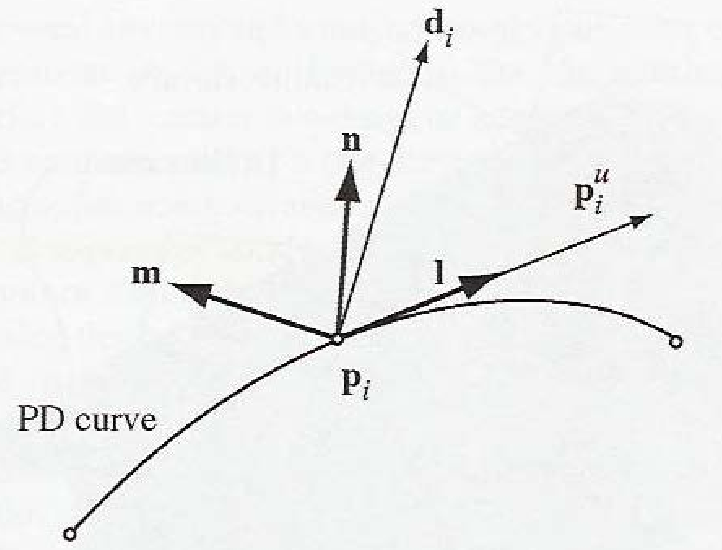


Figure 10.22 Characteristics of a PD curve.

$$\mathbf{l} = \frac{\mathbf{p}_i^u}{|\mathbf{p}_i^u|}, \qquad \mathbf{m} = \frac{\mathbf{d}_i \times \mathbf{p}_i^u}{|\mathbf{d}_i \times \mathbf{p}_i^u|}, \qquad \mathbf{n} = \mathbf{l} \times \mathbf{m}$$

Source: Mortenson, Ch 10

Loss and Eshleman (1974) Position and Direction Specification for Swept Solids

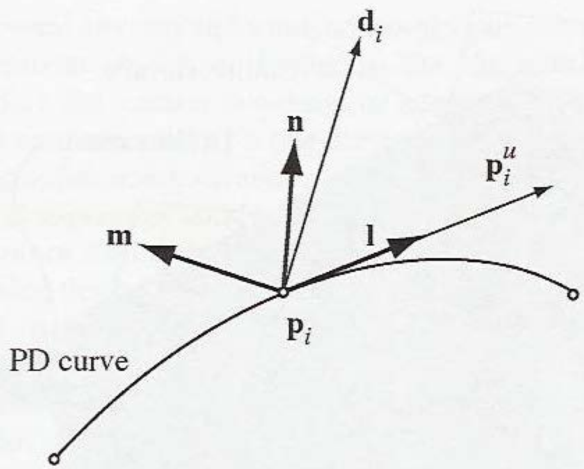


Figure 10.22 Characteristics of a PD curve.

$$\mathbf{l} = \frac{\mathbf{p}_i^u}{\left|\mathbf{p}_i^u\right|}, \qquad \mathbf{m} = \frac{\mathbf{d}_i \times \mathbf{p}_i^u}{\left|\mathbf{d}_i \times \mathbf{p}_i^u\right|}, \qquad \mathbf{n} = \mathbf{l} \times \mathbf{m}$$

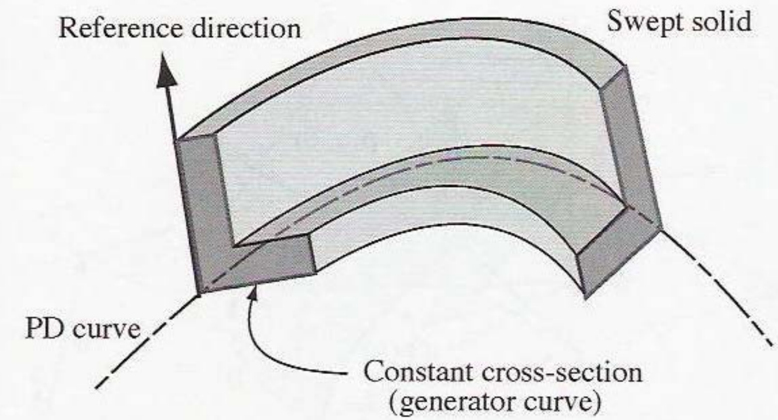


Figure 10.23 A constant cross-section part that curves and twists.

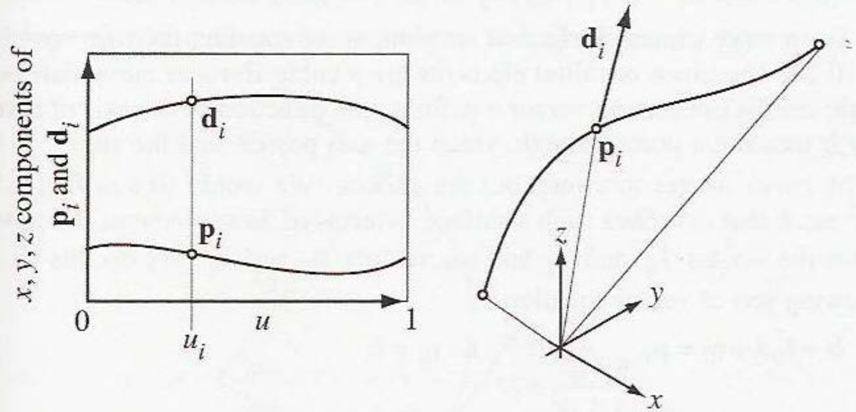


Figure 10.24 Components of a PD curve.

Source: Mortenson, Ch 10

Surfaces of Revolution

Example: z-axis of rotation

$$\mathbf{p}(u) = \mathbf{x}(u) + \mathbf{z}(u)$$

$$\mathbf{p}(u,\theta) = \mathbf{x}(u)\cos\theta + \mathbf{x}(u)\sin\theta + \mathbf{z}(u)$$

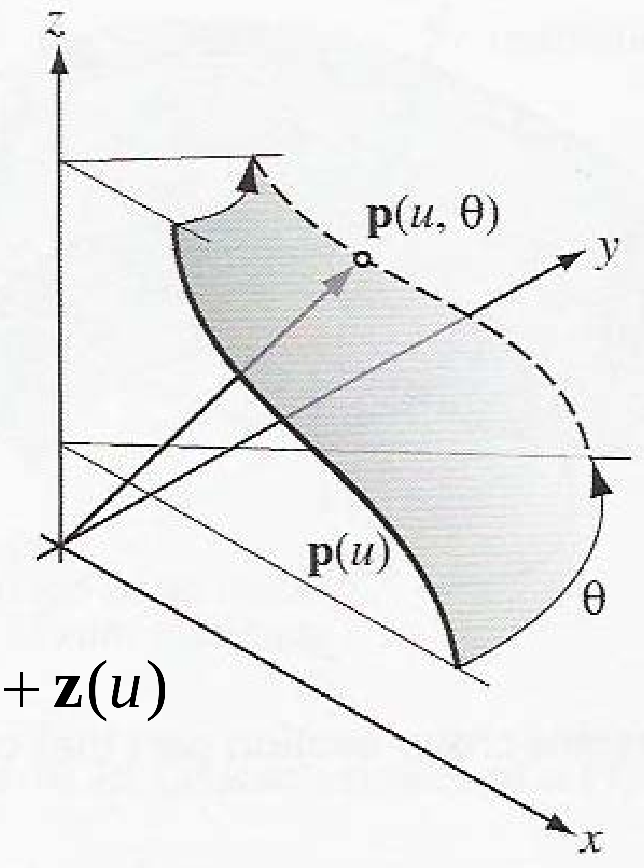


Figure 10.25 Surface of revolution.

Source: Mortenson, Ch 10

Surfaces of Revolution

More general example using cubic Hermite curve: goal is to find a Hermite patch describing the surface.

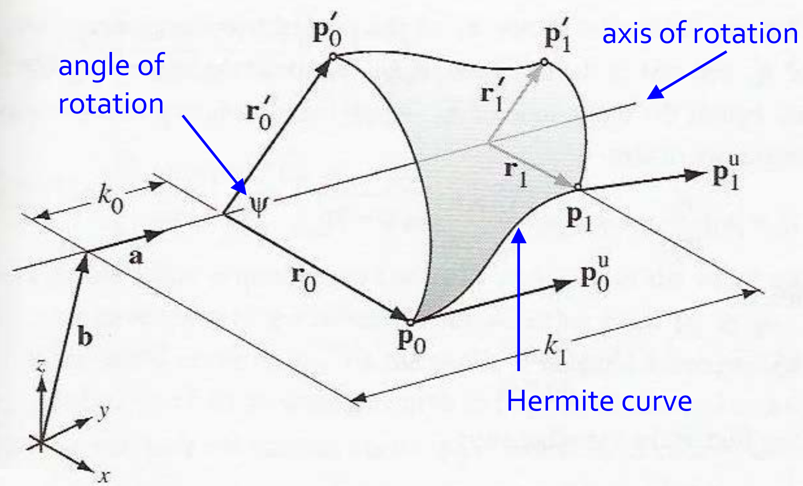


Figure 10.26 Another surface of revolution.

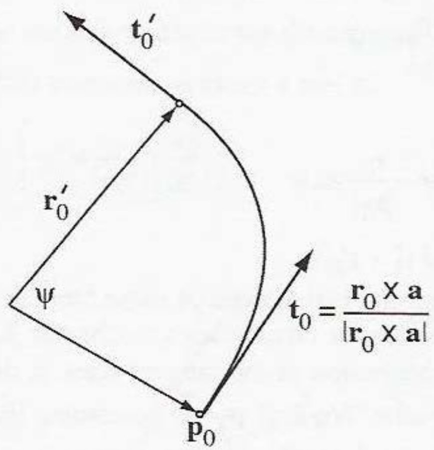


Figure 10.27 Circumferential tangent vectors of a surface of revolution.

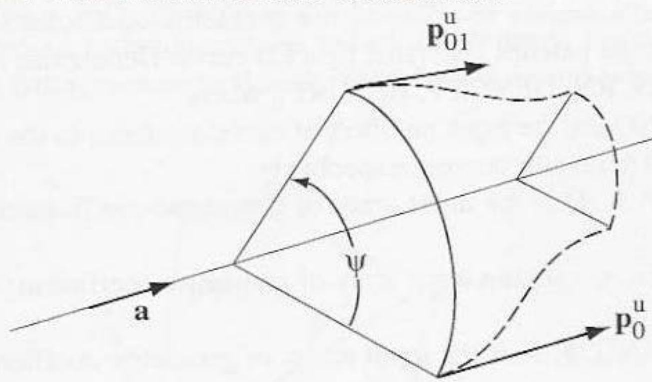
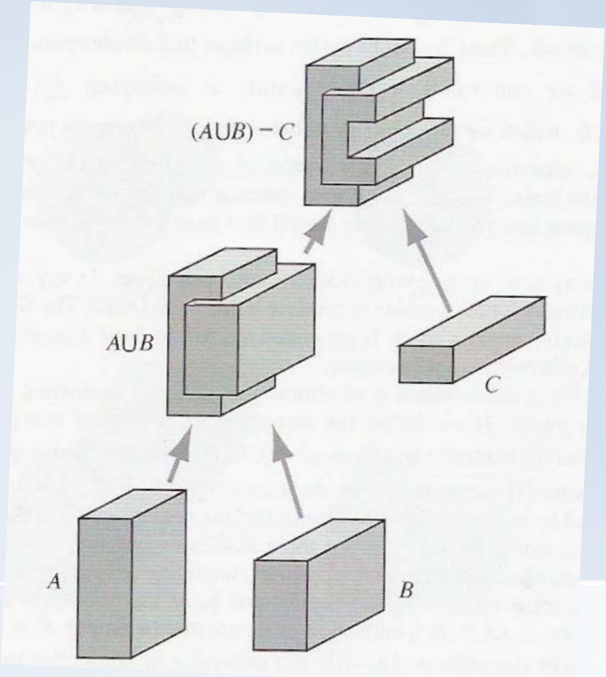


Figure 10.28 Axial tangent vectors of a surface of revolution.

Source: Mortenson, Ch 10

Geometric Modeling 91.580.201

Mortenson Chapter 11



Complex Model Construction

Topics

- Topology of Models
  - Connectivity and other intrinsic properties
- Graph-Based Models
  - Emphasize topological structure
- Boolean Models
  - Set theory, set membership classification, Boolean operators
- Boolean Model Construction
- Constructive Solid Geometry
- Boundary Models (B-Rep)

Model Topology

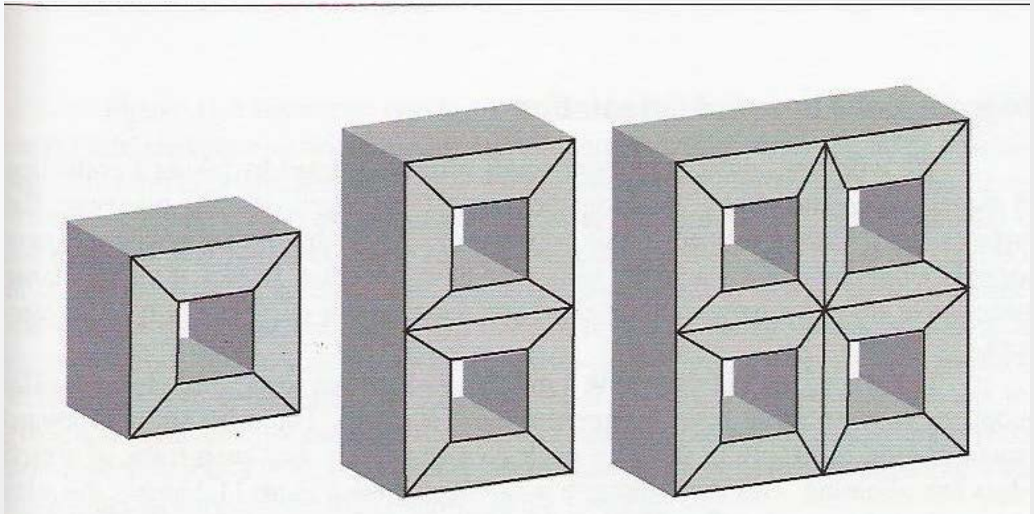


Figure 11.1 Examples of nonsimple polyhedra.

Euler's Formula for 3D Polyhedra: $$V - E + F = 2$$

Poincare's Generalization to n-Dimensional Space: $$N_0 - N_1 + N_2 - \ldots = 1 - (-1)^n$$

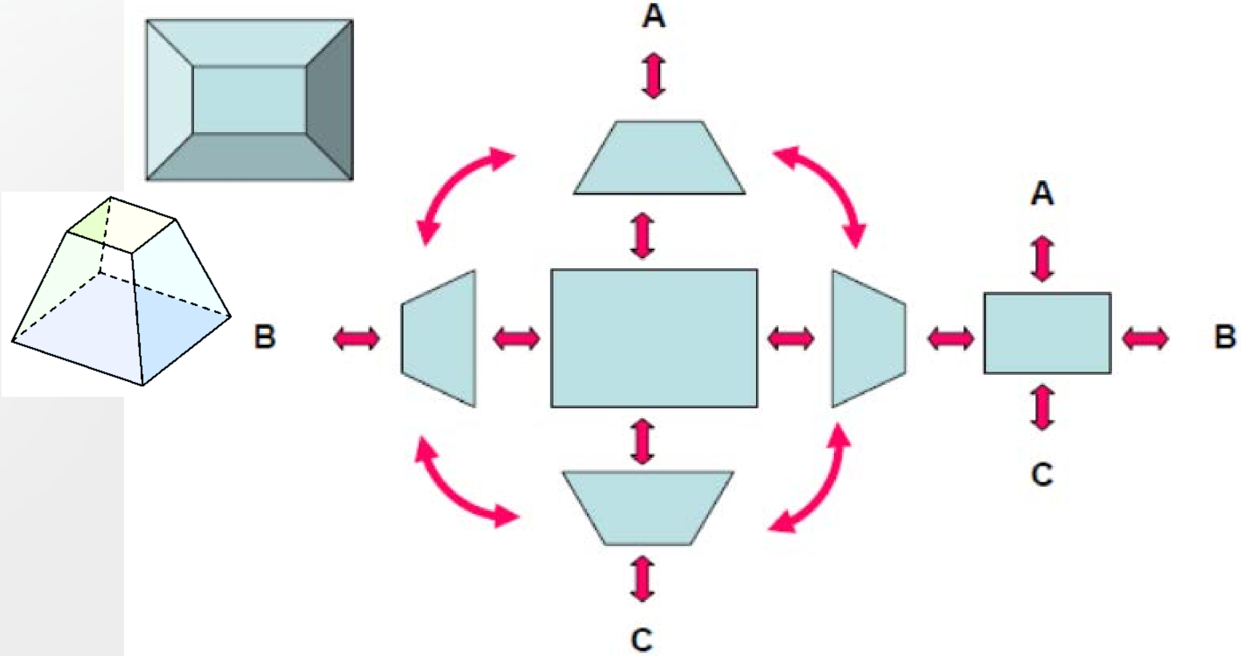
typo fixed

Euler-Poincare Formula:
(G = *genus* = number of "handles") $$V - E + F - 2(1 - G) = 0$$

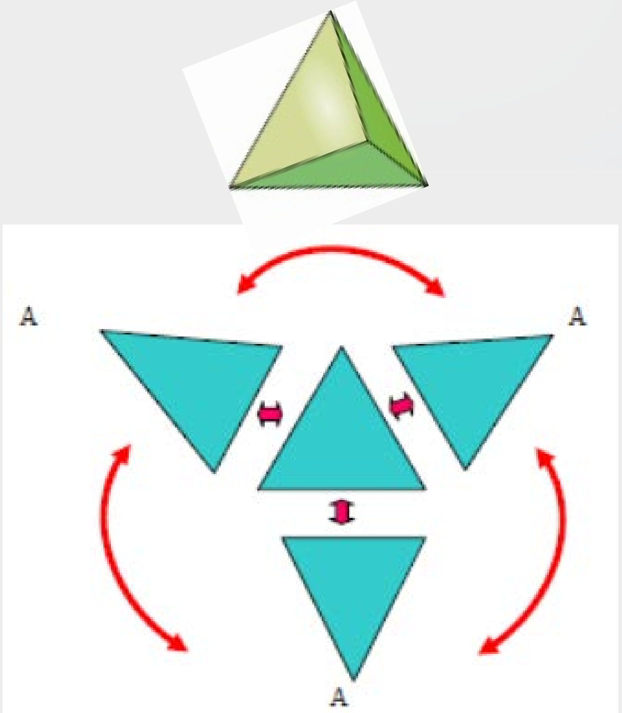
Model Topology (continued)

Topological Atlas and Orientability

The simplest data structure keeps track of adjacent edges. Such a data structure is called an atlas.



Topological Atlas of a Tetrahedron



Model Topology (continued)

2 ways to join a pair of edges (match numbers)

preserves orientation

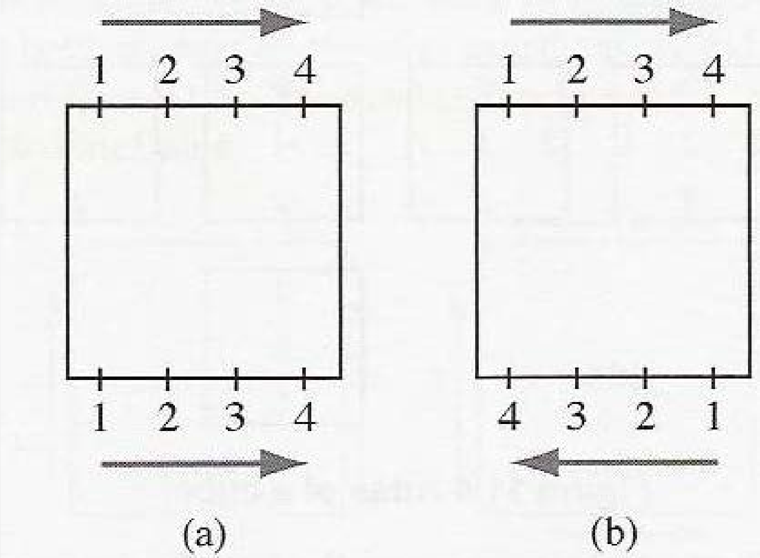


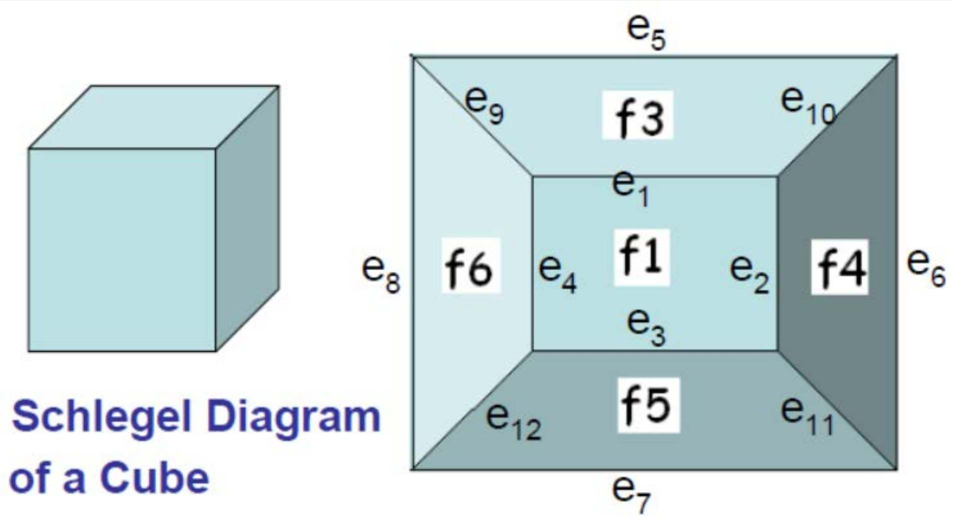
Figure 11.3 Orientation.

reverses orientation

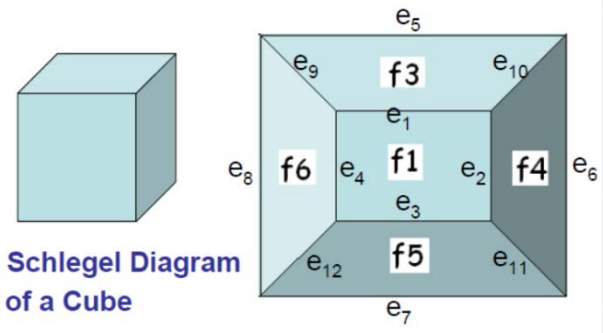
Topological Atlas
and Orientability :
The orientability indicated with arrows or numbers as shown above.
We see that the **orientation preserving** arrows are in two **opposite** rotational directions i.e., clockwise and anticlockwise.
While **orientation reversing** arrows are in **the same** rotational directions.

Schlegel Diagrams

A common form of embedding graphs on planar faces is called **Schlegel Diagram**. It is a projection of its combinatorial equivalent of the vertices, edges and faces of the embedded boundary graph on to its surface. Here the edges may not cross except at their incident vertices and vertices may not coincide.

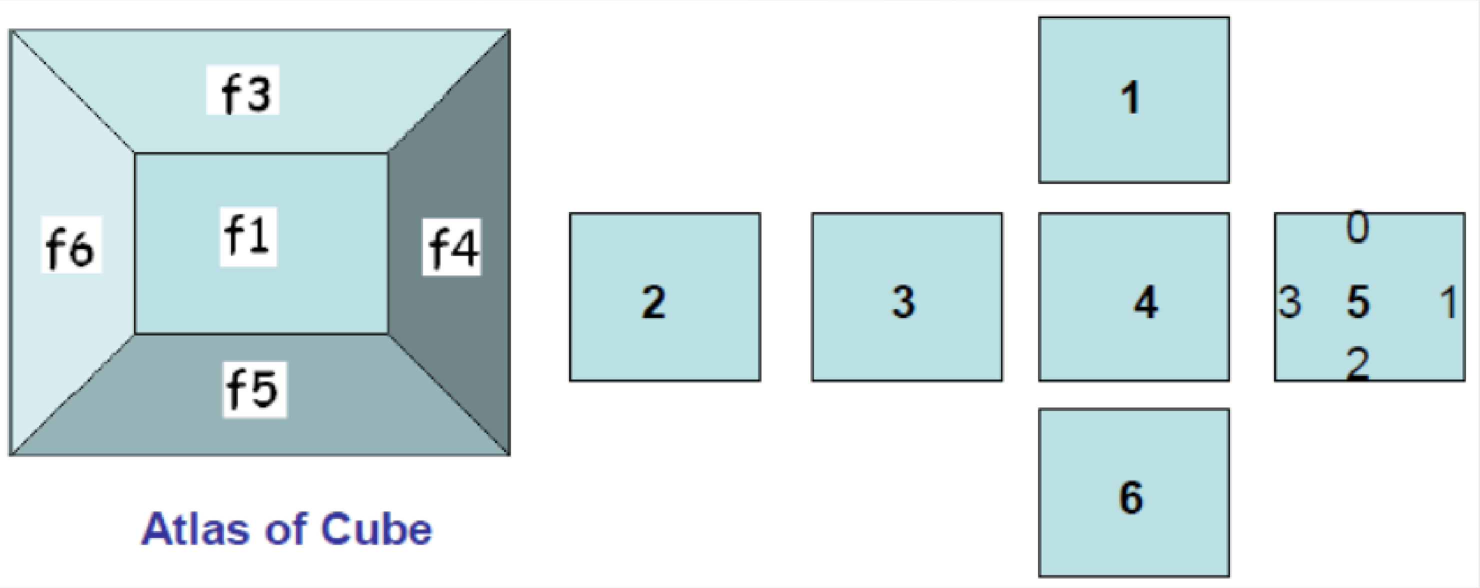


Atlas of Cube



Schlegel Diagram of a Cube

An atlas of a cube can also be given by the arrangement of its faces as shown below



Atlas of Cube

Model Topology (continued)

Atlas of a cube

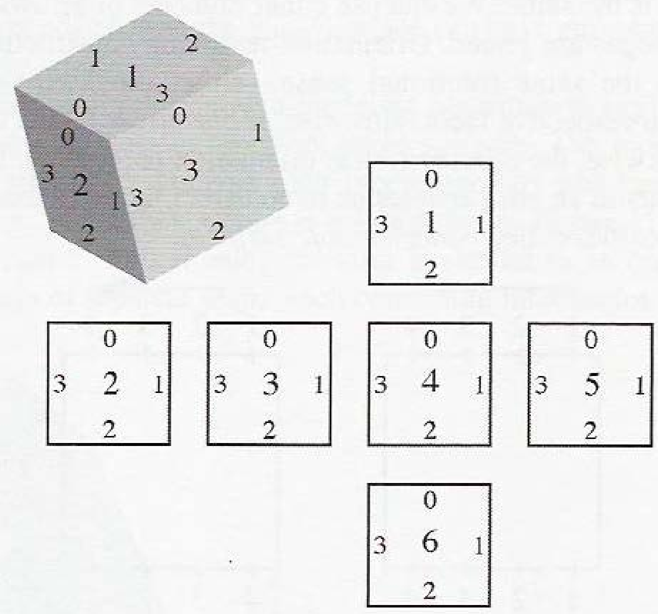


Figure 11.4 Atlas of a cube.

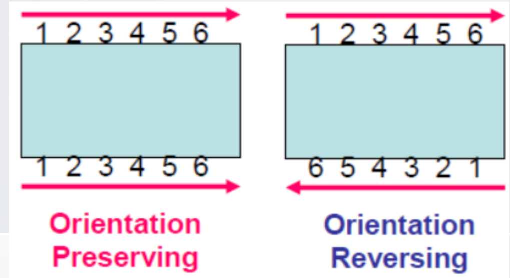
Edge 0 of face 1 matched with edge 0 of face 2…

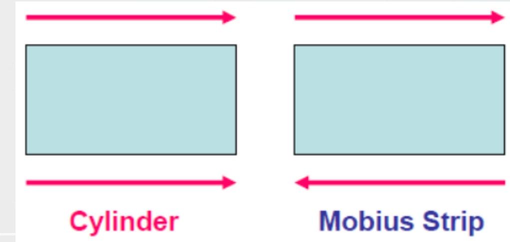
[(1,0) (2,0)] [(1,1) (5,0)] [(1,2) (4,0)] [(1,3) (3,0)]

[(2,1) (3,3)] [(2,2) (6,2)] [(2,3) (5,1)] [(3,1) (4,3)]

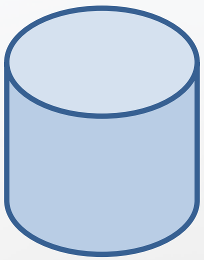
[(3,2) (6,3)] [(4,1) (5,3)] [(4,2) (6,0)] [(5,2) (6,1)]

Model Topology (continued)





Some examples of Atlases



Cylinder: orientable

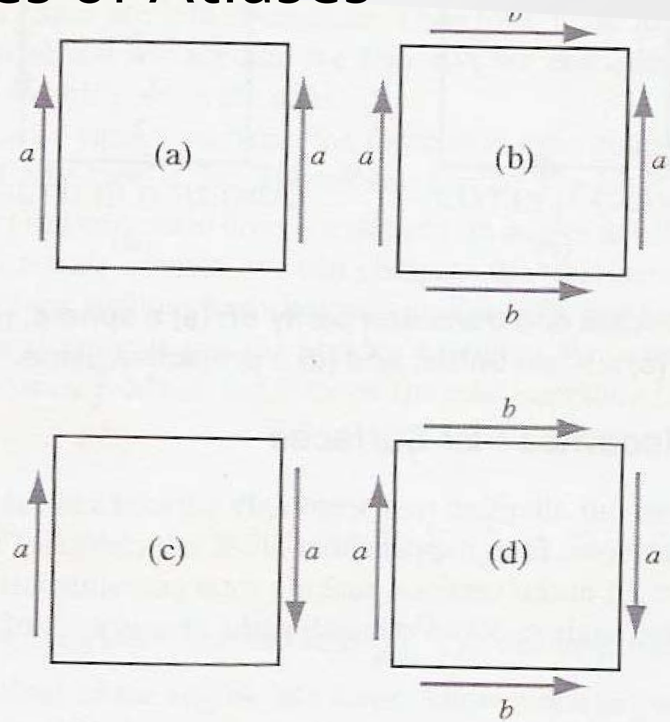
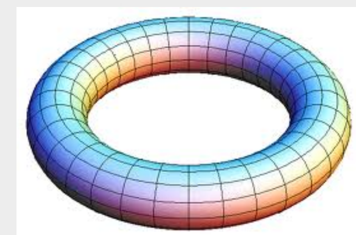
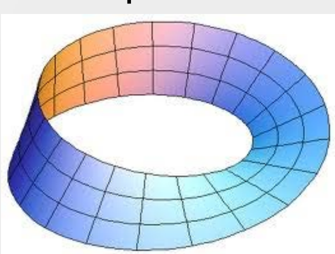


Figure 11.5 Atlas of: (a) a cylinder; (b) a torus; (c) a Möbius strip; and (d) a Klein bottle.

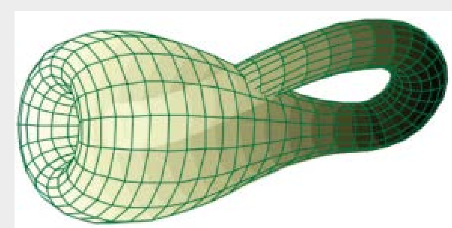
Torus: orientable



Mobius strip: non-orientable, open surface

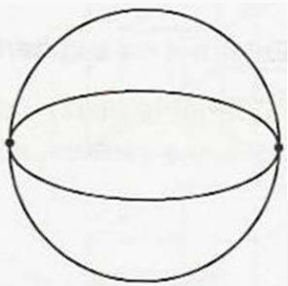


Klein bottle: non-orientable and does not fit into 3D without self-intersections



Orientability is intrinsically defined: left and right are never reversed.
Non-orientable: right & left are not intrinsically defined.

Model Topology (continued)



Sphere:
orientable

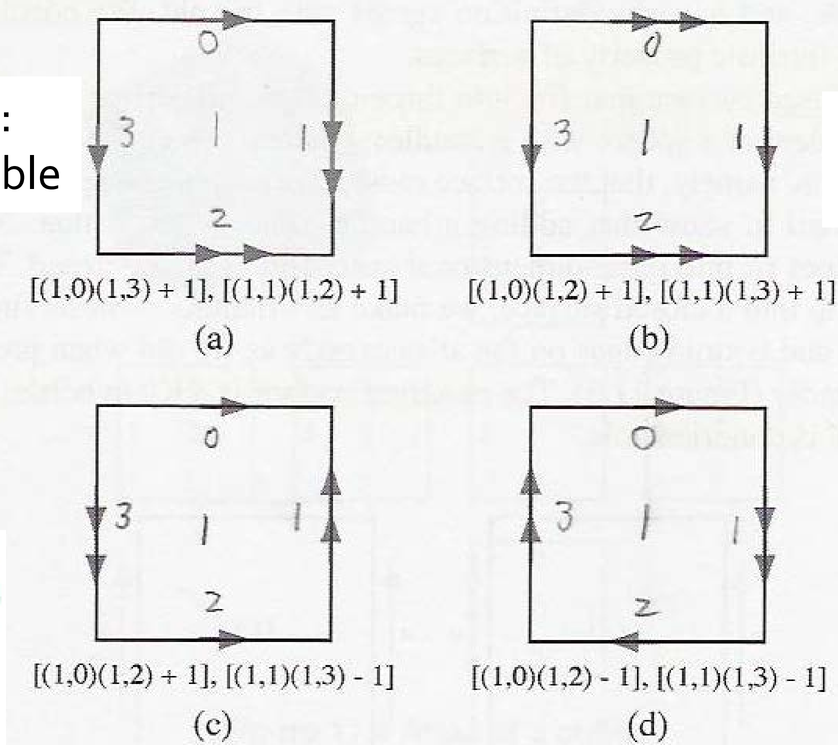
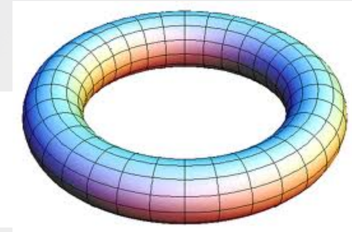
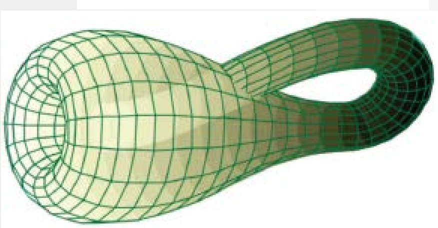


Figure 11.6 Atlas and transition parity of: (a) a sphere; (b) a torus; (c) a Klein bottle; and (d) a projective plane.

Torus:
orientable



Klein bottle:
non-orientable



Projective plane:
non-orientable

Transition Parity = 1 means match up normally.
Transition Parity = -1 means match up in reverse.

Model Topology (continued)

- Curvature of piecewise flat surfaces
  - Curvature concentrated at vertices
  - Sum up angle "excesses" of small paths around each vertex. Let:
    - E_i be <u>*excess*</u> of a path around vertex *i*.
    - T_i be <u>*total turning*</u> of a path around vertex *i*.

$$K = \sum_{i=1}^{V} E_i = \sum_{i=1}^{V} (2\pi - T_i) = 2\pi V - \sum_{i=1}^{V} T_i = 2\pi V - \sum_{i=1}^{F} f_i = 2\pi(V - E + F)$$

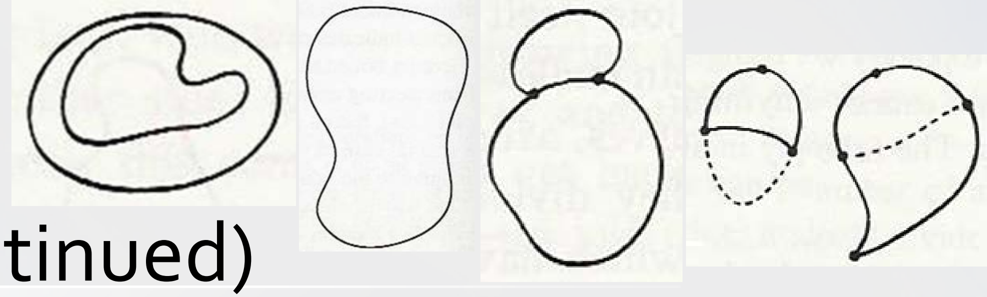
f_i = sum of interior angles of face *i*.

- where last part is for closed, piecewise flat surface

Note this does not require knowledge of how edges are joined.

$$\chi = V - E + F \quad \text{so} \quad K = 2\pi\chi$$

χ = Euler characteristic, which is an intrinsic, topological invariant.



Model Topology (continued)

- Topology of Closed, Curved Surfaces
  - Net = arbitrary collection of simple arcs (terminated at each end by a vertex) that divide the surface everywhere into topological disks.
  - All valid nets on the same closed surface have the same Euler characteristic.

- 2 elementary net transformations
  - Adding (or deleting) a face by modifying an edge
  - Adding (or deleting) a vertex

  χ is invariant under these net transformations.

Model Topology (continued)

- Euler Operators
  - Euler Object = connected network of faces, vertices, edges
  - All valid nets on the same closed surface have the same Euler characteristic.
- Euler's formula for polyhedra requires:
  - All faces are topological disks.
  - Object's complement is connected.
  - Each edge adjoins 2 faces with vertex at each end.
  - At least 3 edges meet at each vertex.

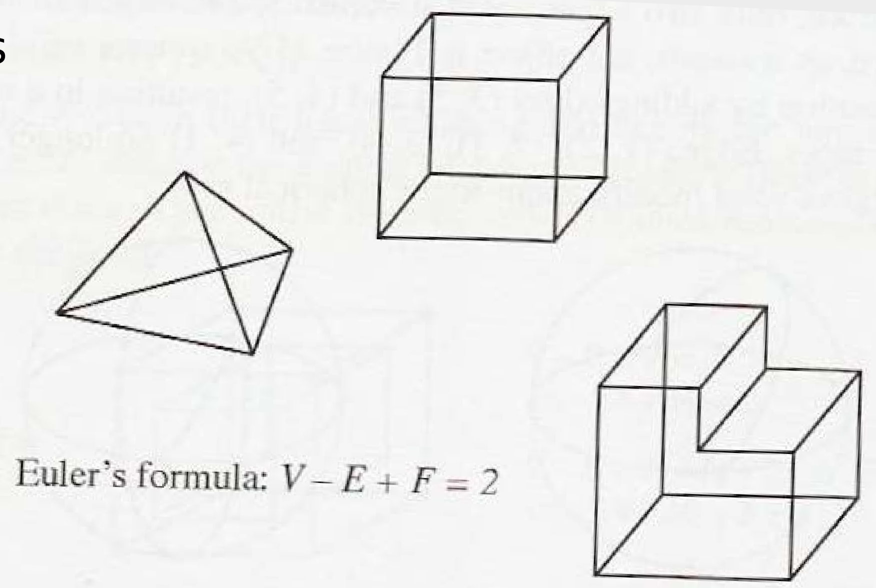


Figure 11.7 Euler's formula and simple polyhedra.

Model Topology (continued)

- Spherical net example
  - Nets are proper:
    - collection of simple arcs (edges)
    - terminated at each end by a vertex
    - divide surface into topological disks
  - Curving edges preserves validity of Euler's formula

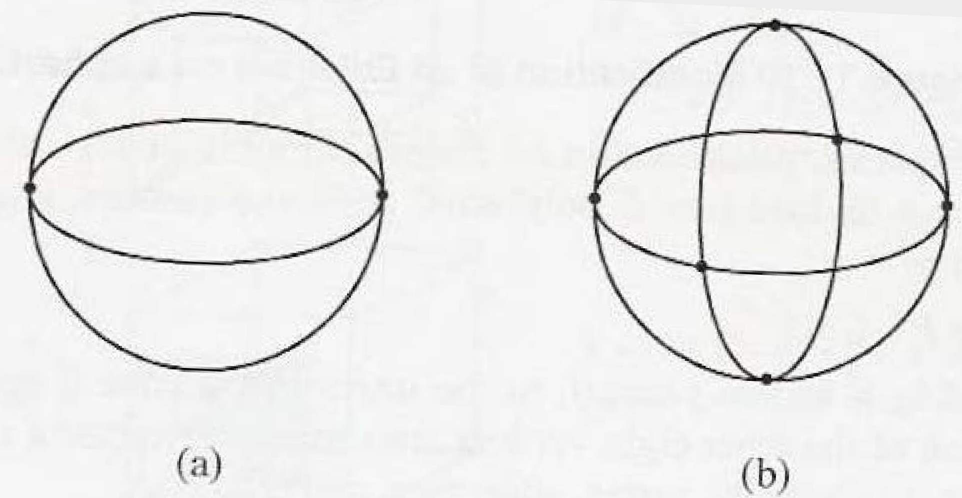


Figure 11.8 Euler's formula applied to a spherical net.

Model Topology (continued)

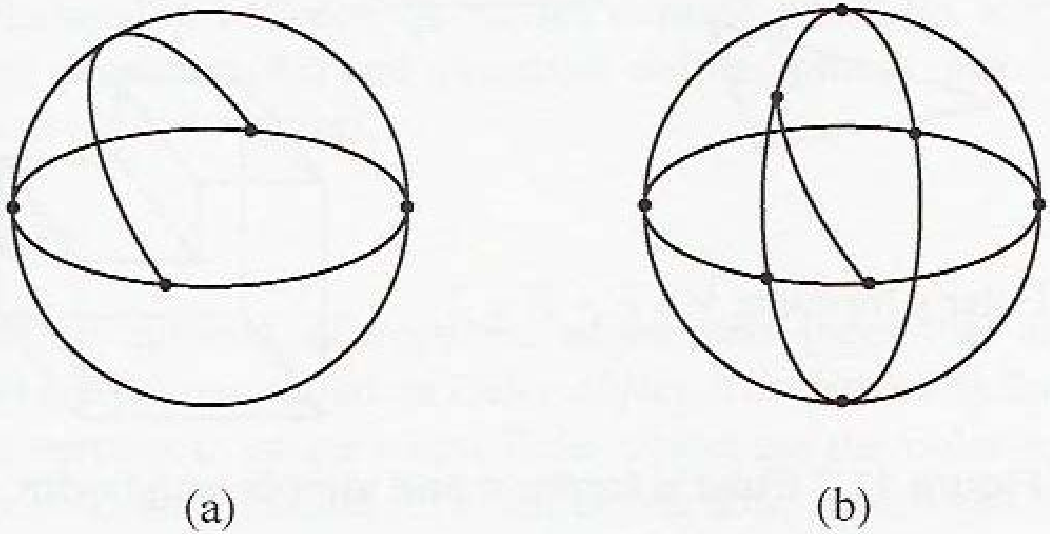
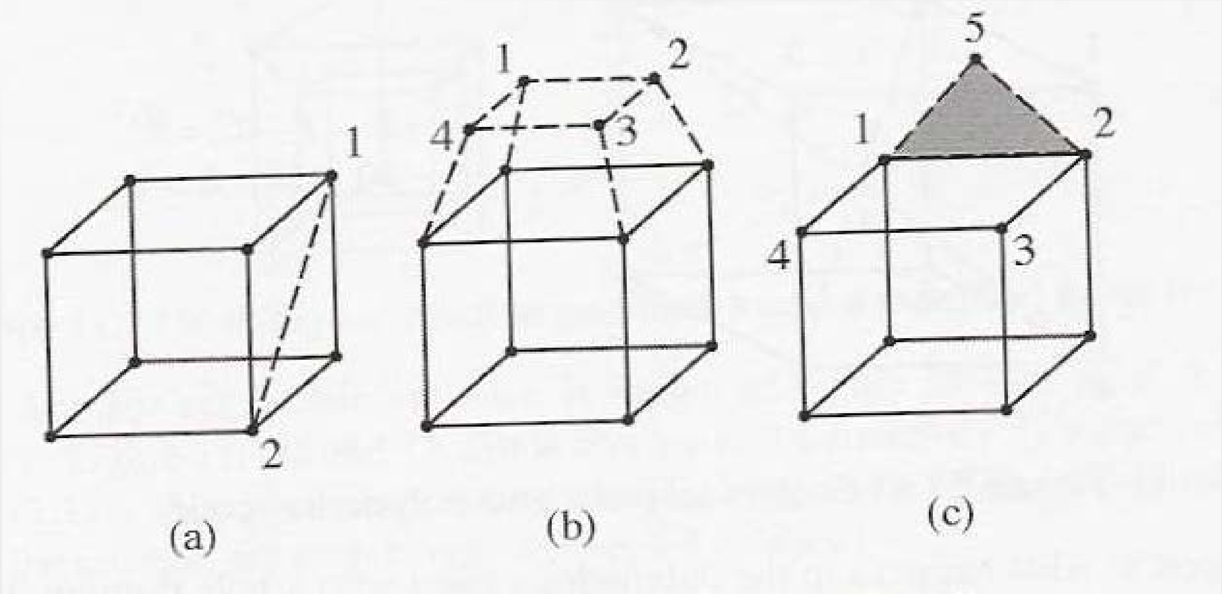


Figure 11.10 Modification of an Euler net on a sphere.

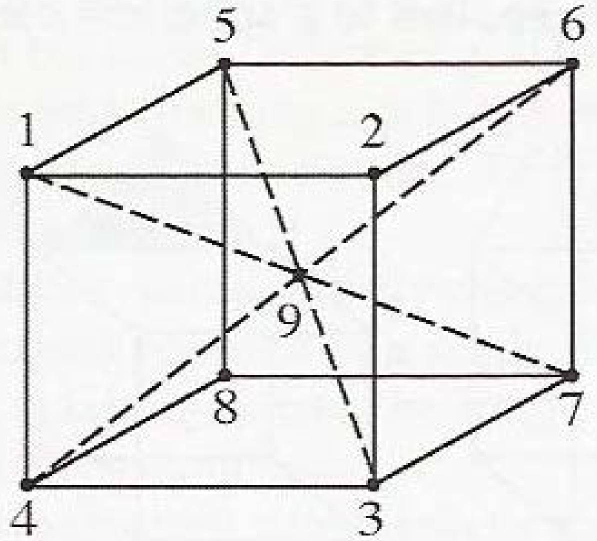
valid modifications to spherical nets

Model Topology (continued)



valid modifications of (a) and (b) invalid modification of (c)

Model Topology (continued)



$$V - E + F - C = 1$$

$$9 - 20 + 18 - 6 = 1$$

Figure 11.11 Euler's formula and polyhedral cells.

C = number of polyhedral cells in 3D

Model Topology (continued)

(a) Object with hole. External faces of hole are inadmissible.

H = # holes in faces

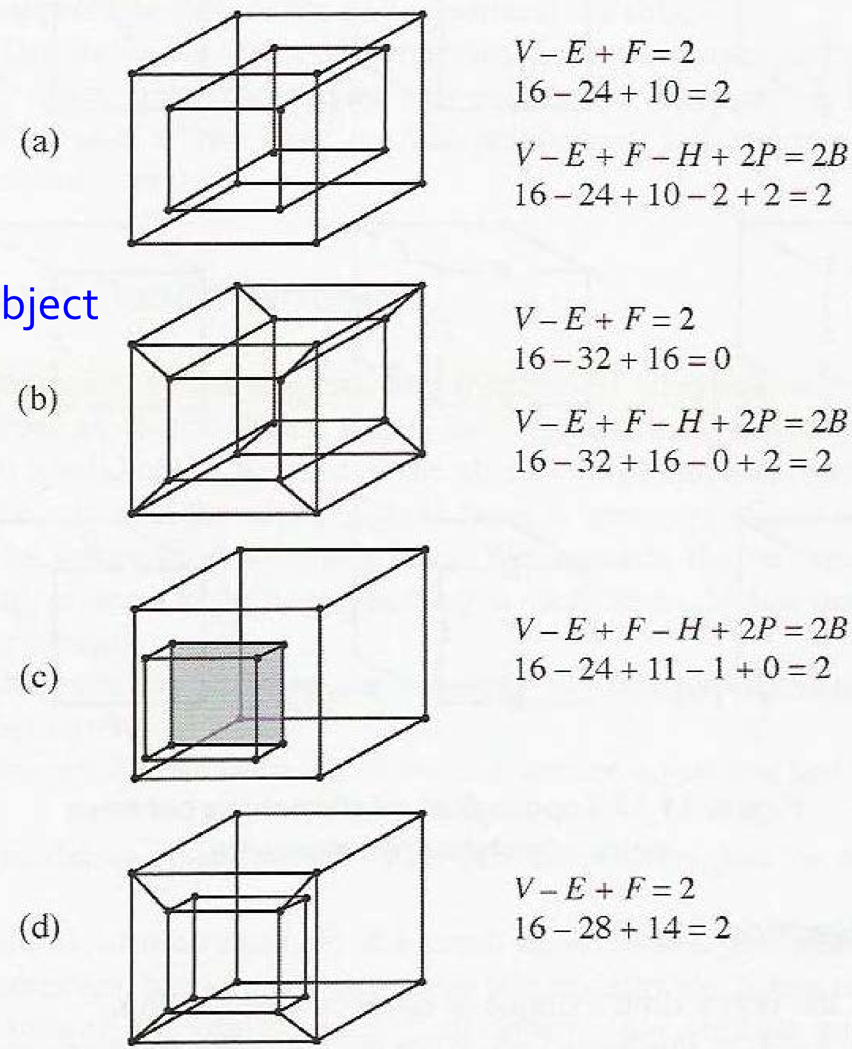
P = # holes entirely through object

B = # separate objects

(b) Edges added to correct inadmissibility.

(c) Acceptable concavity.

(d) Adding edges satisfies original Euler formula.



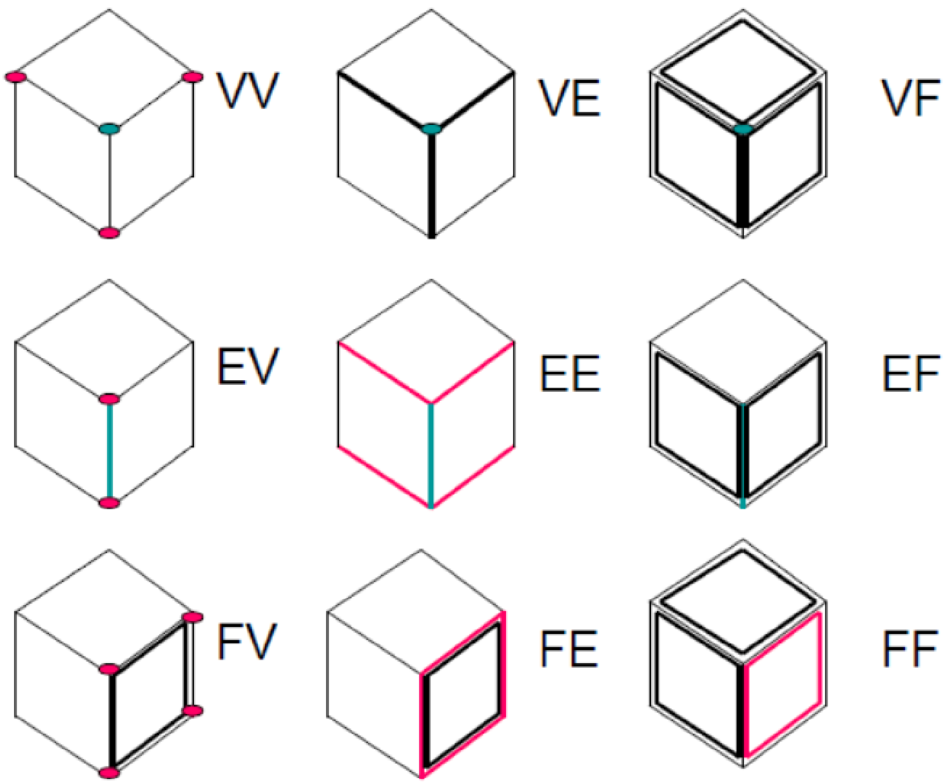
formula modification

Figure 11.12 Multiply-connected polyhedra and a modified Euler formula.

Model Topology (continued)

From \ To	Face	Edge	Vertex
Face	F: {F} 1:N	F: {E} 1:N	F: {V} 1:N
Edge	E: {F} 1:2	E: {E} 1:N	E: {V} 1:2
Vertex	V: {F} 1:N	V: {E} 1:N	V: {V} 1:N

Adjacency Topology in B-rep



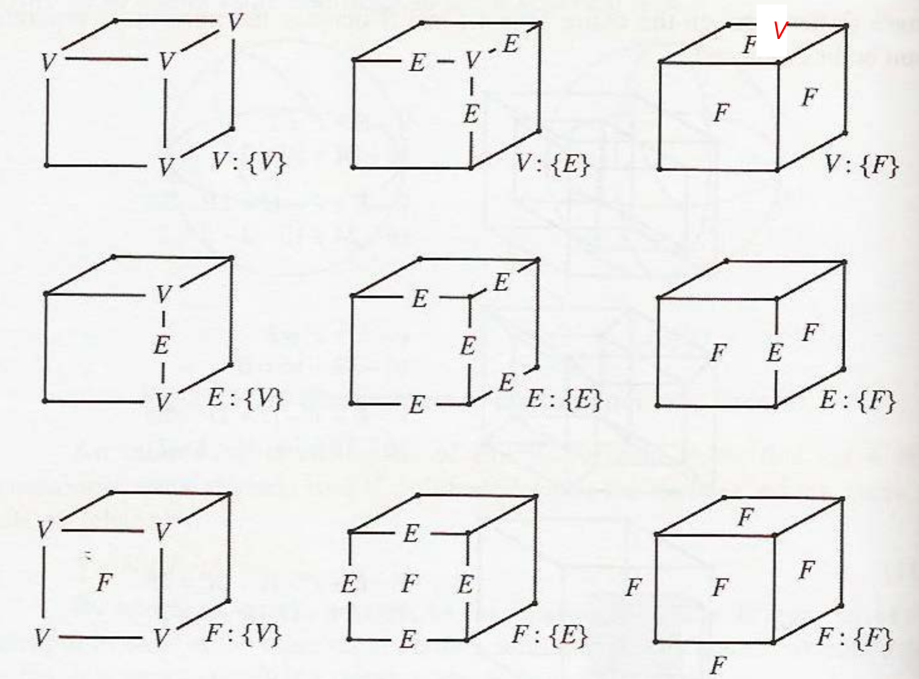
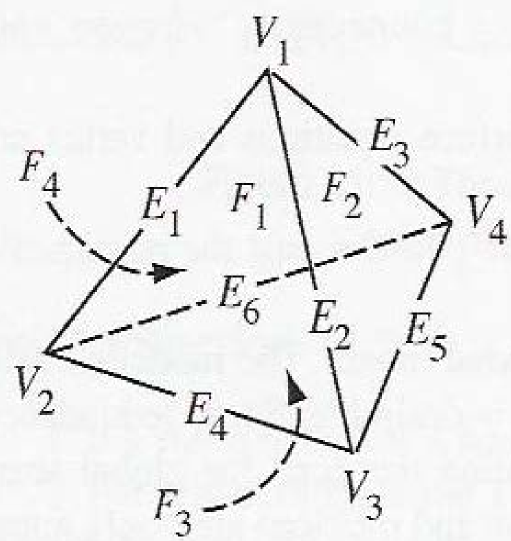


Figure 11.13 Topological relationships between pairs of polyhedron elements.

9 classes of topological relationships between pairs of 3 types of elements

Graph-Based Models

- Geometric model emphasizing topological structure
- Data pointers link object's faces, edges, vertices
- Trade-off: redundancy yields search speed

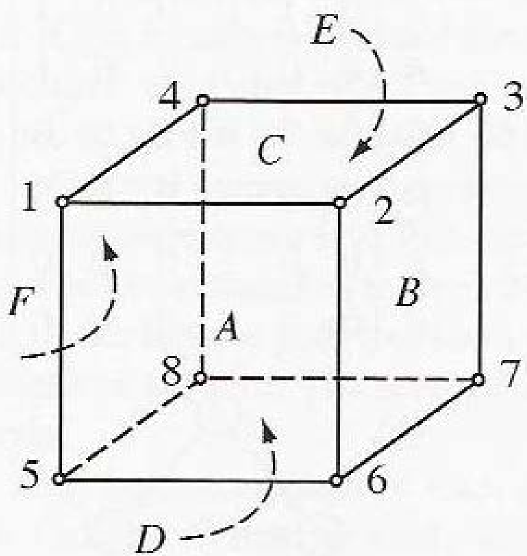


V_1	E_1	F_1
V_2, V_3, V_4	V_1, V_2	V_1, V_2, V_3
E_1, E_2, E_3	E_2, E_3, E_4, E_6	E_1, E_4, E_2
F_1, F_2, F_4	F_1, F_4	F_2, F_3, F_4
V_2	E_2	F_2
V_1, V_3, V_4	V_1, V_3	V_1, V_3, V_4
E_1, E_4, E_6	E_1, E_3, E_4, E_5	E_2, E_5, E_3
F_1, F_3, F_4	F_1, F_2	F_1, F_3, F_4
V_3	E_3	F_3
V_1, V_2, V_4	V_1, V_4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

Figure 11.14 A graph-based model.

Graph-Based Models (continued)

- For planar-faced polyhedra connectivity (adjacency) matrices can be used.



Vertex × Vertex

Vertex	1	2	3	4	5	6	7	8
1	0	1	0	1	1	0	0	0
2	1	0	1	0	0	1	0	0
3	0	1	0	1	0	0	1	0
4	1	0	1	0	0	0	0	1
5	1	0	0	0	0	1	0	1
6	0	1	0	0	1	0	1	0
7	0	0	1	0	0	1	0	1
8	0	0	0	1	1	0	1	0

Face × Face

Face	A	B	C	D	E	F
A	0	1	1	1	0	1
B	1	0	1	1	1	0
C	1	1	0	0	1	1
D	1	1	0	0	1	1
E	0	1	1	1	0	1
F	1	0	1	1	1	0

Figure 11.15 Connectivity matrices for a polyhedron.

Graph-Based Models (continued)

This is the connectivity matrix for Figure 11.16b:

$$
\begin{array}{c c}
 & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\end{array}
$$

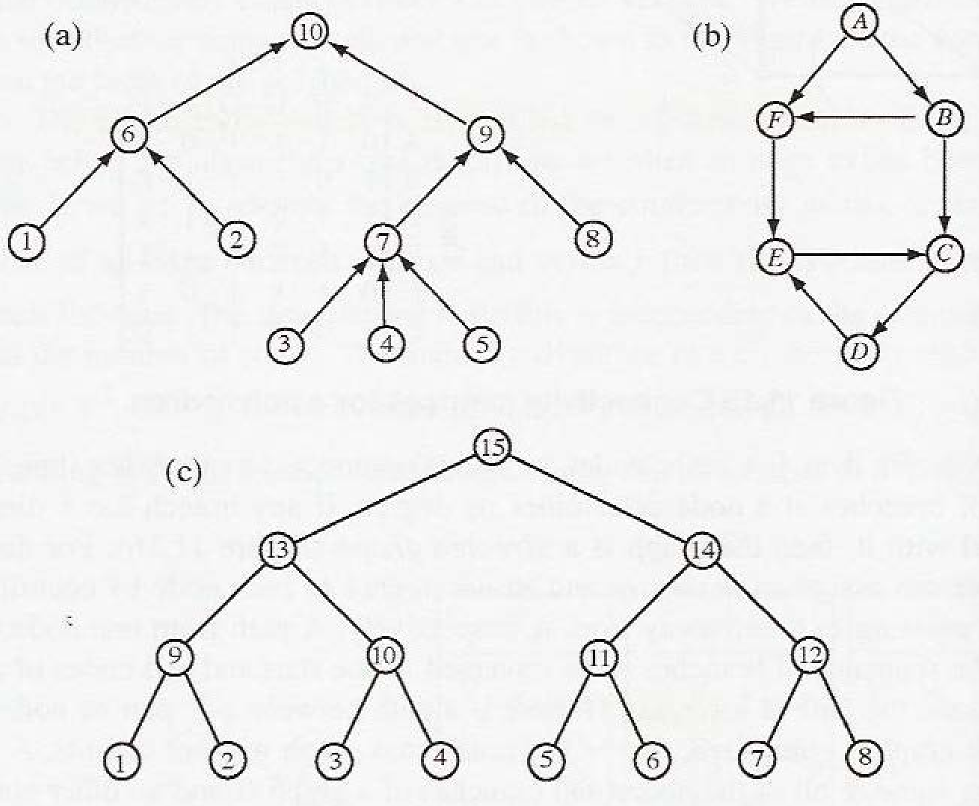


Figure 11.16 Examples of graphs.

Boolean Models

Table 11.1 Properties of Operations on Sets

Union Properties:

1. $A \cup B$ is a set.	Closure property
2. $A \cup B = B \cup A$	Commutative property
3. $(A \cup B) \cup C = A \cup (B \cup C)$	Associative property
4. $A \cup \varnothing = A$	Identity property
5. $A \cup A = A$	Idempotent property
6. $A \cup cA = E$	Complement property

Intersection Properties

1. $A \cap B$ is a set.	Closure property
2. $A \cap B = B \cap A$	Commutative property
3. $(A \cap B) \cap C = A \cap (B \cap C)$	Associative property
4. $A \cap E = A$	Identity property
5. $A \cap A = A$	Idempotent property
6. $A \cap cA = \varnothing$	Complement property

Distributive Properties

1. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	Union is distributive over intersec- ion
2. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	Intersection is distributive over un- on

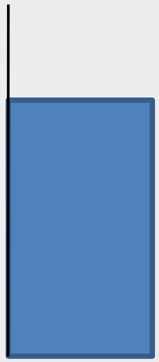
Complementation Properties

1. $cE = \varnothing$	The complement of the universal set is the empty set.
2. $c\varnothing = E$	The complement of the empty set is the universal set.
3. $c(cA) = A$	The complement of a complement of a set A is A.
4. $c(A \cup B) = cA \cap cB$	DeMorgan's law.
5. $c(A \cap B) = cA \cup cB$	DeMorgan's law.

Boolean Models (continued)

Set Membership Classification

$$X = bX \cup iX$$



- Goal: define regularized set
  - closure of interior
  - no “dangling edges” or disconnected lower-dimensional parts
- Set membership classification differentiates between 3 subsets of any regularized set *X*:
  - *bX*: boundary of *X*
  - *iX*: interior of *X*
  - *cX*: complement of *X*

Boolean Models (continued)
Set Membership Classification

- Some similar geometric modeling problems:
  - Point inclusion: point inside or outside a solid?
  - Line/polygon clipping: line segment vs. polygon
  - Polygon intersection: 2 polygons
  - Solid interference: 2 solids

Boolean Models (continued)

Set Membership Classification

(a) 2 points same or different?

(b) point vs. curve: 3 cases

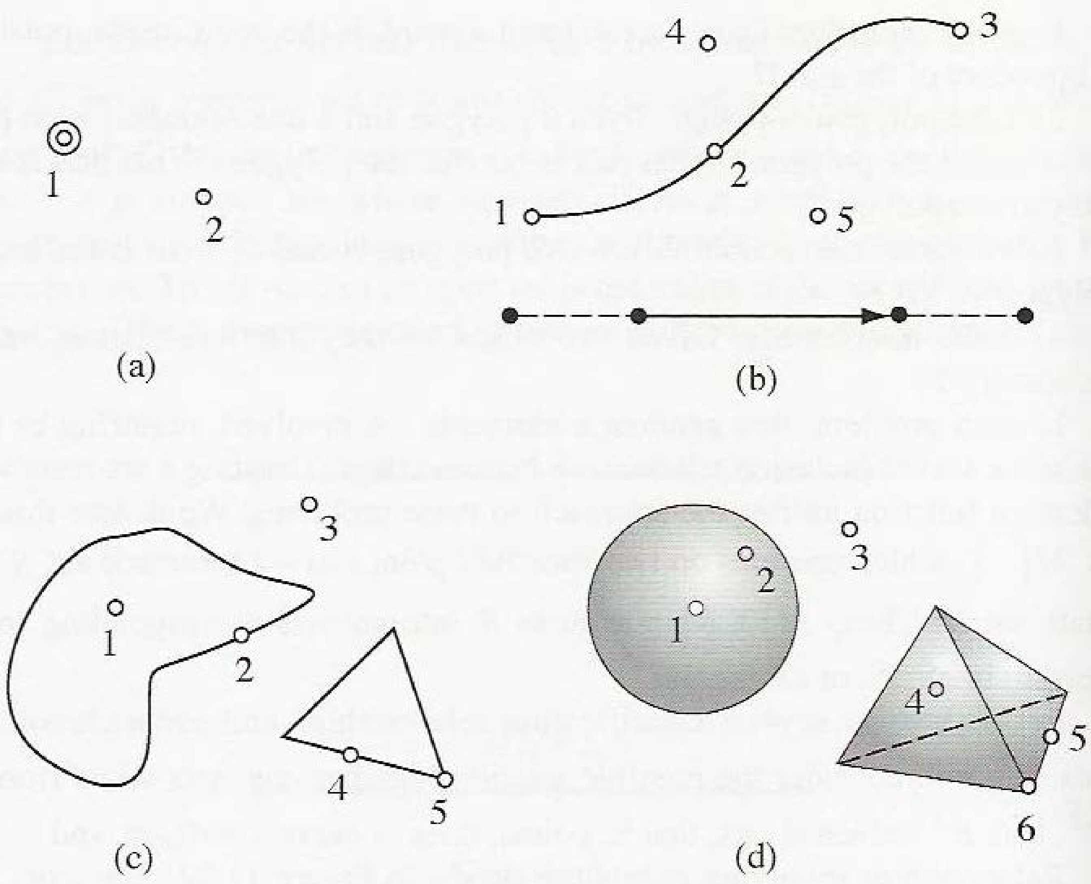


Figure 11.22 Point classification.

(c) point vs. curve or polygon

(d) point vs. curved or polyhedral object

Boolean Models (continued)

Set Membership Classification

applicable to topological disc

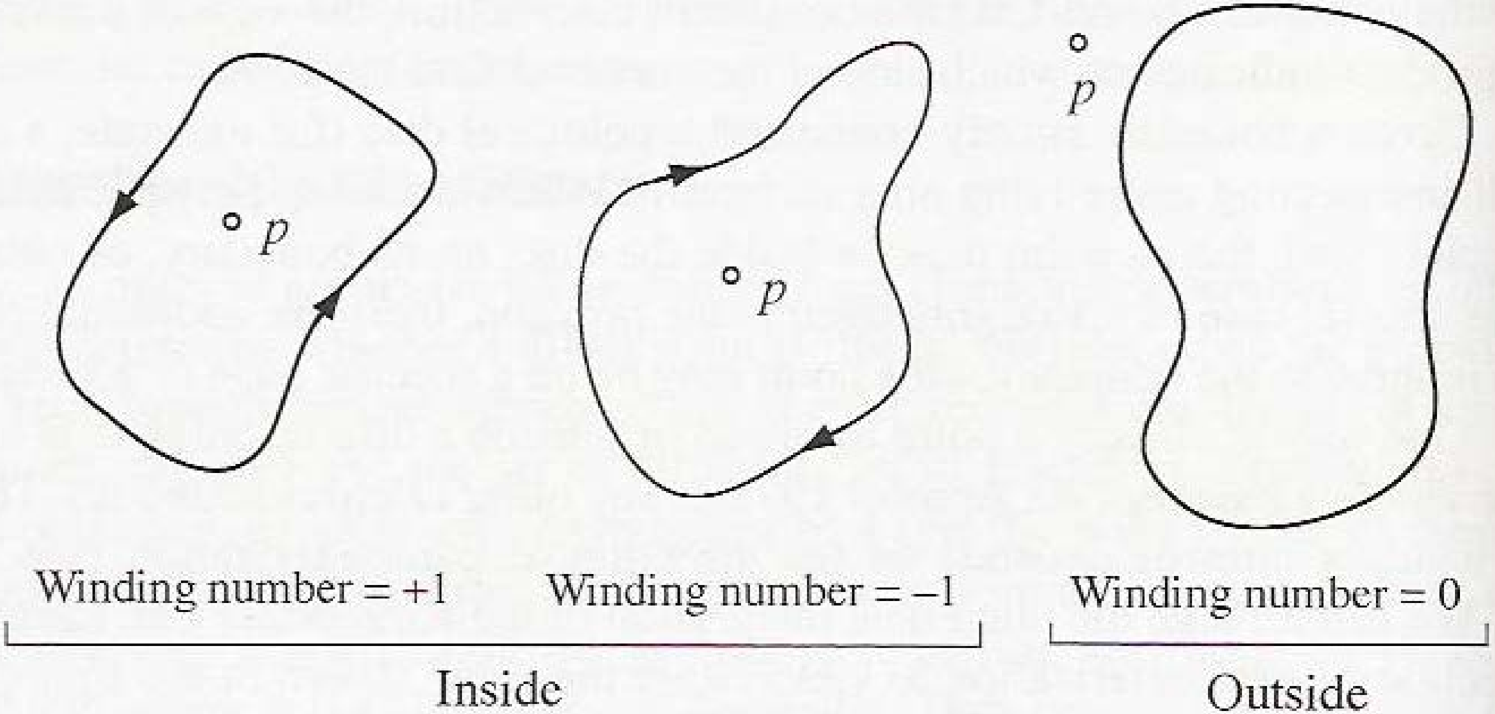


Figure 11.23 The winding number and the inside-outside classification.

Boolean Models (continued)

Set Membership Classification

(a) point vs. sphere as parametric surface (assumes knowledge of closest point *q*)

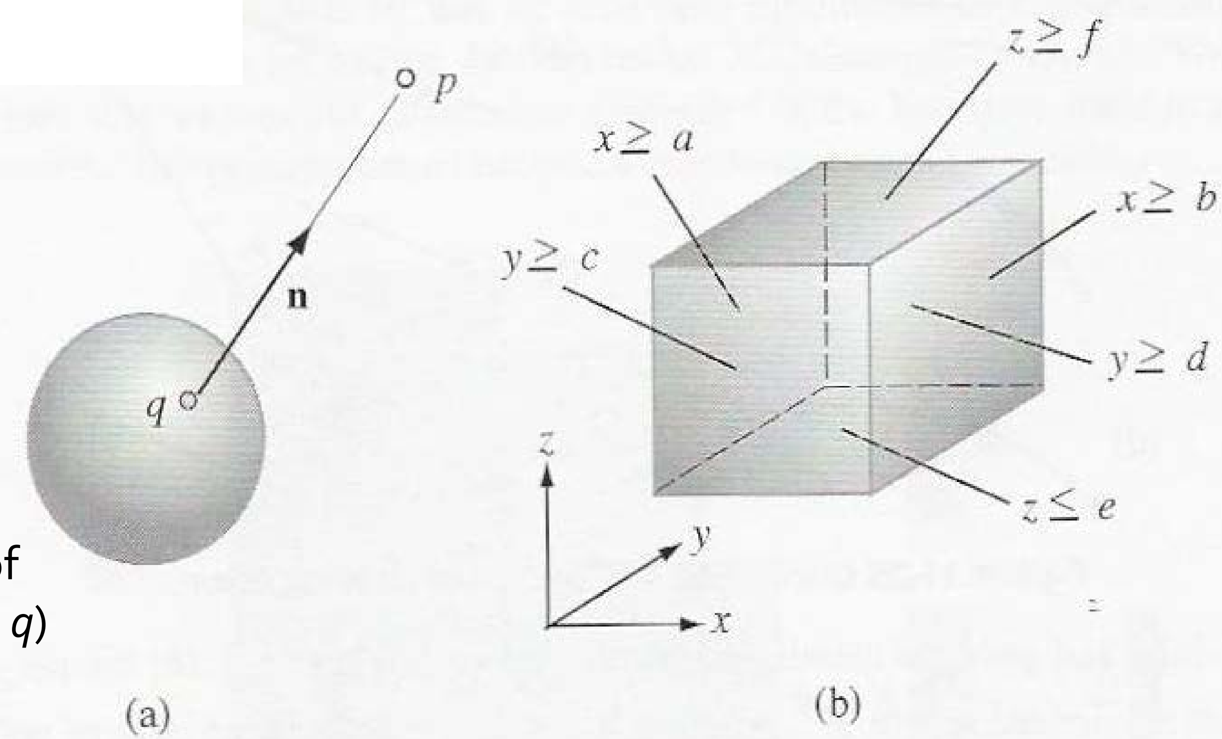


Figure 11.24 Inside and outside a solid.

(b) point vs. parallelepiped defined as Boolean intersection of half-spaces

Boolean Models (continued)

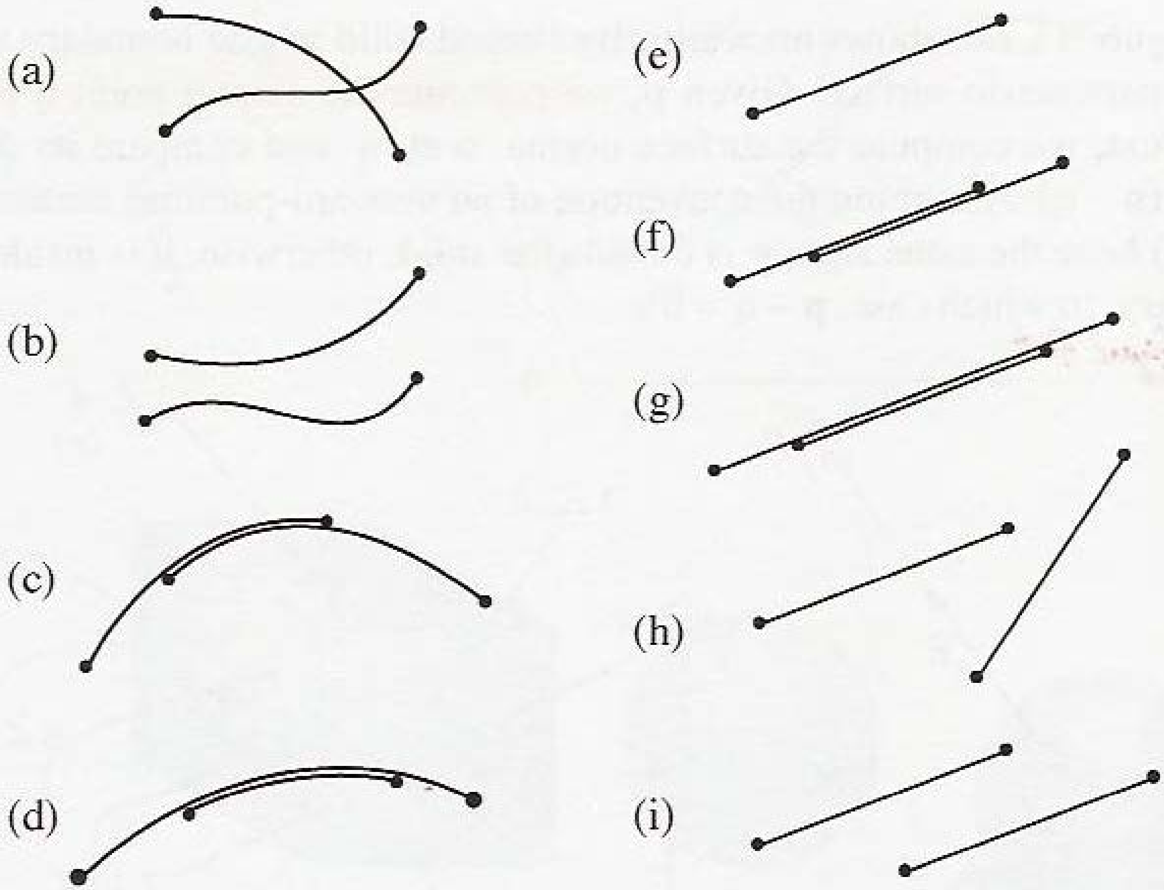
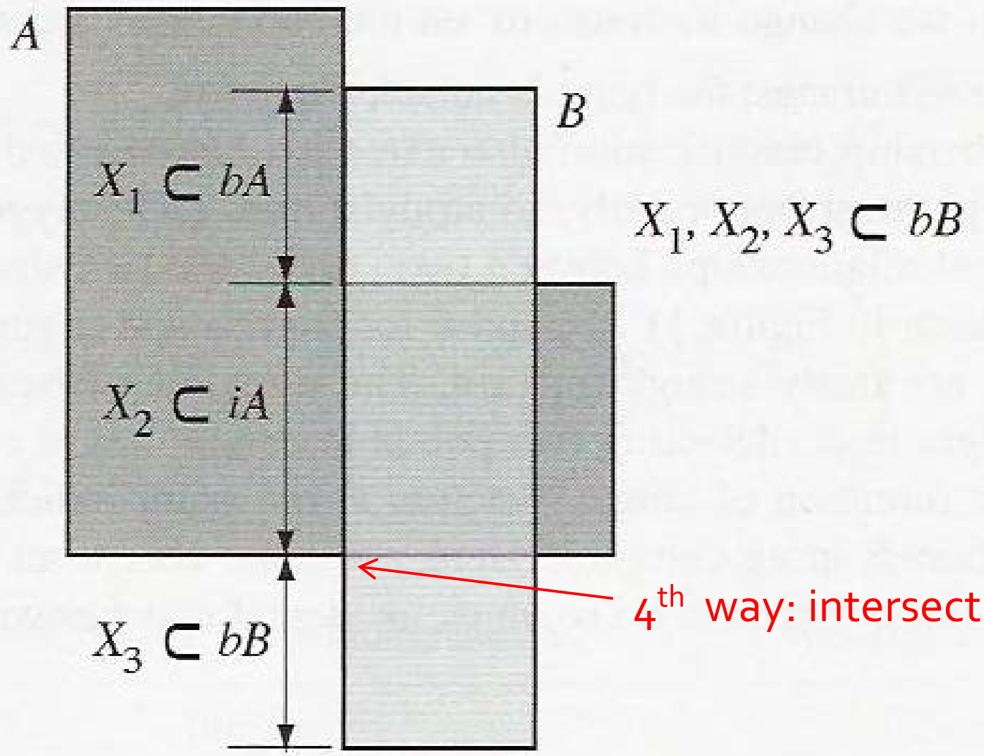


Figure 11.25 Curve and line segment classification.

Boolean Models (continued)

edge of *B* intersects *A* in 4 ways



4^{th} way: intersection is a point

Figure 11.26 Line and polygon classifications.

2 regularized polygons *A* and *B*

Boolean Models (continued)

Point 2 is problematic with respect to intersection of *A* and *B*.

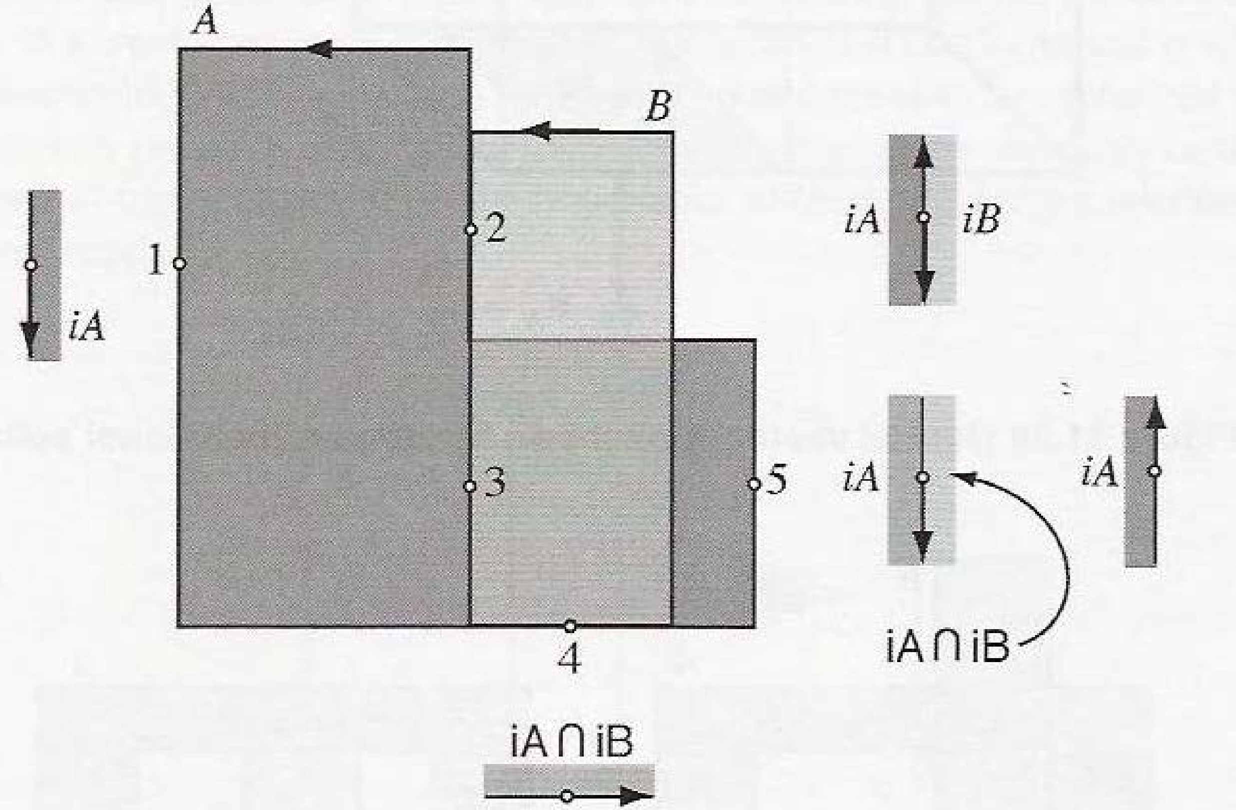


Figure 11.27 Tangent vector convention for two-dimensional objects.

Boolean Models (continued)

Outward pointing normals can aid intersection of 3D solids *A* and *B*.

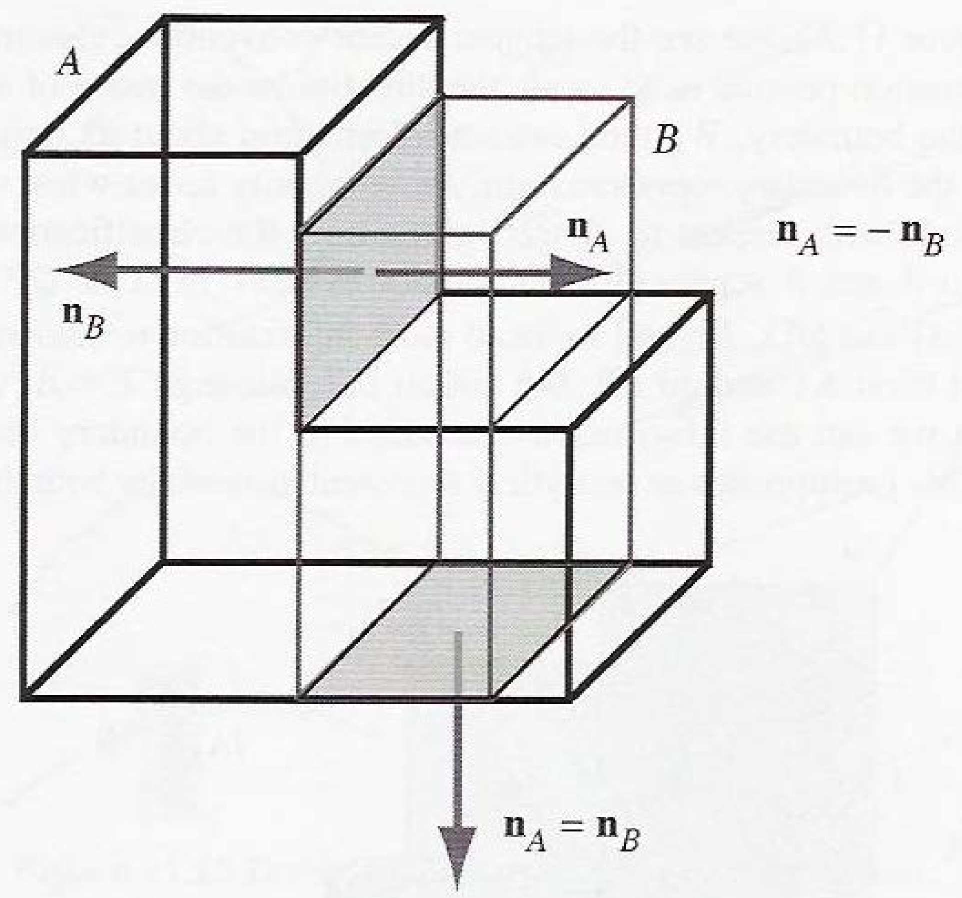


Figure 11.28 Normal vector convention for three-dimensional solids.

Boolean Models (continued)

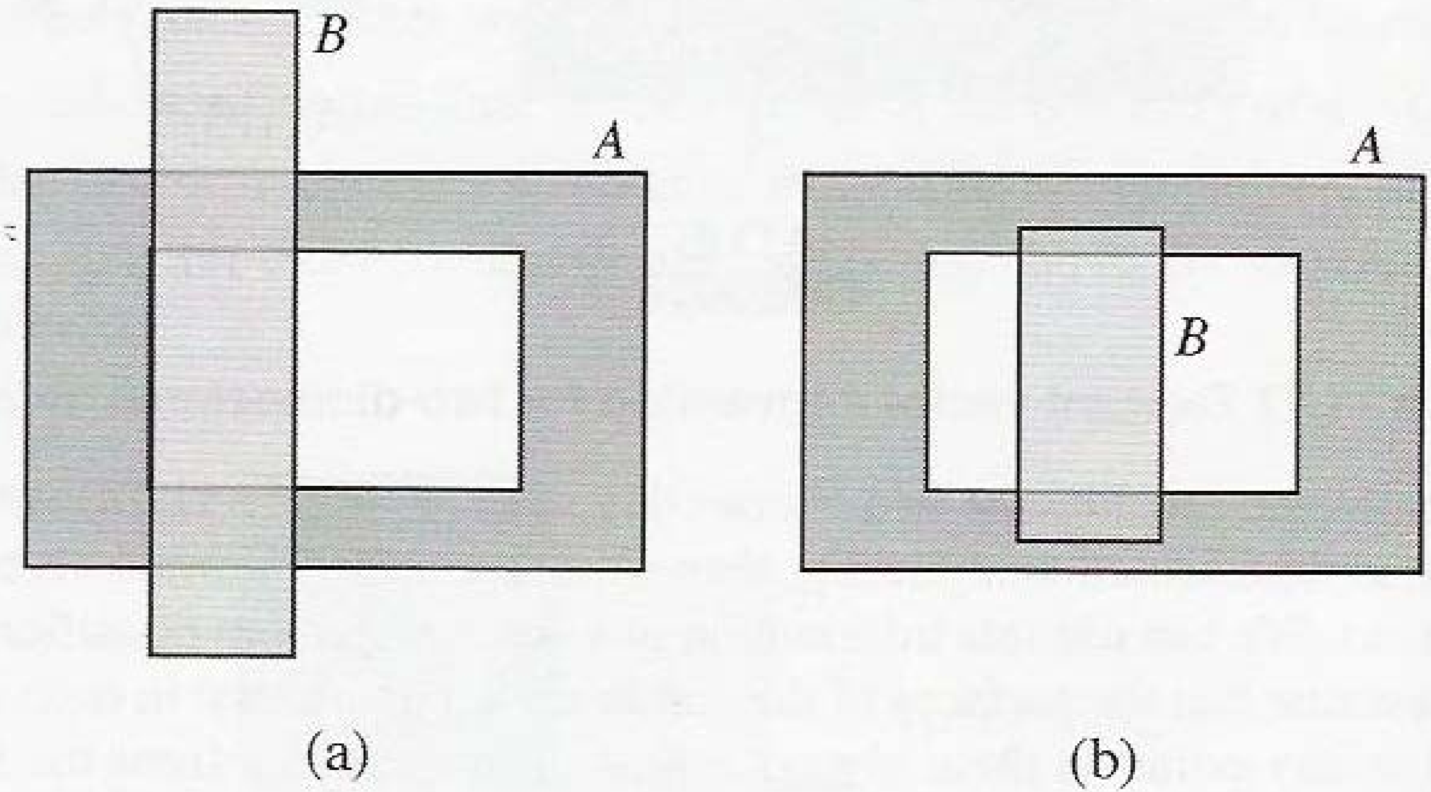
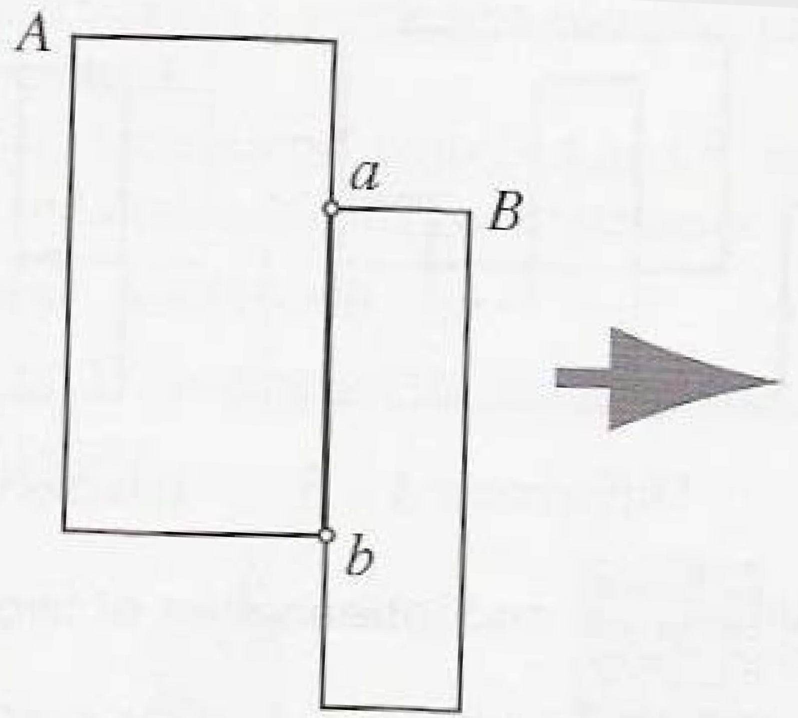


Figure 11.29 Problems for set-membership classification.

Boolean Models (continued)



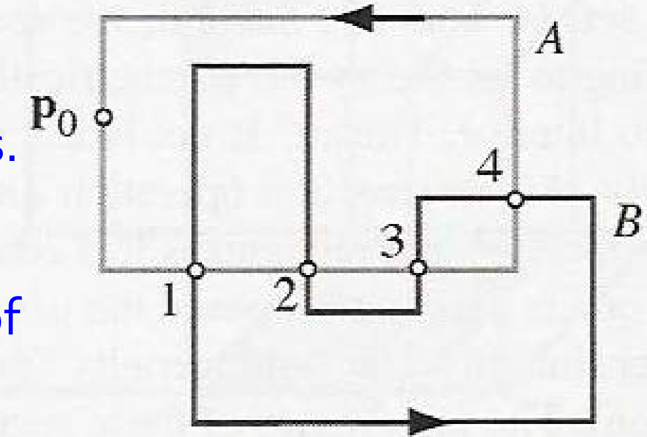
Degenerate intersection of 2 well-defined 2D objects.

Boolean Models (continued)

Find intersection points.
Segment intersected edges.

For **Union**:

- Find point on boundary of A outside B.
- Trace around loop of edges.
- Trace additional loops if needed.



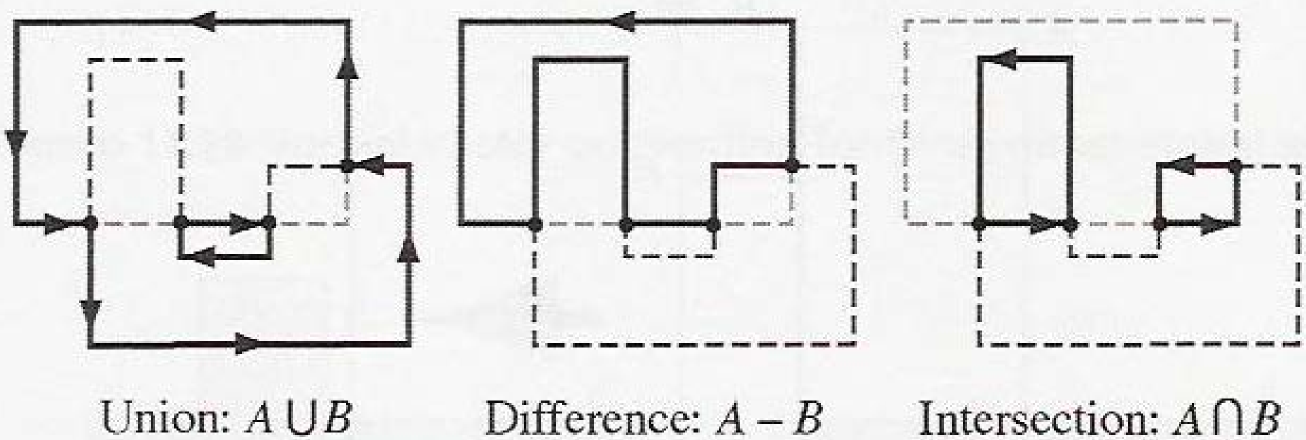
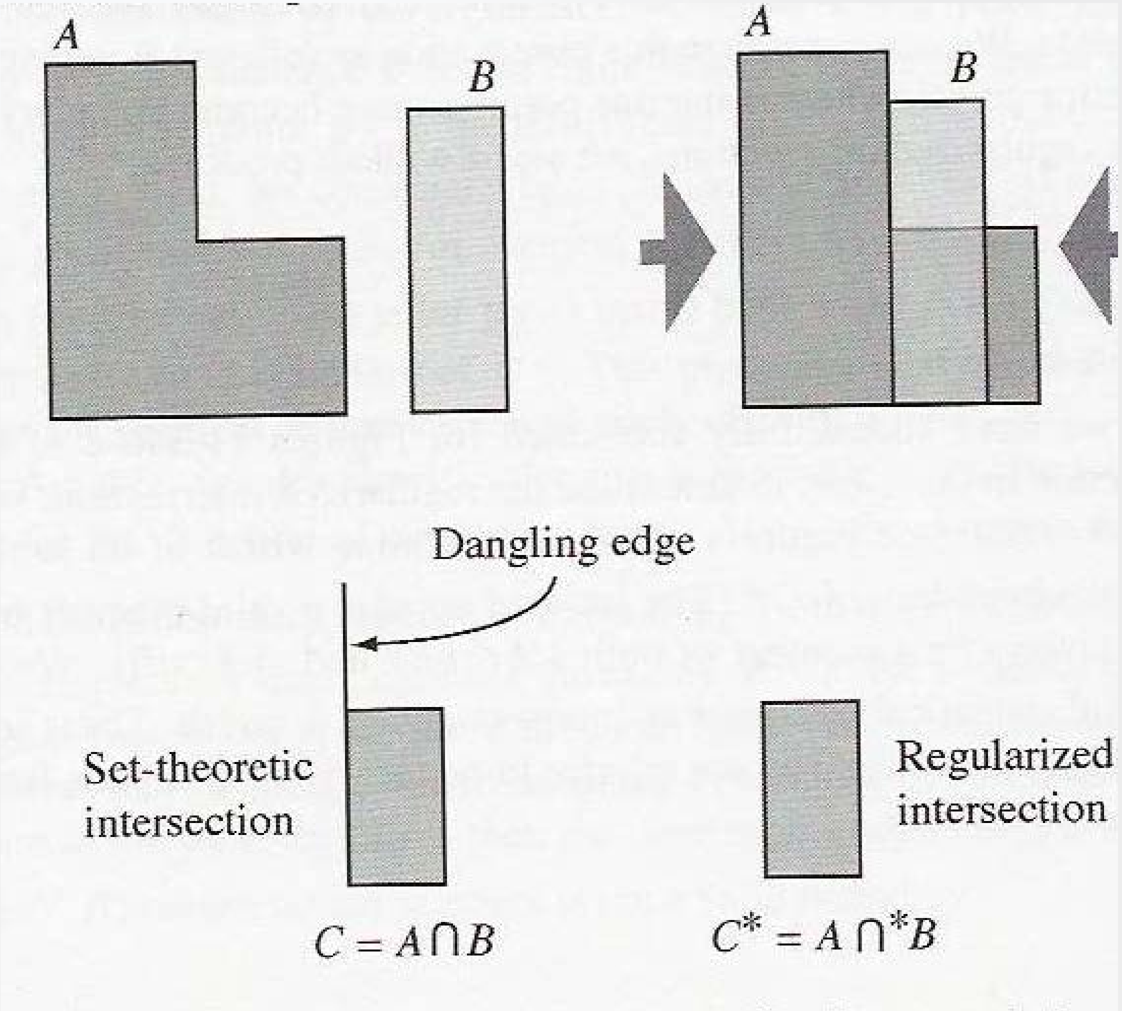


Figure 11.31 Union, difference, and intersection of two simple polygons.

Boolean Models (continued)
Intersection



Set-theoretic and regularized Boolean intersections.

Boolean Models (continued)

Intersection

Need to distinguish between segments 1 & 2 (see next slide).

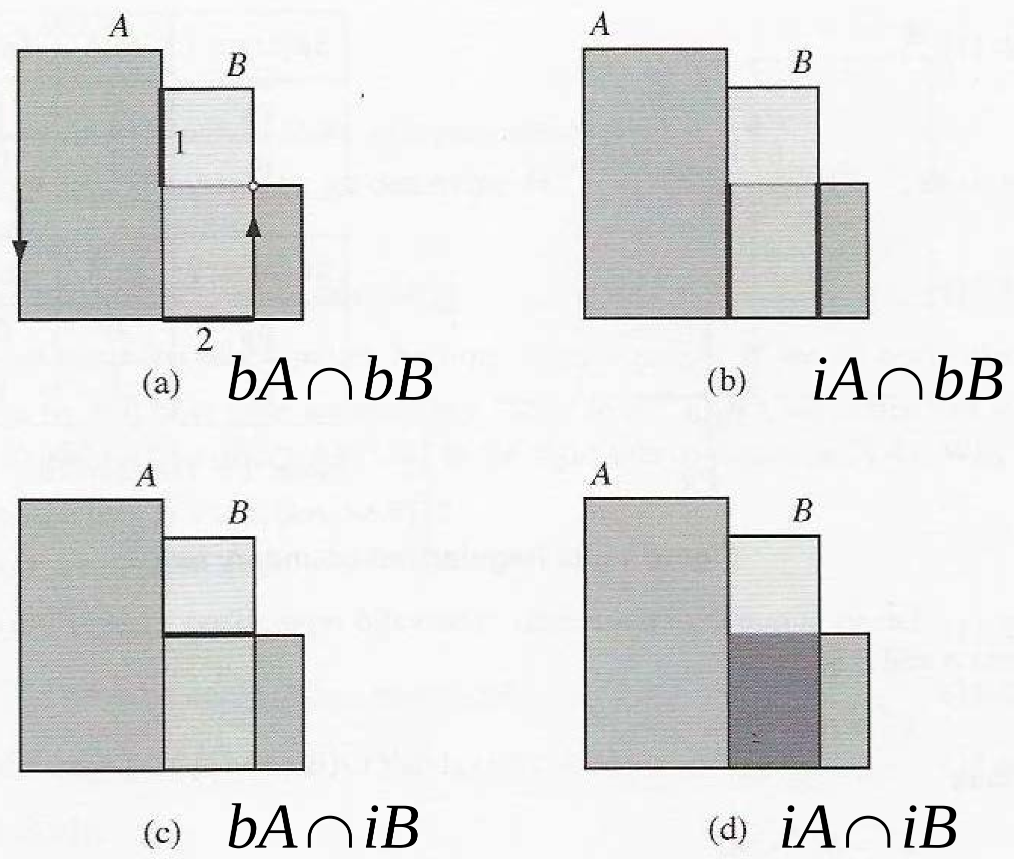


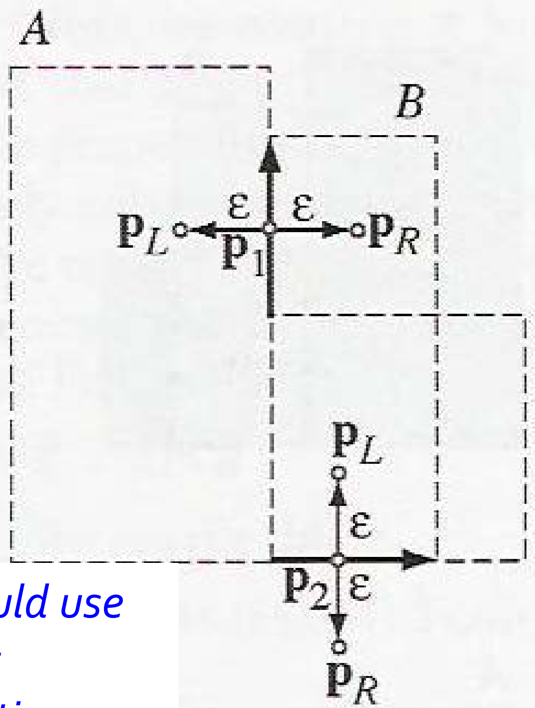
Figure 11.33 Candidate components of a regularized Boolean intersection.

Boundary points can become interior points. Interior points cannot become boundary points.

$$C = (bA \cap bB) \cup (iA \cap bB) \cup (bA \cap iB) \cup (iA \cap iB)$$

Boolean Models (continued)

Intersection



Segment 1	In A	In B
$\mathbf{p}_R$	0	1
$\mathbf{p}_L$	1	0

Segment 2	In A	In B
$\mathbf{p}_R$	0	0
$\mathbf{p}_L$	1	1

Note: 1 = Yes, 2 = No

Figure 11.34 Regularized boundary test.

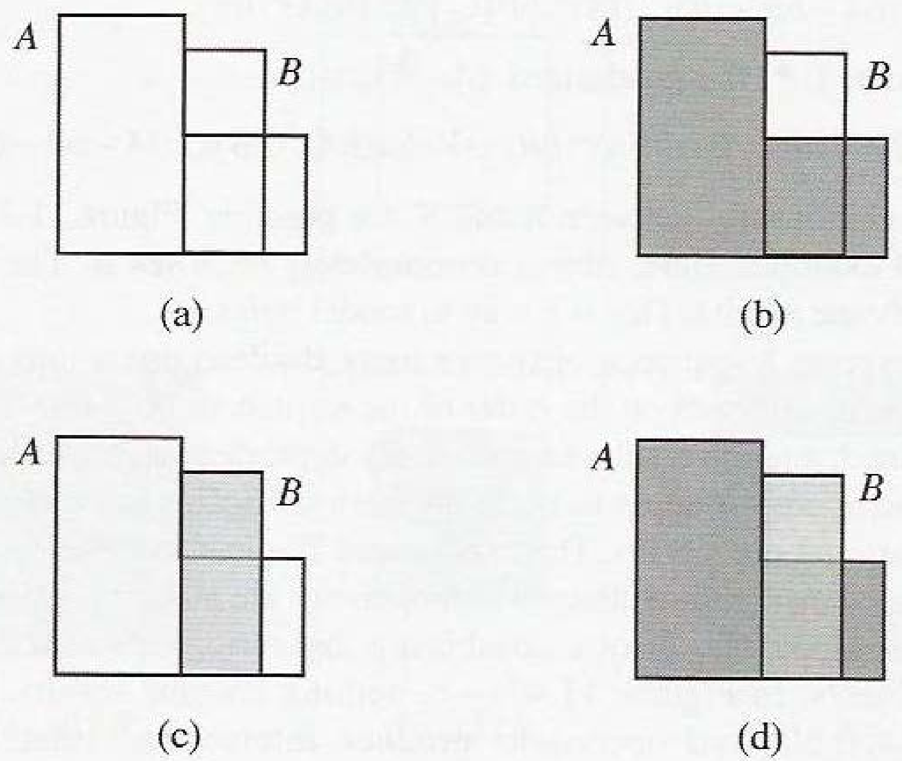
Simpler test would use consistent parameterization directions and tangent vector directions.

Summarizing overall intersection approach…

$$C^* = bC^* \cup iC^* = Valid_b(bA \cap bB) \cup (iA \cap bB) \cup (bA \cap iB) \cup (iA \cap iB)$$

Boolean Models (continued)

Union



Candidate components of a regularized Boolean union.

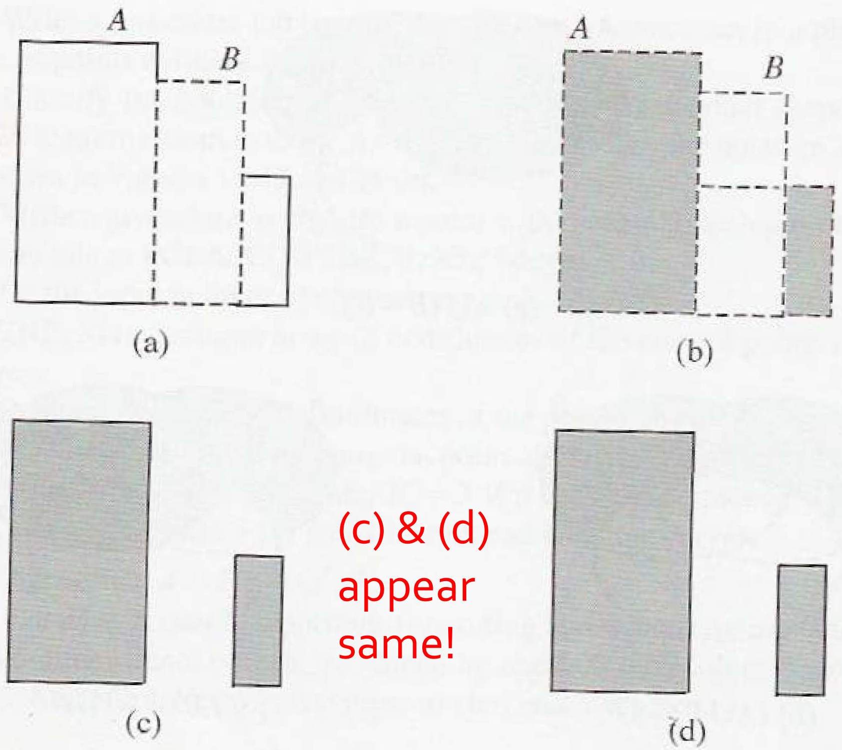
$$C = (bA \cup bB) \cup (iA \cup bB) \cup (bA \cup iB) \cup (iA \cup iB) = bA \cup bB \cup iA \cup iB$$

$$iC^* = iA \cup iB \cup [Valid_i(bA \cap bB)]$$

$$bC^* = bA \cup bB - [(bA \cap iB) \cup (bB \cap iA) \cup Valid_b(bA \cap bB)]$$

Boolean Models (continued)

Difference



(c) & (d) appear same!

Candidate components of a regularized Boolean difference.

$$C^* = (bA - bB - iB) \cup (iA \cap bB) \cup Valid(bA \cap bB) \cup (iA - bB - iB)$$

Boolean Models (continued)

A encloses B.

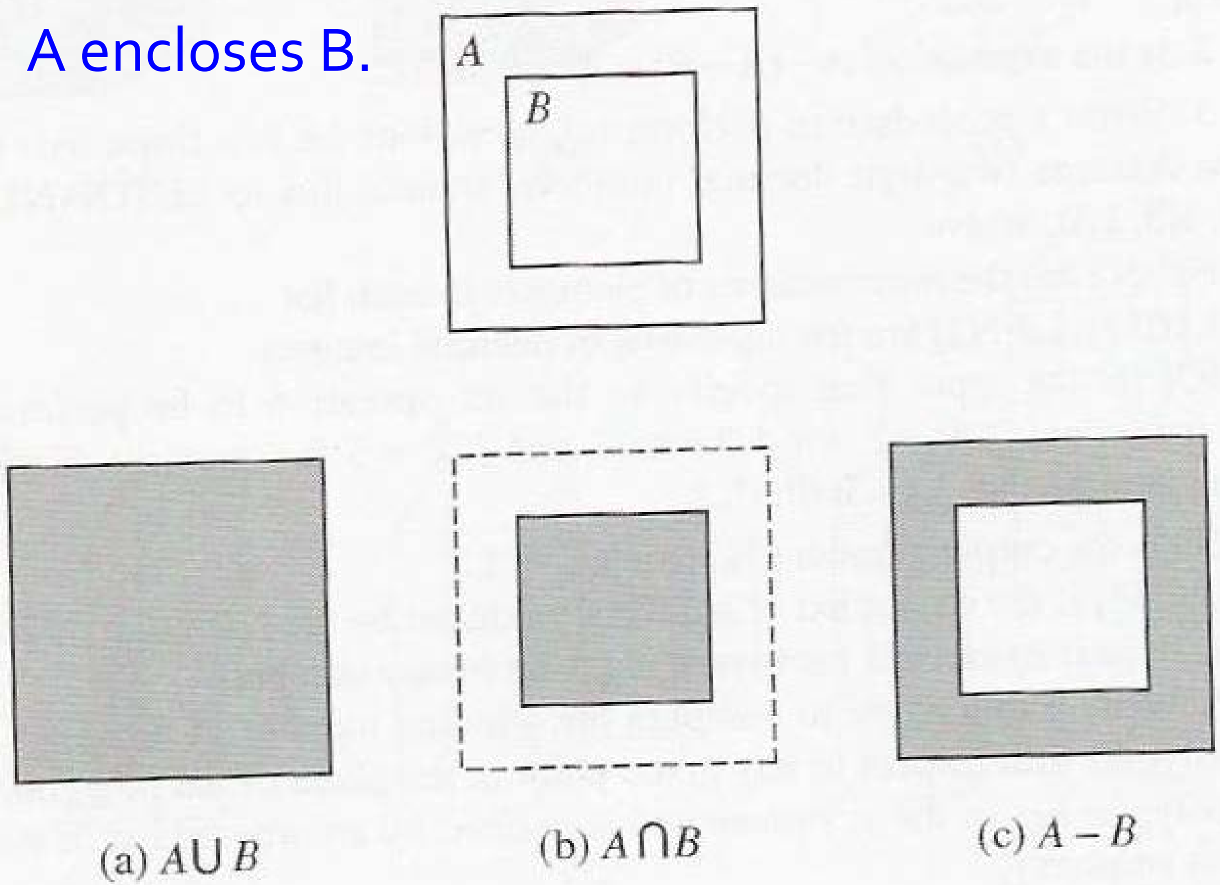


Figure 11.37 Examples of Boolean operations.

Useful for modeling holes.

Boolean Models (continued)

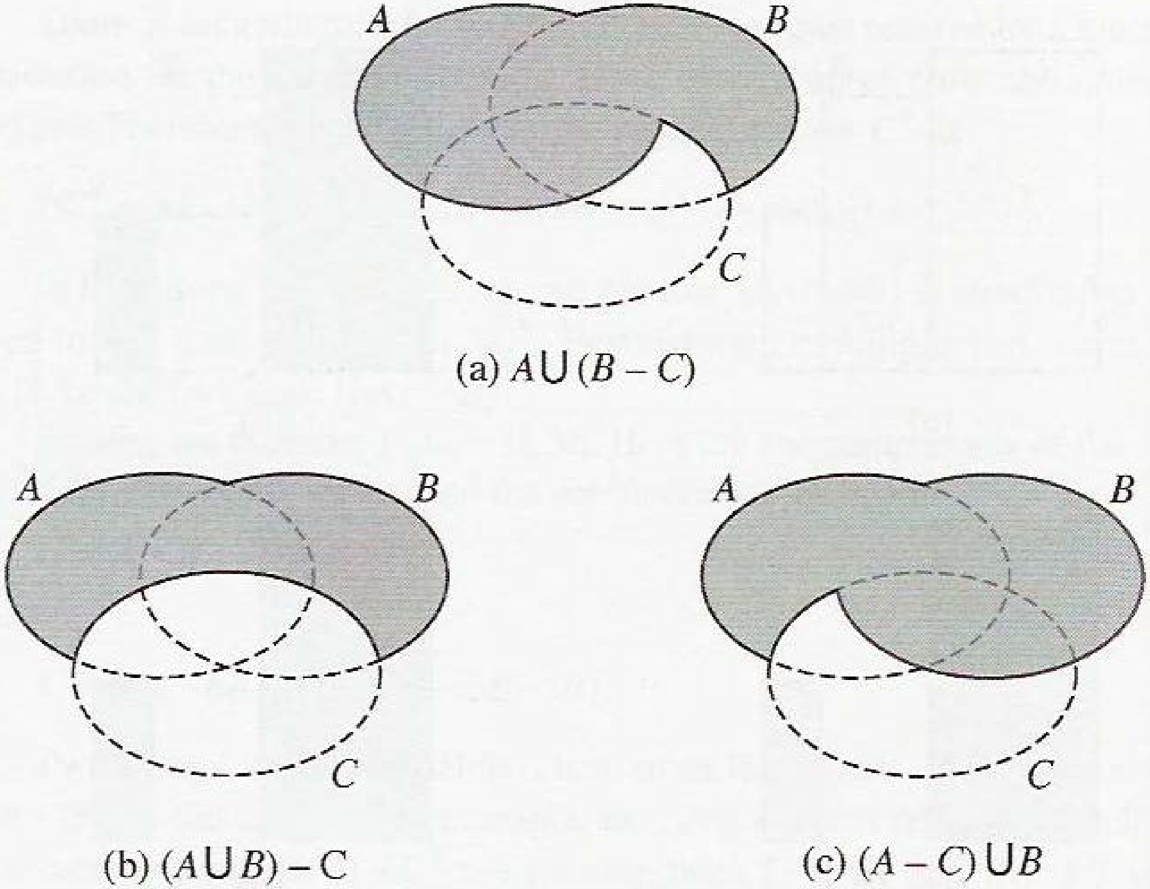
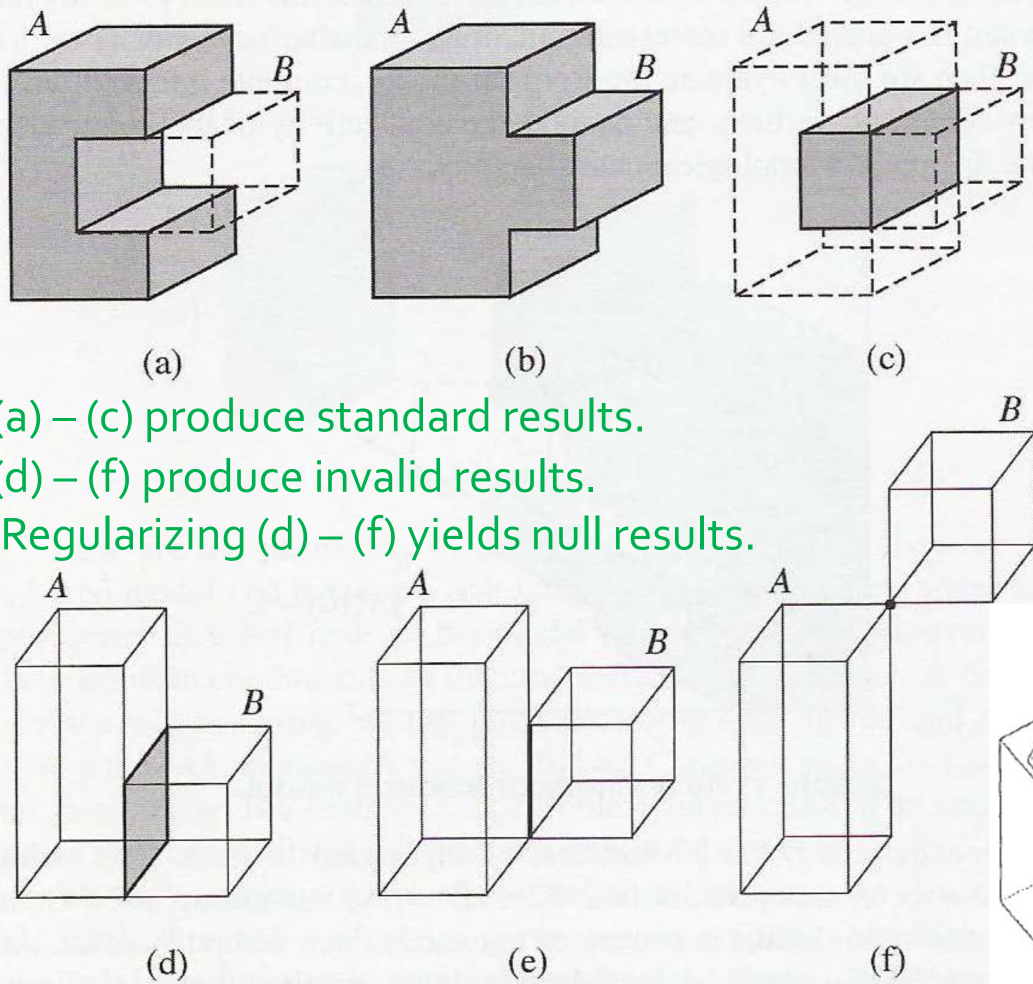


Figure 11.38 Order dependence on Boolean operations.

Boolean Models (continued)



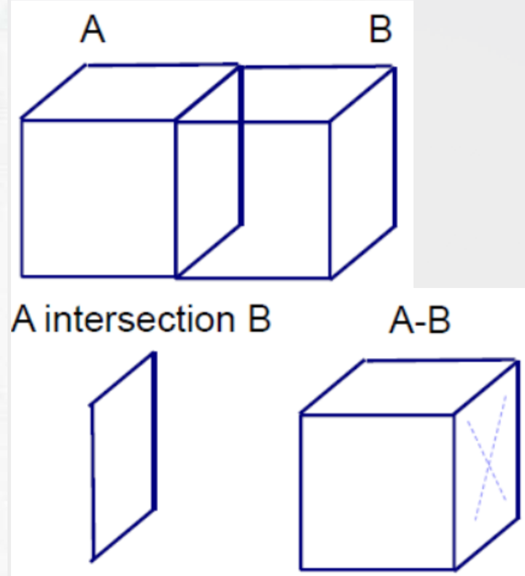
(a) – (c) produce standard results.

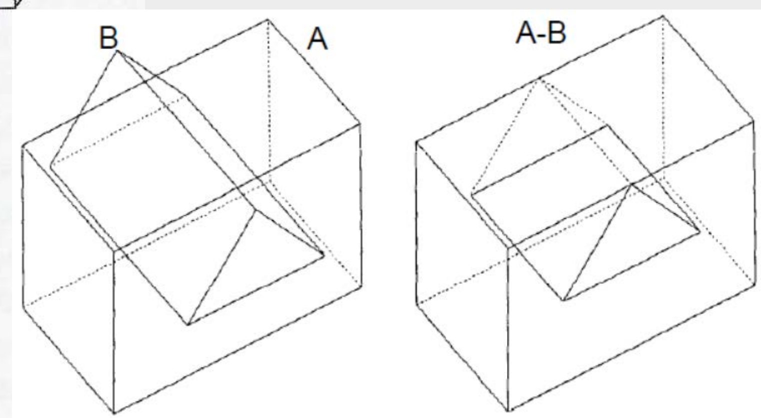
(d) – (f) produce invalid results.

Regularizing (d) – (f) yields null results.

Figure 11.39 Boolean operations on a three-dimensional solid.

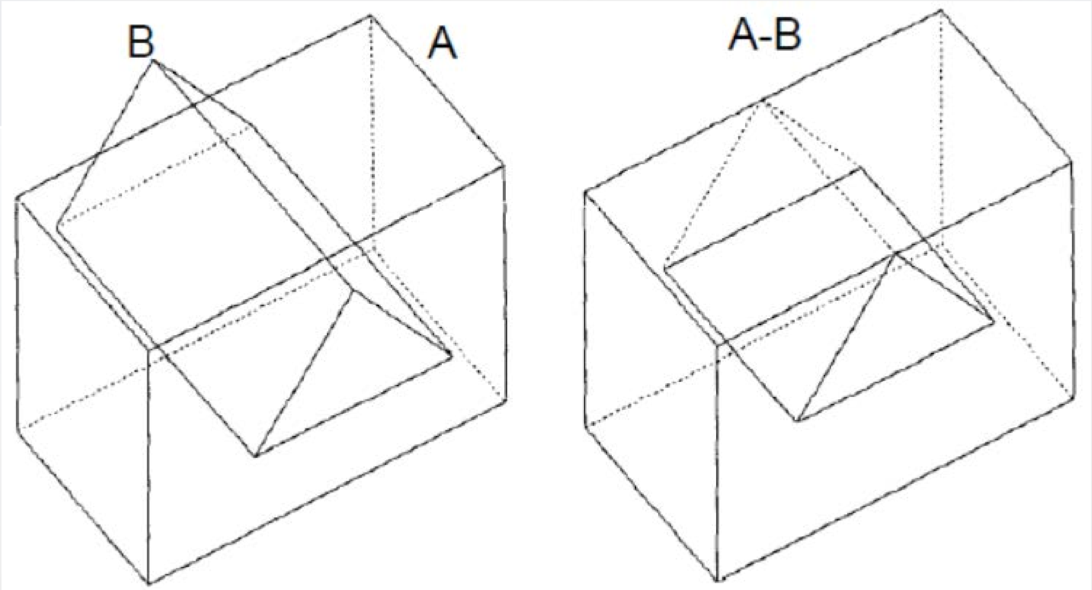
Coincidences problem



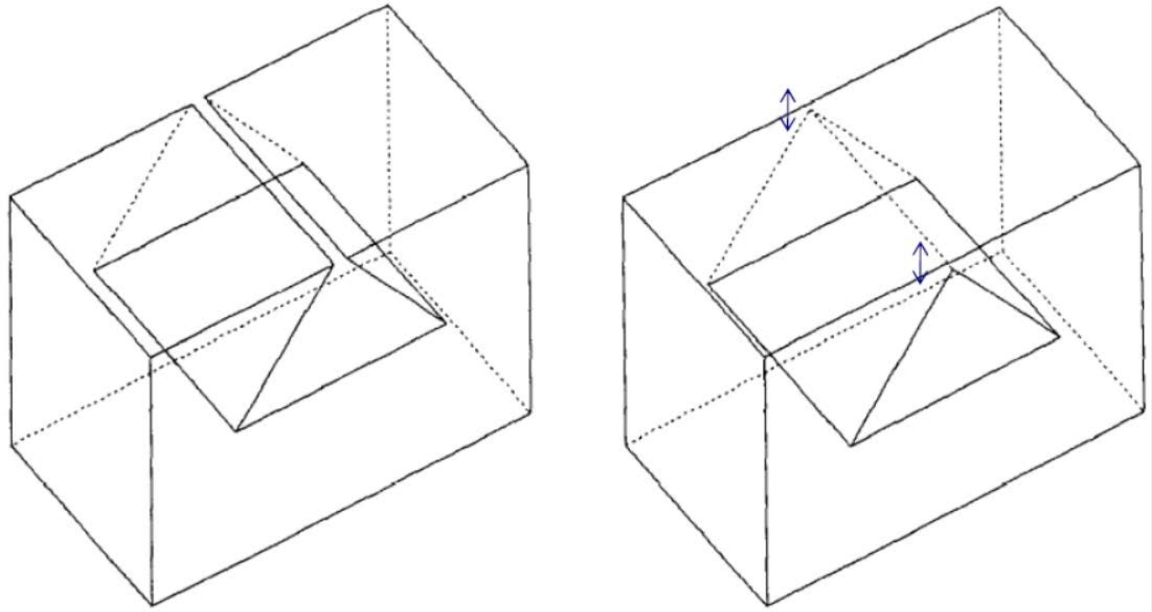


Boolean Models

Coincidences problem

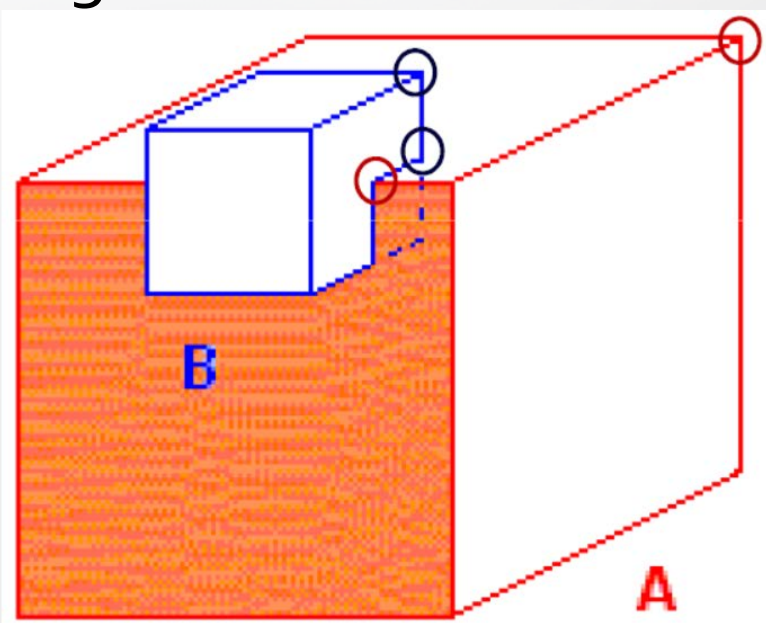


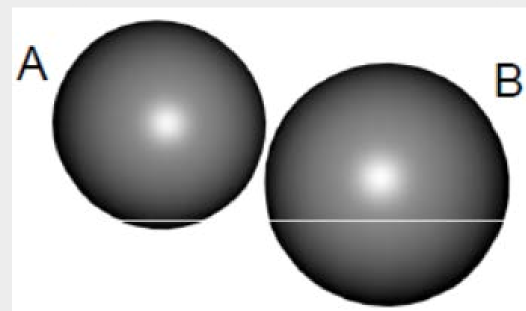
Pseudo manifolds

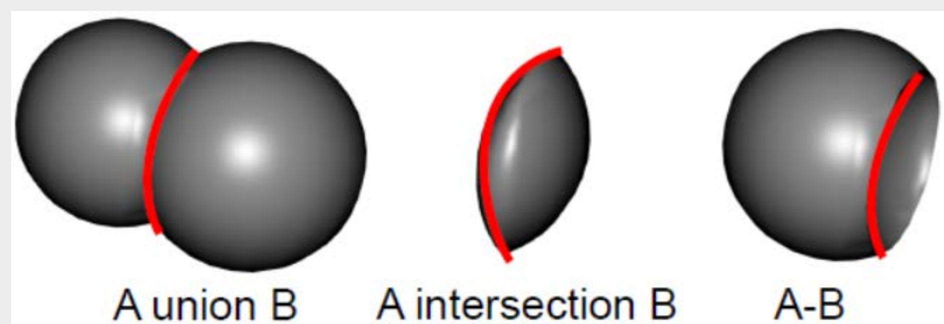


Algorithms for Boolean operations

Based on face classification (Algorithm 1)
Based on vertex classification (Algorithm 2)
Algorithm 2

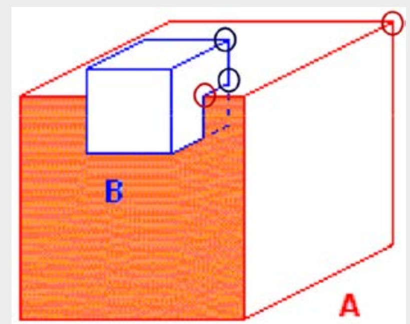






Algorithm 2 (vertex classification)

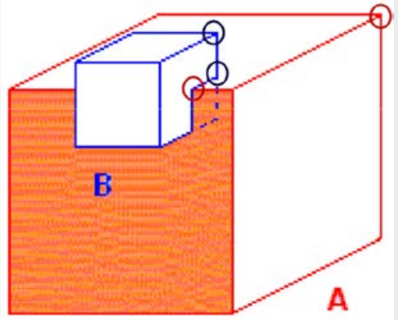
Algorithm Boolean Op (vertex classification)
// 1. Classify existing vertices
addVertices(A, B, LV); // add to LV vertices from A classified wrt B
addVertices(B, A, LV); // add to LV vertices from B classified wrt A



// 2. Compute new vertices
Foreach edge e from A
    foreach face f from B
      if intersect(f,e) add(intersectionVertex (f,e),LV)

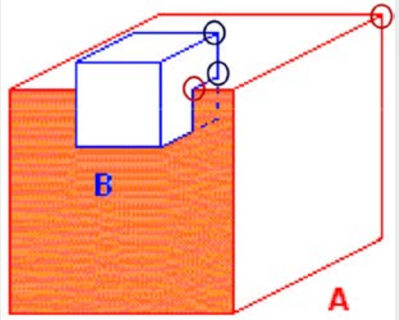
Foreach edge e from B
    foreach face f from A
      if intersect(f,e) add(intersectionVertex (f,e),LV)

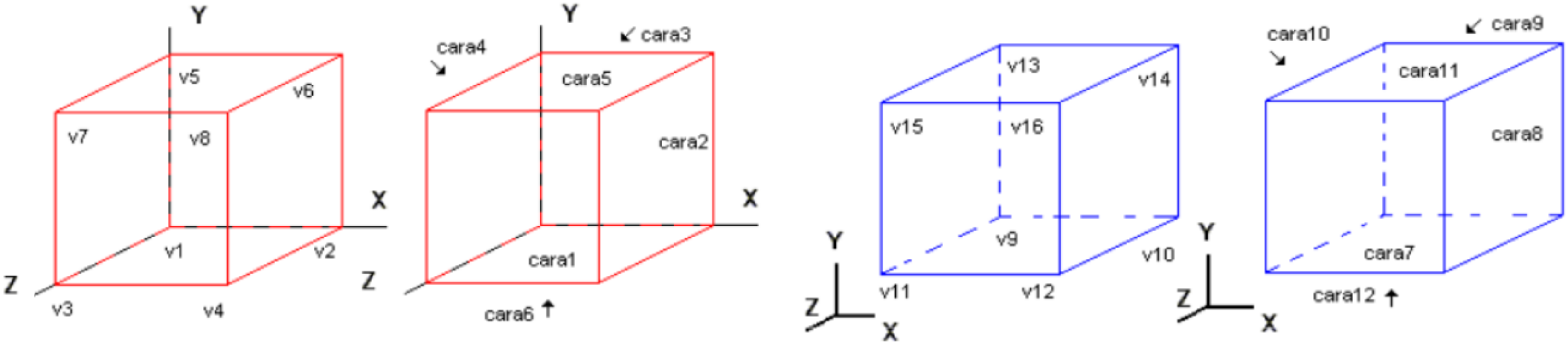
Algorithm 2




```
// 3. Select output vertices according to the boolean operation
foreach vertex v in LV
  if v.type=NEW  add(result,v) otherwise
  case
    union: if v.type = deAoutB or v.type = deBoutA  add(result,v)
    inters: if v.type = deAinB or v.type = deBinA  add(result,v)
    A-B : if v.type = deAoutB or v.type = deBinA  add(result,v)
    B-A : if v.type = deAinB or v.type = deBoutA  add(result,v)
  end
  // 4. Build F:{V} from V:{F}
  buildFaces(C) // change from reverse rep. to hierarchical rep.
end
```

Example 1: A-B



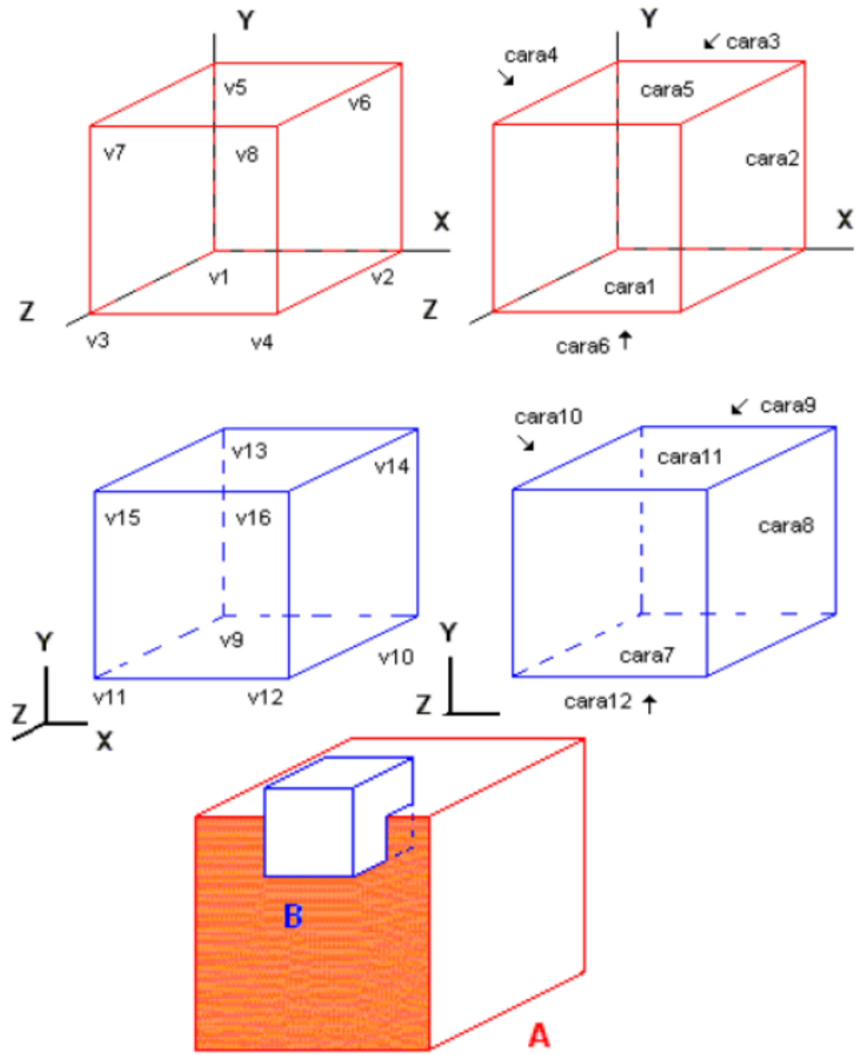


Objecte A Objecte B

Cares	Vertexs	Cares	Vertexs
1:{3,4,8,7}	1:(0,0,0)	7:{11,12,16,15}	9:(1,2,2)
2:{2,6,8,4}	2:(3,0,0)	8:{10,14,16,12}	10:(2,2,2)
3:{1,5,6,2}	3:(0,0,3)	9:{ 9,13,14,10}	11:(1,2,5)
4:{1,3,7,5}	4:(3,0,3)	10:{ 9,11,15,13}	12:(2,2,5)
5:{5,7,8,6}	5:(0,3,0)	11:{13,15,16,14}	13:(1,4,2)
6:{1,2,4,3}	6:(3,3,0)	12:{ 9,10,12,11}	14:(2,4,2)
	7:(0,3,3)		15:(1,4,5)
	8:(3,3,3)		16:(2,4,5)

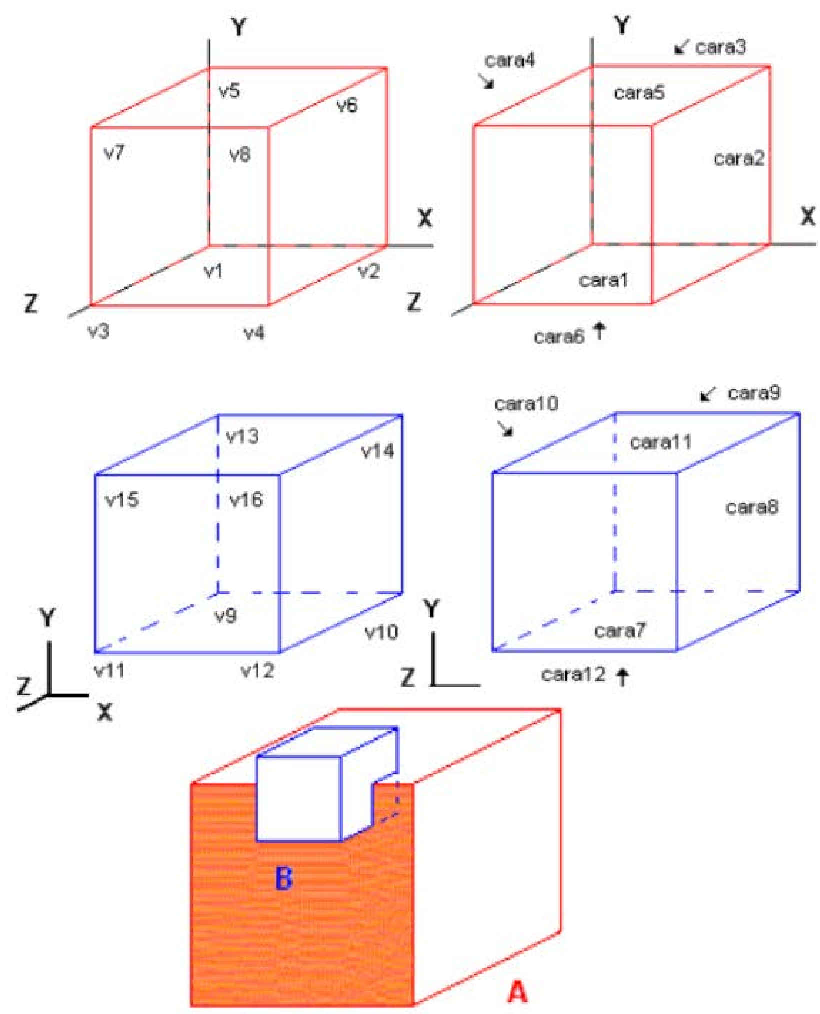
Step 1: Classify vertices

```
Exemple 1

V 1: xyz, {4,3,6},   deAoutB
V 2: xyz, {3,2,6},   deAoutB
V 3: xyz, {1,4,6},   deAoutB
V 4: xyz, {2,1,6},   deAoutB
V 5: xyz, {5,3,4},   deAoutB
V 6: xyz, {5,2,3},   deAoutB
V 7: xyz, {1,5,4},   deAoutB
V 8: xyz, {2,5,1},   deAoutB
V 9: xyz, {9,10,12}, deBinA
V10: xyz, {9,8,12},  deBinA
V11: xyz, {7,10,12}, deBoutA
V12: xyz, {8,7,12},  deBoutA
V13: xyz, {11,9,10}, deBoutA
V14: xyz, {11,8,9},  deBoutA
V15: xyz, {7,11,10}, deBoutA
V16: xyz, {8,11,7},  deBoutA
```



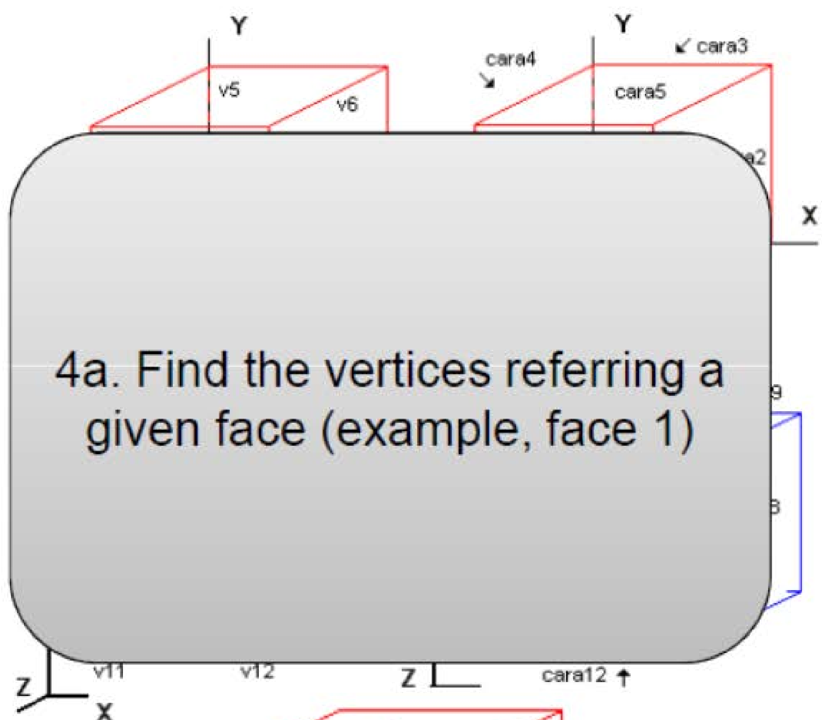
Step 3: Select output vertices

```
V 1: xyz, {4,3,6},   deAoutB
V 2: xyz, {3,2,6},   deAoutB
V 3: xyz, {1,4,6},   deAoutB
V 4: xyz, {2,1,6},   deAoutB
V 5: xyz, {5,3,4},   deAoutB
V 6: xyz, {5,2,3},   deAoutB
V 7: xyz, {1,5,4},   deAoutB
V 8: xyz, {2,5,1},   deAoutB
V 9: xyz, {9,10,12}, deBinA
V10: xyz, {9,8,12},  deBinA
V11: xyz, {7,10,12}, deBoutA
V12: xyz, {8,7,12},  deBoutA
V13: xyz, {11,9,10}, deBoutA
V14: xyz, {11,8,9},  deBoutA
V15: xyz, {7,11,10}, deBoutA
V16: xyz, {8,11,7},  deBoutA
V17: xyz, {9,10,5}, Nou
V18: xyz, {8,9,5},  Nou
V19: xyz, {1,5,10}, Nou
V20: xyz, {1,5,8},  Nou
V21: xyz, {10,12,1},Nou
V22: xyz, {8,12,1}, Nou
```

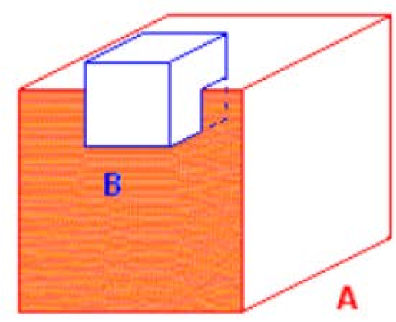


Step 4: Build faces

```
V 1: xyz, {4,3,6},   deAoutB
V 2: xyz, {3,2,6},   deAoutB
V 3: xyz, {1,4,6},   deAoutB
V 4: xyz, {2,1,6},   deAoutB
V 5: xyz, {5,3,4},   deAoutB
V 6: xyz, {5,2,3},   deAoutB
V 7: xyz, {1,5,4},   deAoutB
V 8: xyz, {2,5,1},   deAoutB
V 9: xyz, {9,10,12}, deBinA
V10: xyz, {9,8,12},  deBinA
V11: xyz, {7,10,12}, deBoutA
V12: xyz, {8,7,12},  deBoutA
V13: xyz, {11,9,10}, deBoutA
V14: xyz, {11,8,9},  deBoutA
V15: xyz, {7,11,10}, deBoutA
V16: xyz, {8,11,7},  deBoutA
V17: xyz, {9,10,5}, Nou
V18: xyz, {8,9,5},  Nou
V19: xyz, {1,5,10}, Nou
V20: xyz, {1,5,8},  Nou
V21: xyz, {10,12,1},Nou
V22: xyz, {8,12,1}, Nou
```

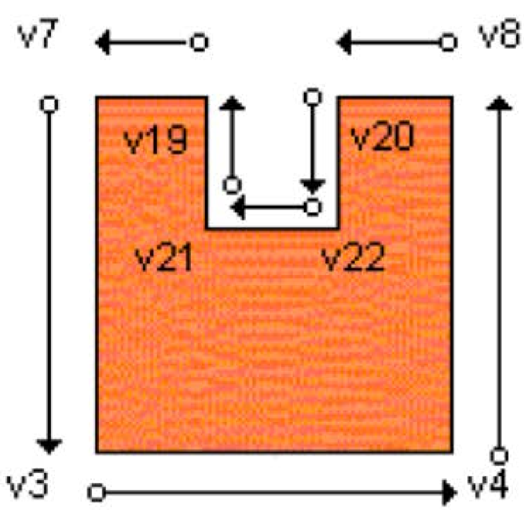


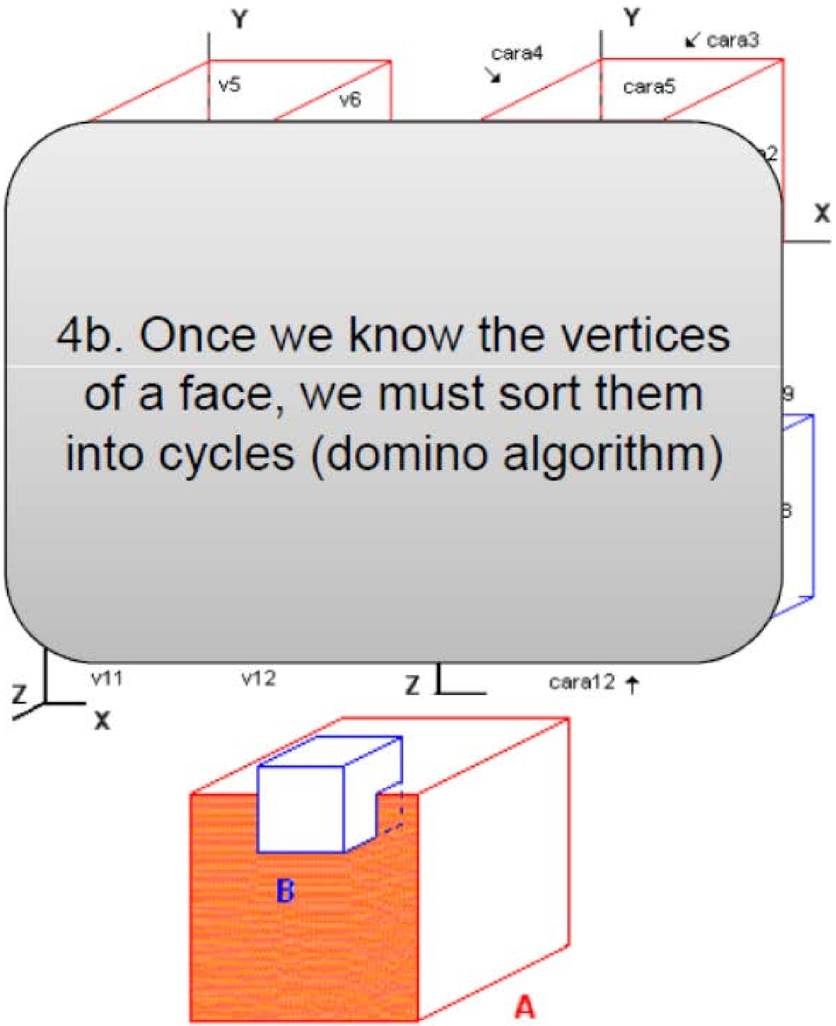
4a. Find the vertices referring a given face (example, face 1)



Step 4: Build faces

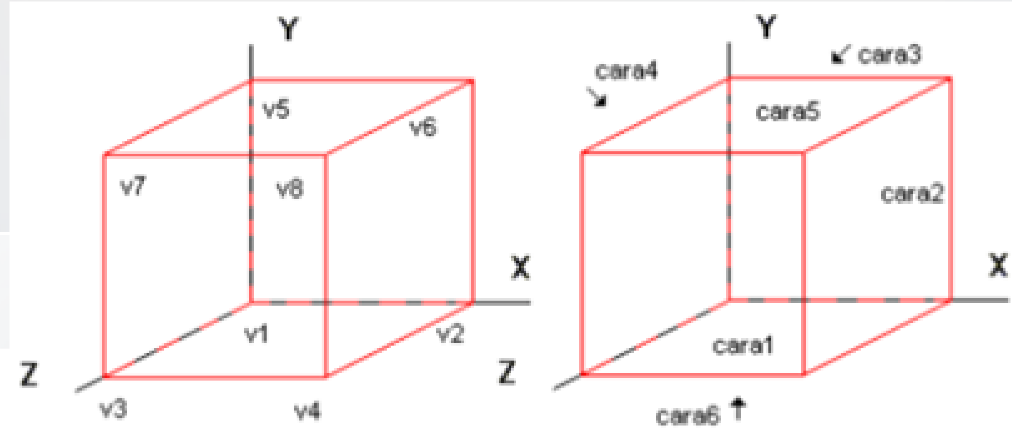
```
V 3: xyz, {4,1,6}, deAoutB
V 4: xyz, {2,1,6}, deAoutB
V 7: xyz, {4,1,5}, deAoutB
V 8: xyz, {5,1,2}, deAoutB

V19: xyz, {10,1,5}, Nou
V20: xyz, {8,1,5},  Nou
V21: xyz, {12,1,10},Nou
V22: xyz, {8,1,12}, Nou
```



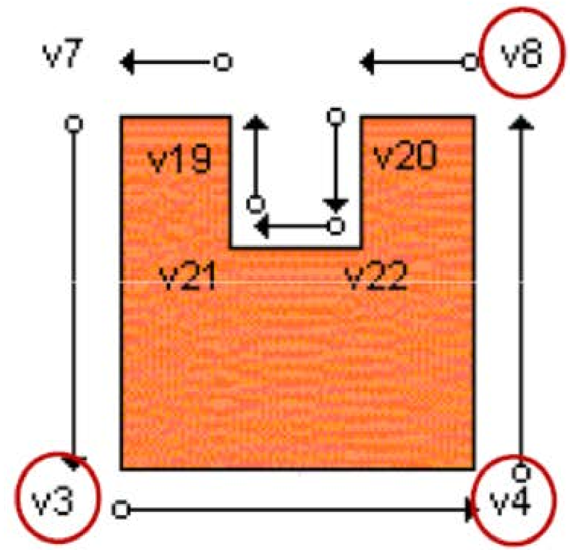


4b. Once we know the vertices of a face, we must sort them into cycles (domino algorithm)

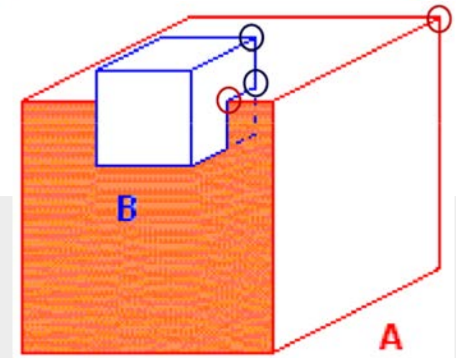
Step 4: Build faces




```
❶V 3: xyz,  {4,1,6},  deAoutB
❷V 4: xyz,  {2,1,6},  deAoutB
 V 7: xyz,  {4,1,5},  deAoutB
❸V 8: xyz,  {5,1,2},  deAoutB

 V19: xyz,  {10,1,5},  Nou
 V20: xyz,  {8,1,5},   Nou
 V21: xyz,  {12,1,10},Nou
 V22: xyz,  {8,1,12},  Nou
```




```
V3 ------> V4 ------> V8 ------> ?
    Cara 6     Cara 2     Cara 5
```

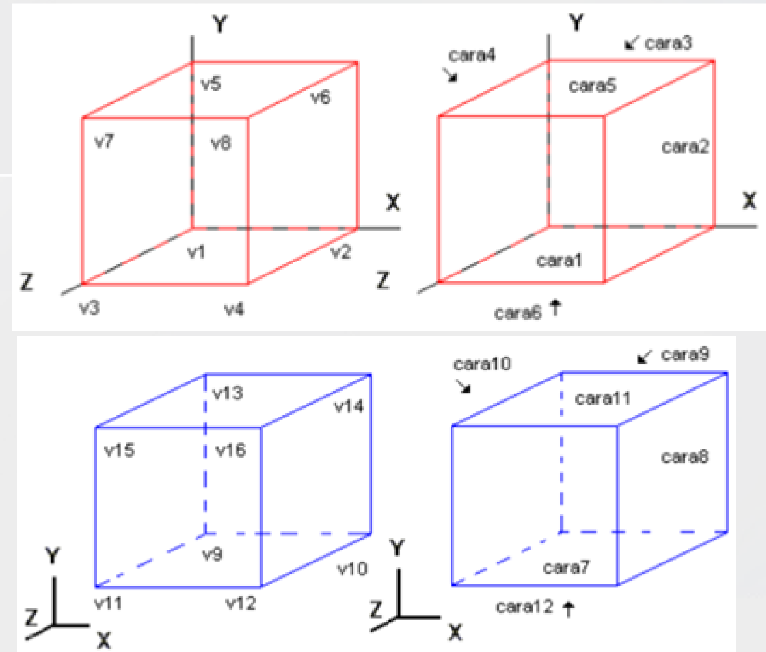


Step 4: Build faces

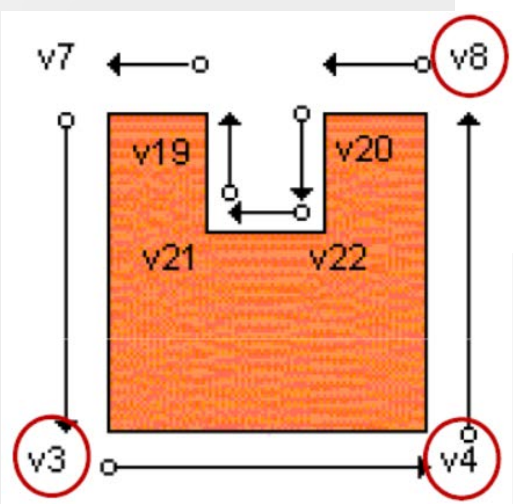
To solve the indetermination:

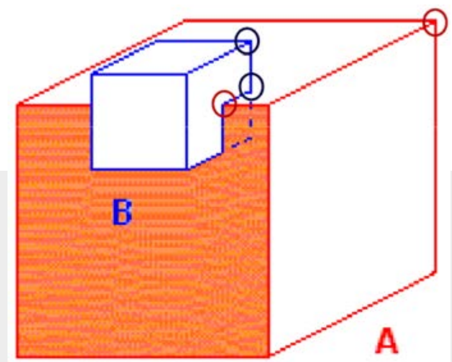
1. Sort the vertices involved according to the parameter of the supporting line: V8, V20, V19, V7

2. Group forming pairs (will become edges of the result): (V8, V20) (V19, V7).



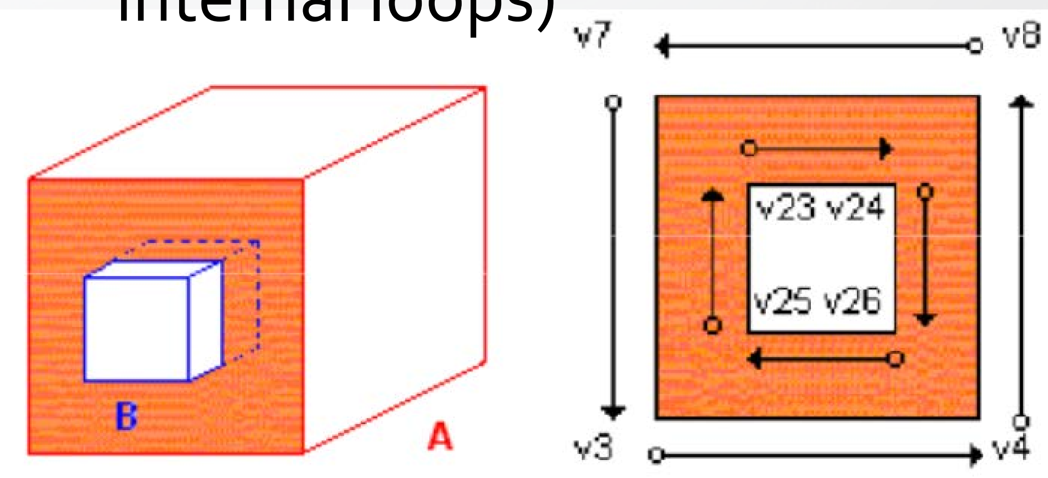
V3 ------> V4 ------> V8 ------> V20 ------> V22 ------> V21 -------> V19
Cara 6 Cara 2 Cara 5 Cara 8 Cara 12 Cara 10
| Cara
V3 <------ V7 <--
Cara 4

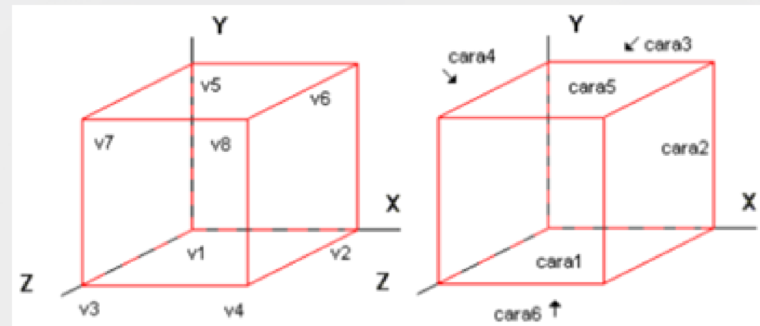


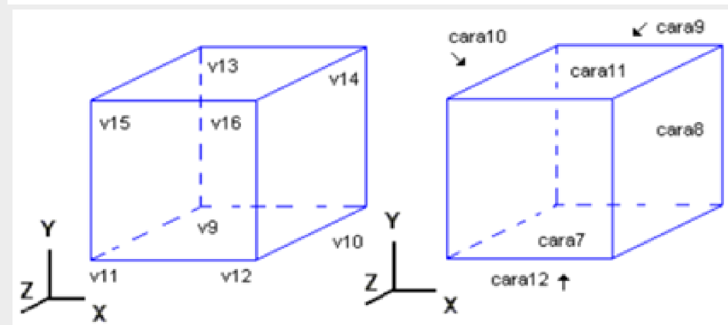


Example 2. Still A-B

The domino algorithm can detect more than one cycle (faces with internal loops)

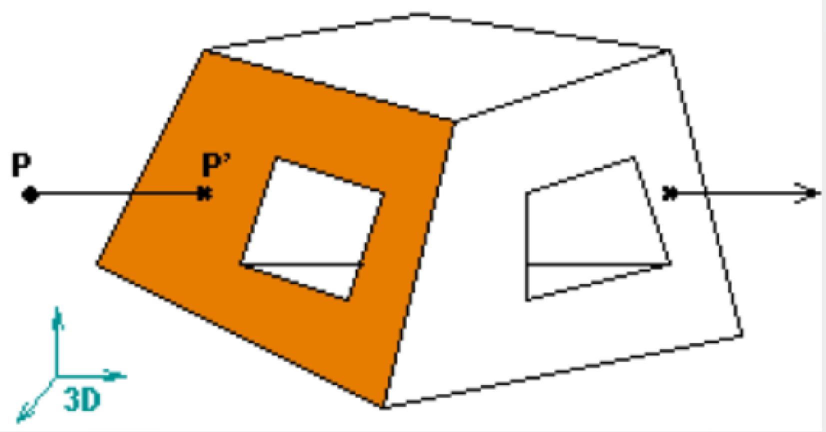







```
    V3 ------> V4 ------> V8 ------> V7 ------> V3
       Cara 6     Cara 2     Cara 5     Cara 4

V23 -------> V24 ------> V26 -------> V25 -------> V23
   Cara 11      Cara 8      Cara 12       Cara 10
```

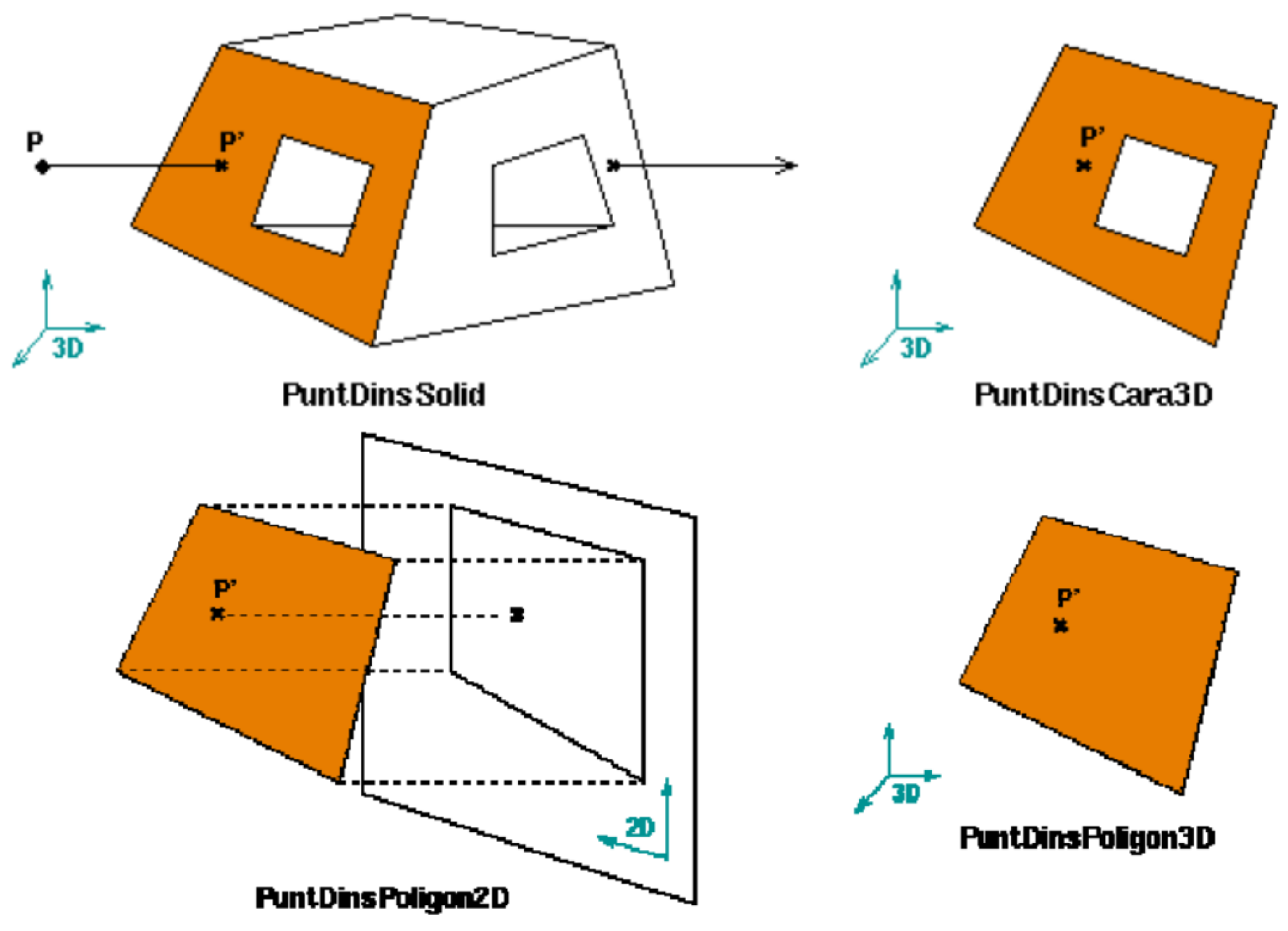
Geometric tests



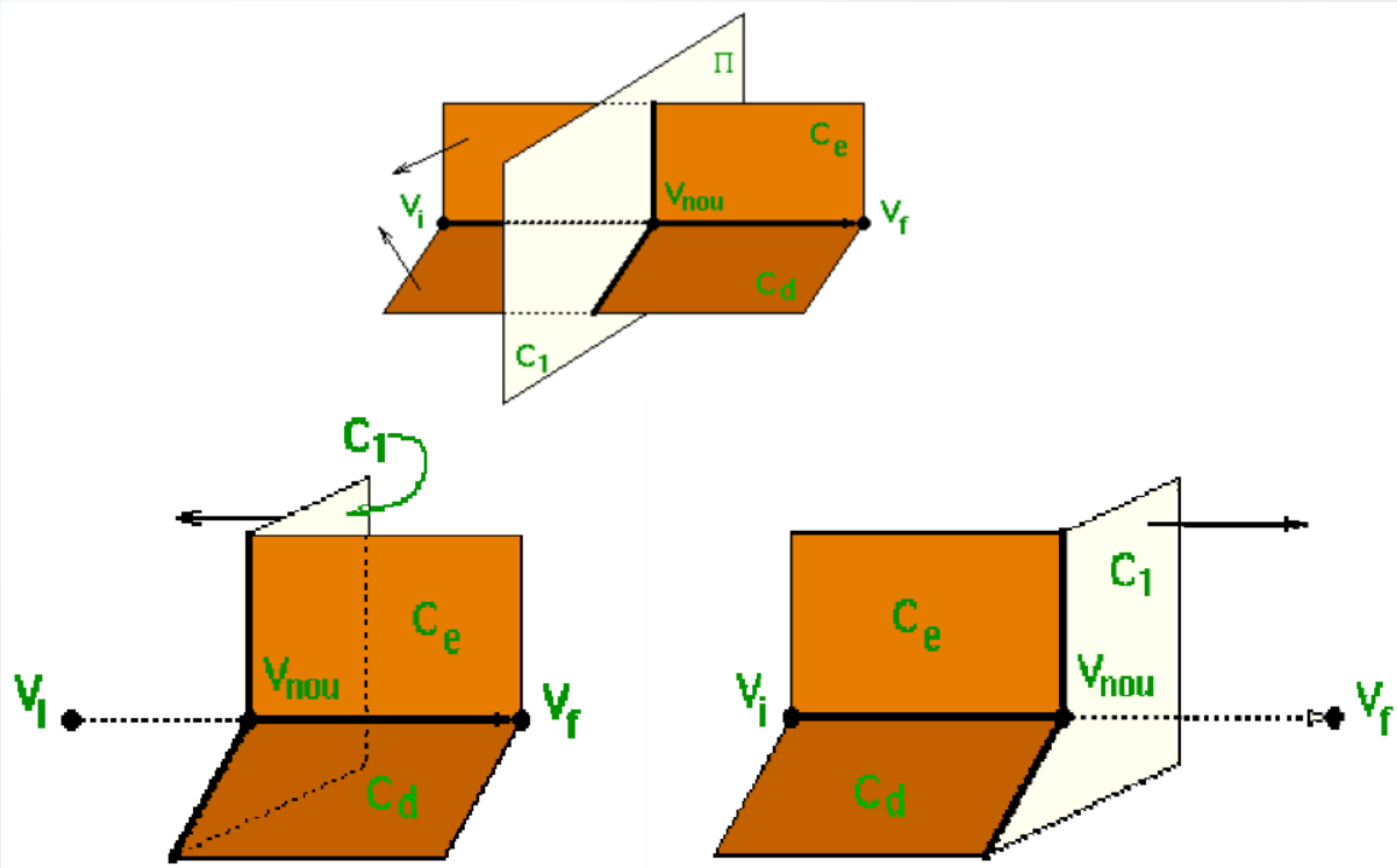
- Point inside solid

- Convexity of an edge
- Sorting faces around a vertex
- Classify cycles as interior/exterior

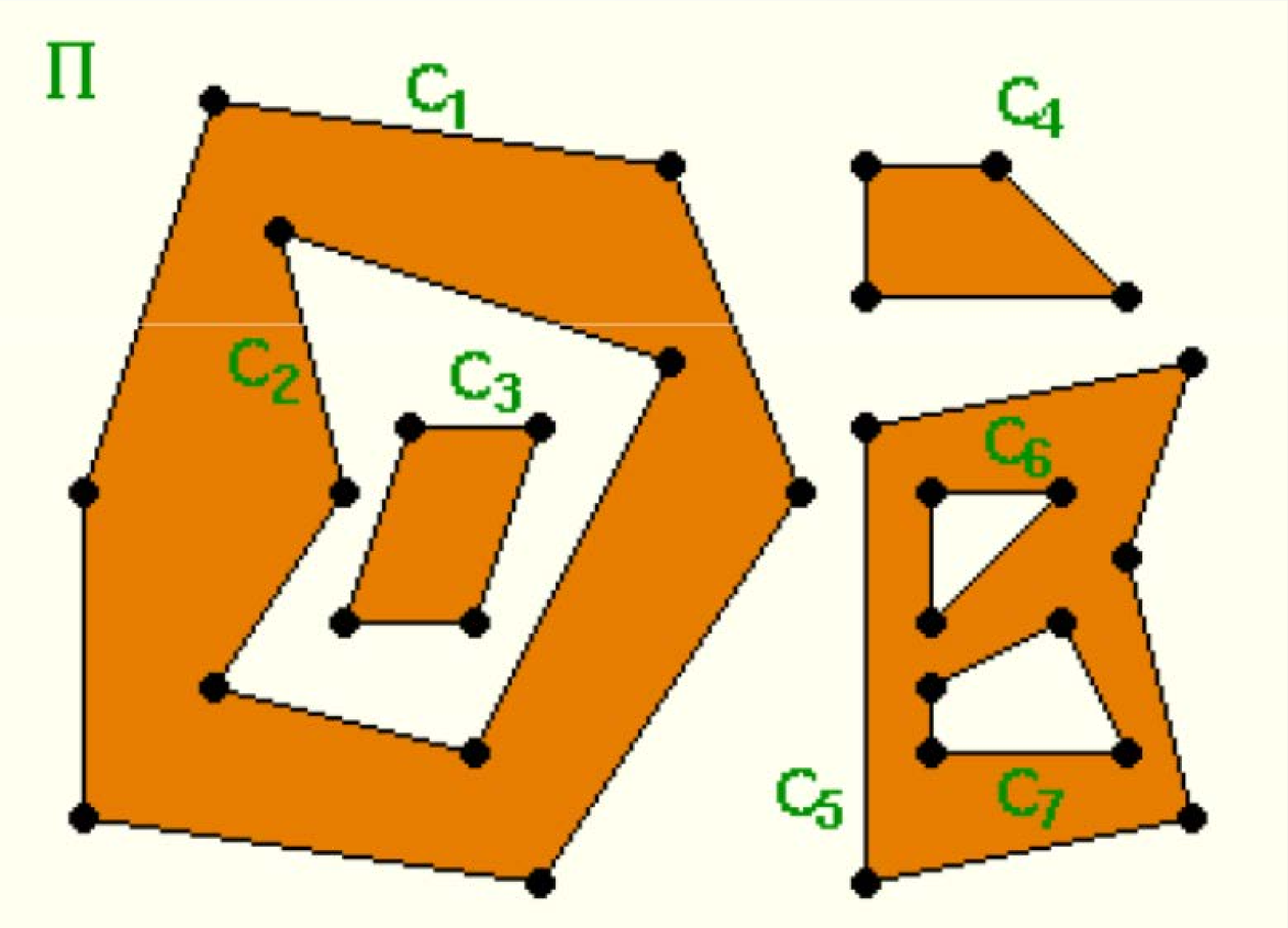
Point inside solid



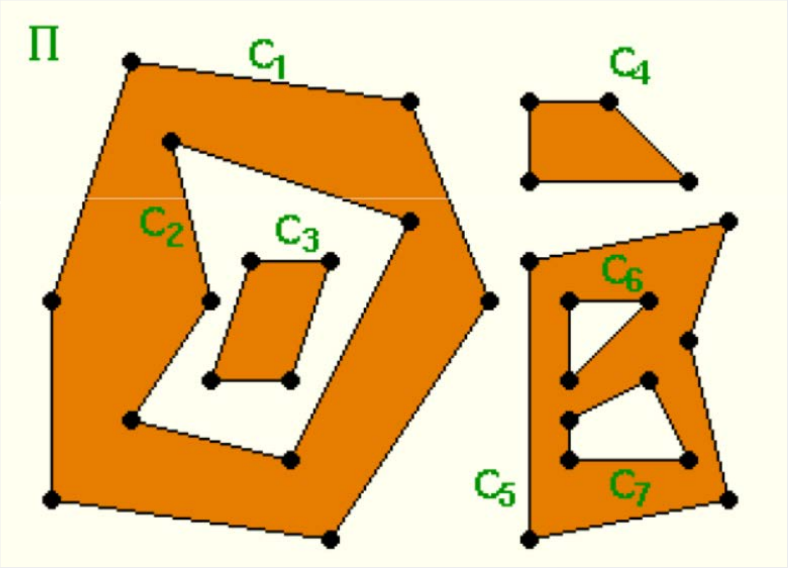
Sorting faces around a vertex



Classify cycles as in/out



Classify cycles as in/out



parity=**true**
C:=set of loops
while C is not empty **do**
    D := Ø
  for each loop cx in C fer
    if cx is inside to some loop cy in C **then**
        classify cx as an internal loop of cy
    else D := D + {cx}
  if parity **then** loops in D are exterior loops of faces
  else the loops in D are interior loops
  parity:=**not** parity
  C:=C-D
end

Boolean Model Construction

Boolean Model: combination of > 1 simpler solid objects.

Boolean Model is *procedural*: shows how to combine parts.

Assume regularization.

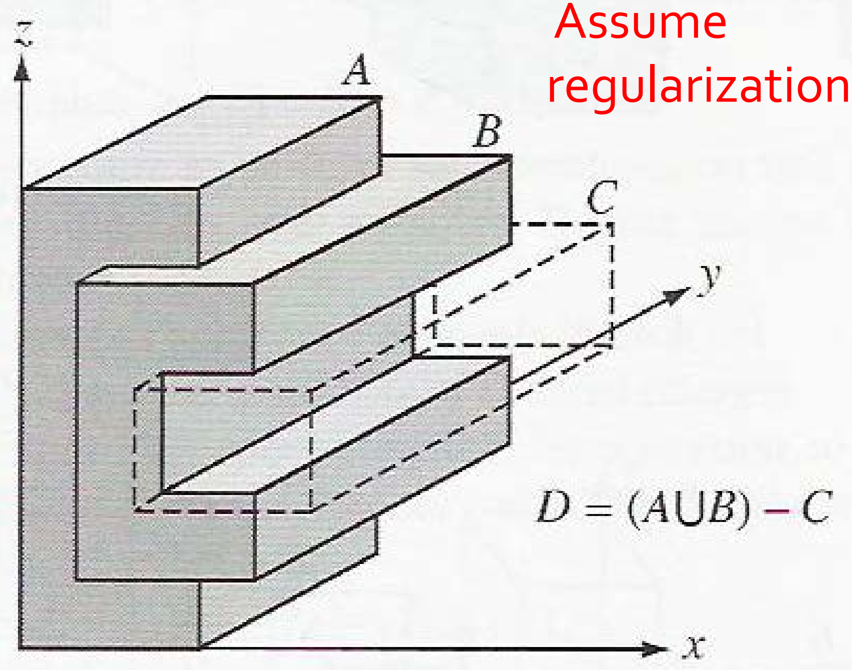


Figure 11.40 A simple procedural model.

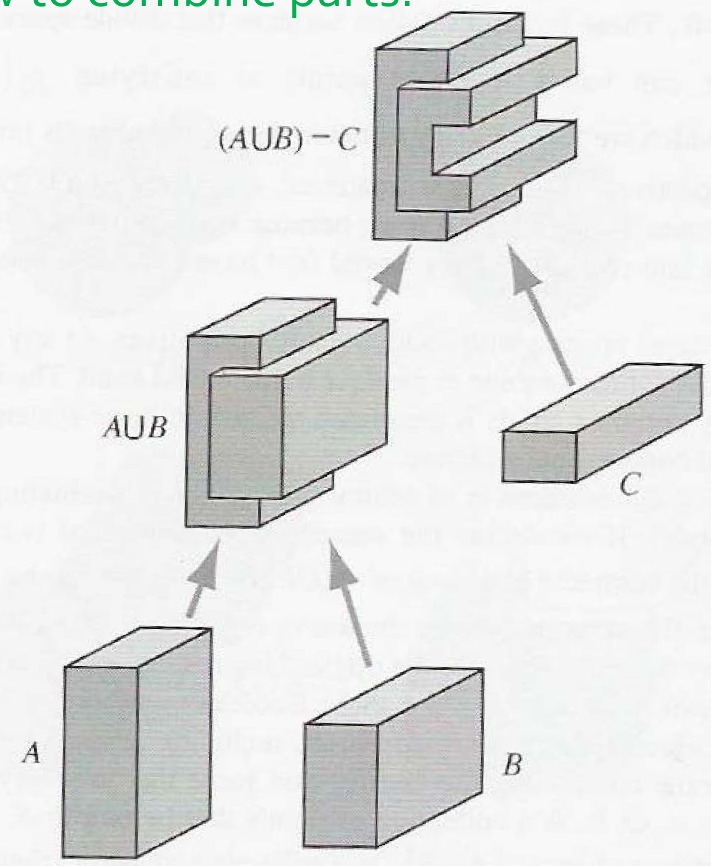


Figure 11.41 The binary tree for $D = (A \cup B) - C$.

Boolean Model Construction (continued)

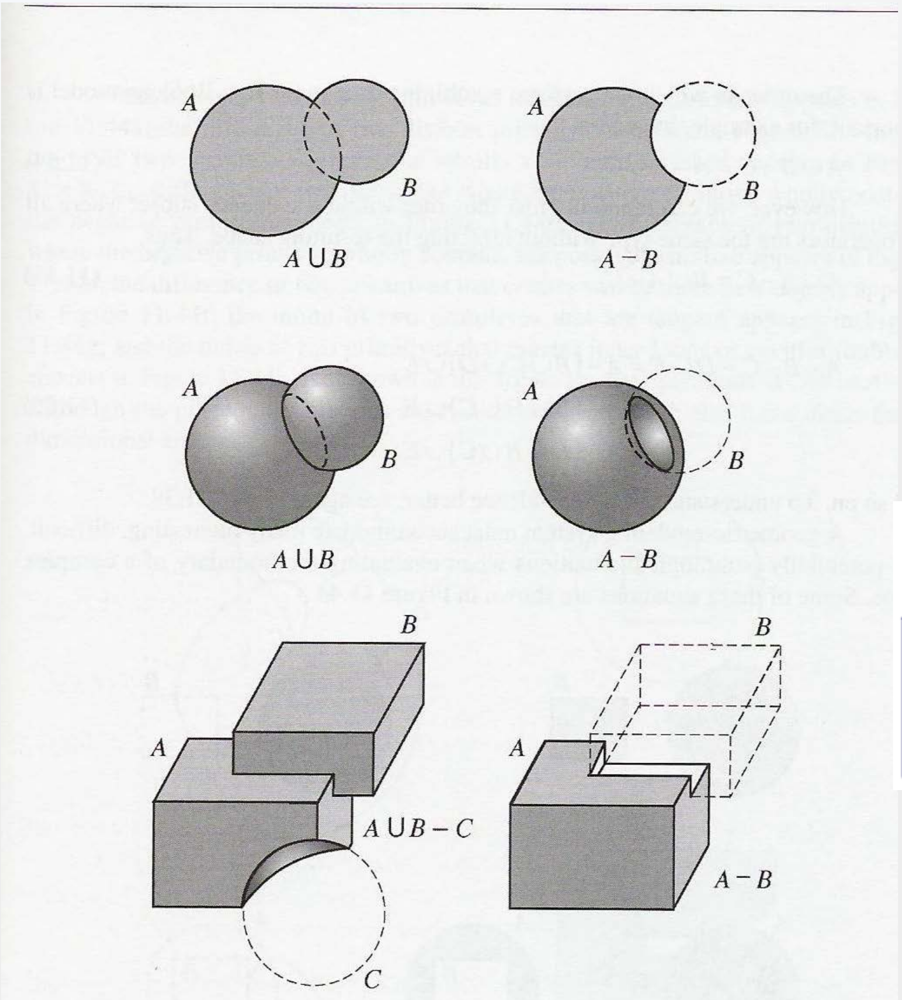
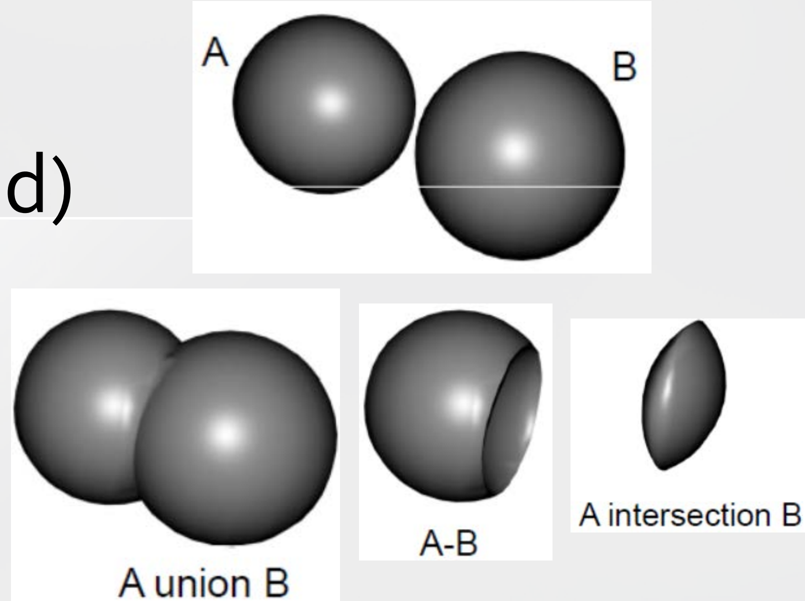
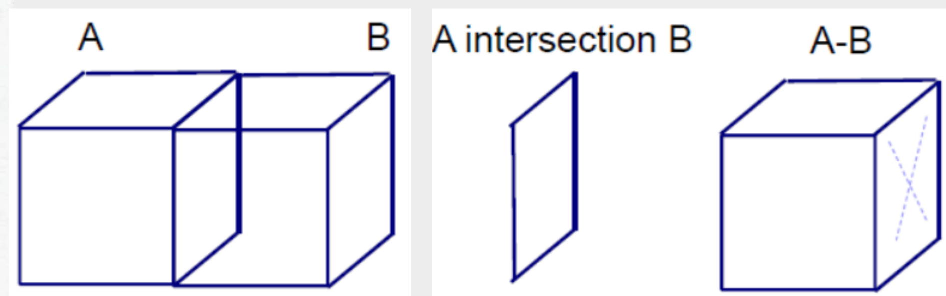


Figure 11.42 Examples of union and difference.



Coincidences problem



Boolean Model Construction (continued)

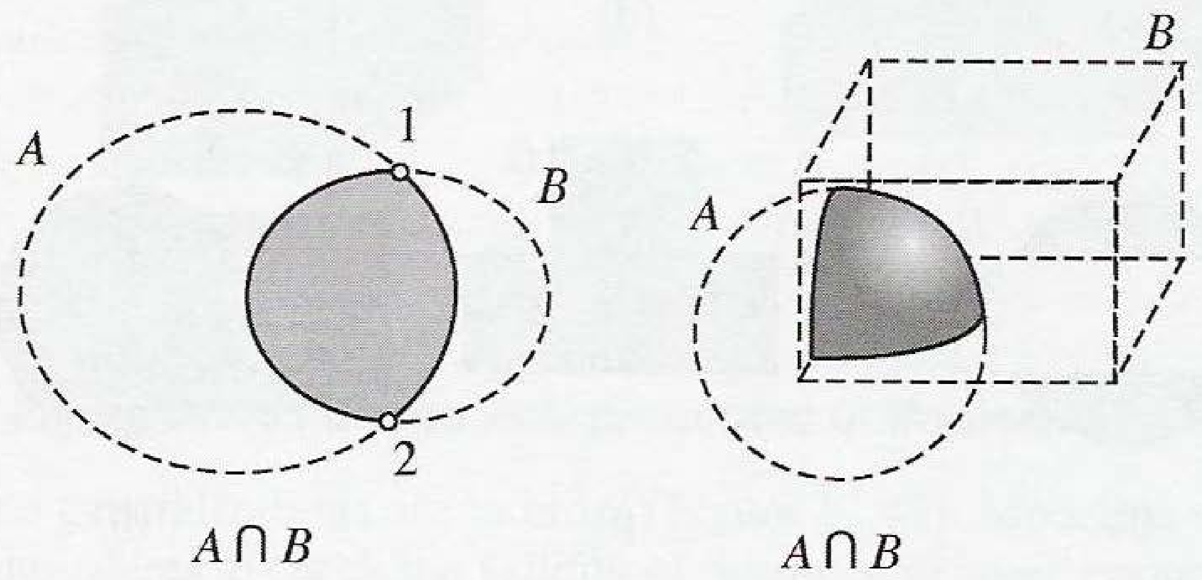


Figure 11.43 The intersection operation.

Boolean Model Construction (continued)

(a) union of disjoint A and B

(b) difference of disjoint A and B: A - B

(c) union of A encompassing B

(d) difference of A encompassing B: A - B

(e) B - A

(f) A – B yields 2 objects

(g) –(h): union of A and B

(h) makes concavity

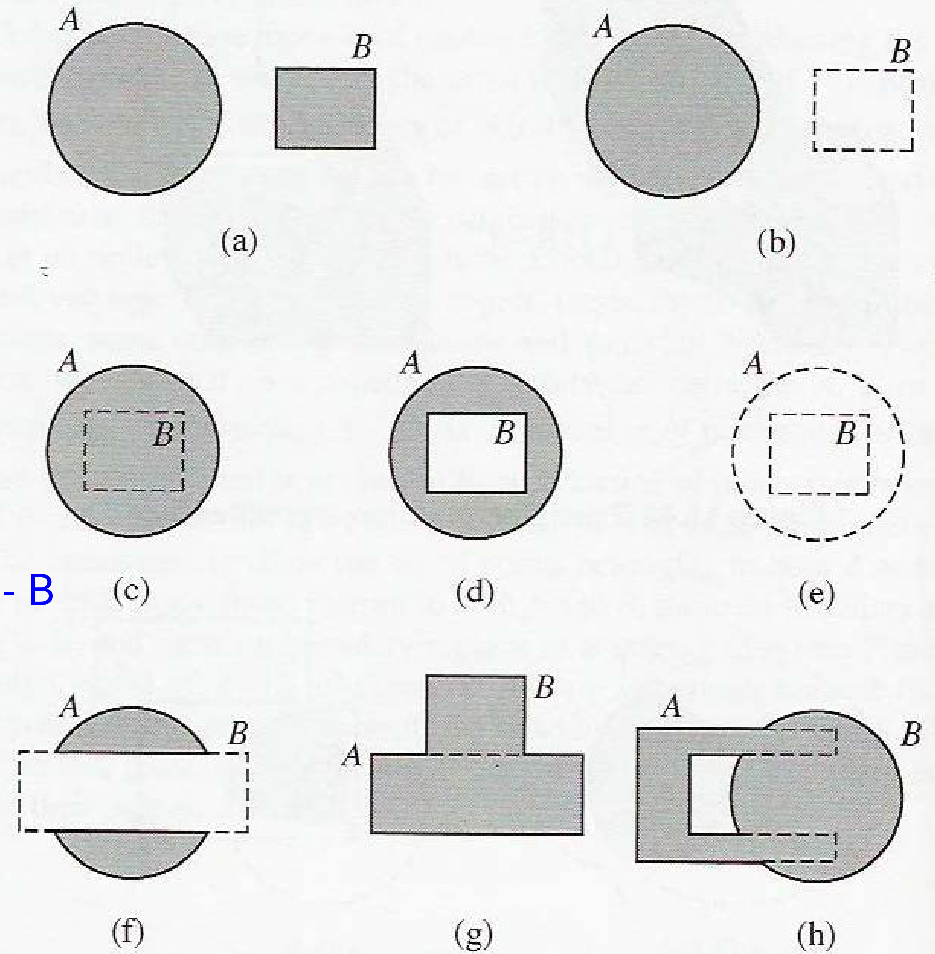


Figure 11.44 A variety of Boolean modeling situations.

Boolean Model Construction (continued)

(a) 2 intersecting, closed, planar curves intersect an even number of times.

(b) If curves A and B do not intersect & a point of B is inside curve A, then B is inside A.

(c) Closed, planar curve intersects 3D solid an even number of times.

(d) Plane P intersects bounding surface of S in 3 disjoint, closed loops.

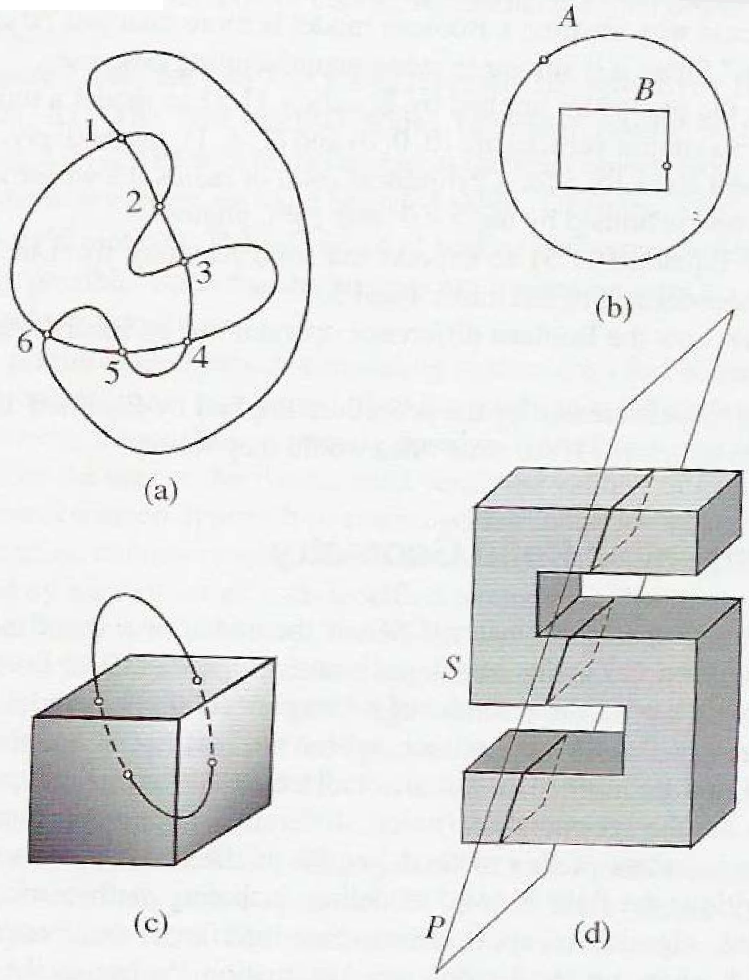


Figure 11.45 Four general properties of Boolean models.

Constructive Solid Geometry (CSG)

CSG: Modeling methods defining complex solids as compositions of simpler solids.

Root node represents final result.

Internal nodes represent Boolean operations & their results.

Leaves are primitive shapes.

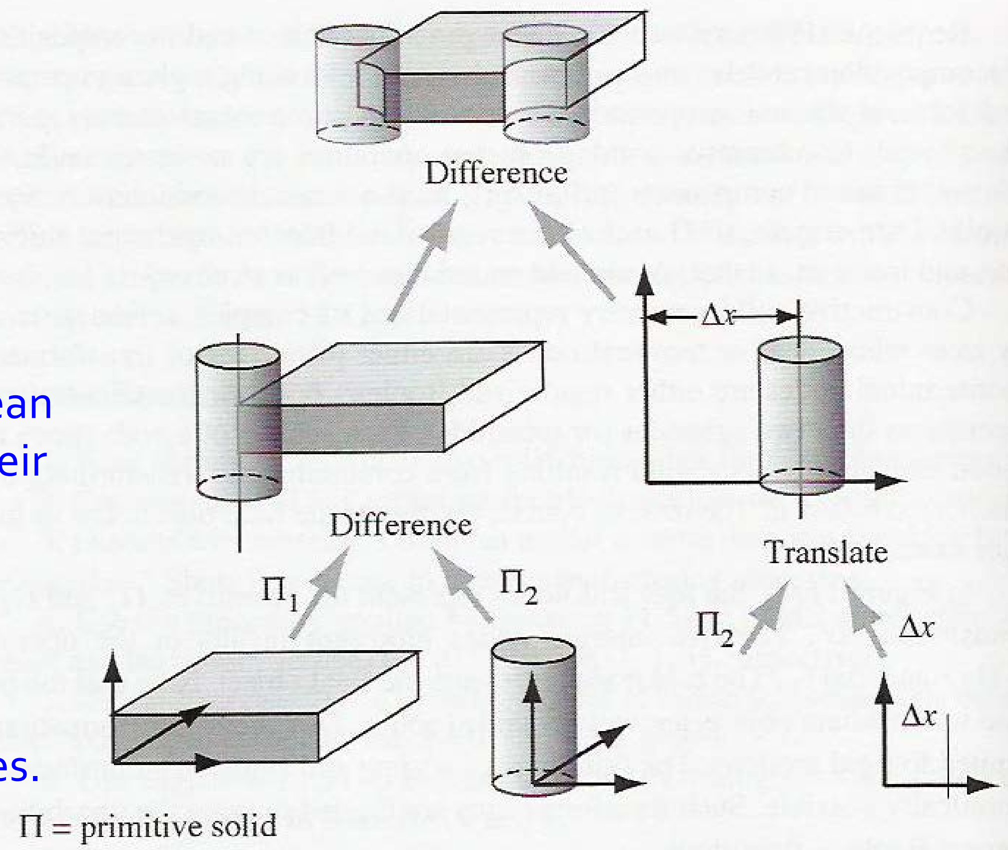


Figure 11.46 Constructive solid geometry representation.

Constructive Solid Geometry (continued)

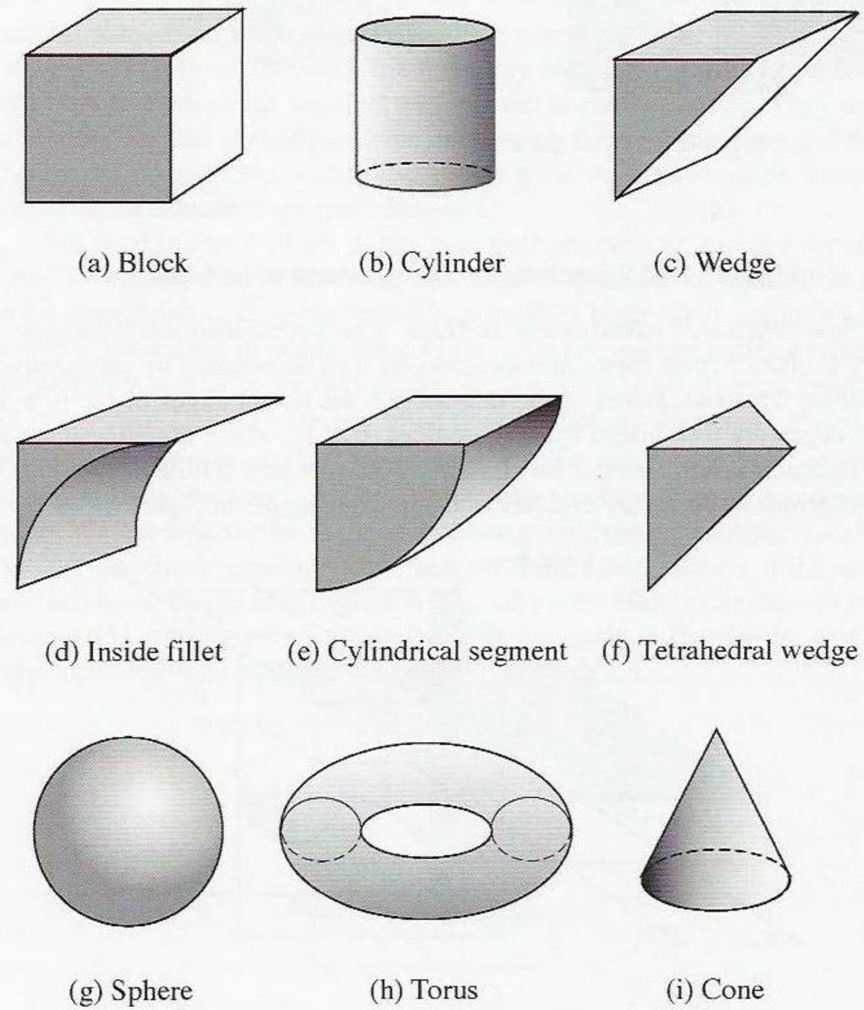


Figure 11.47 Primitive solids.

Constructive Solid Geometry (continued)

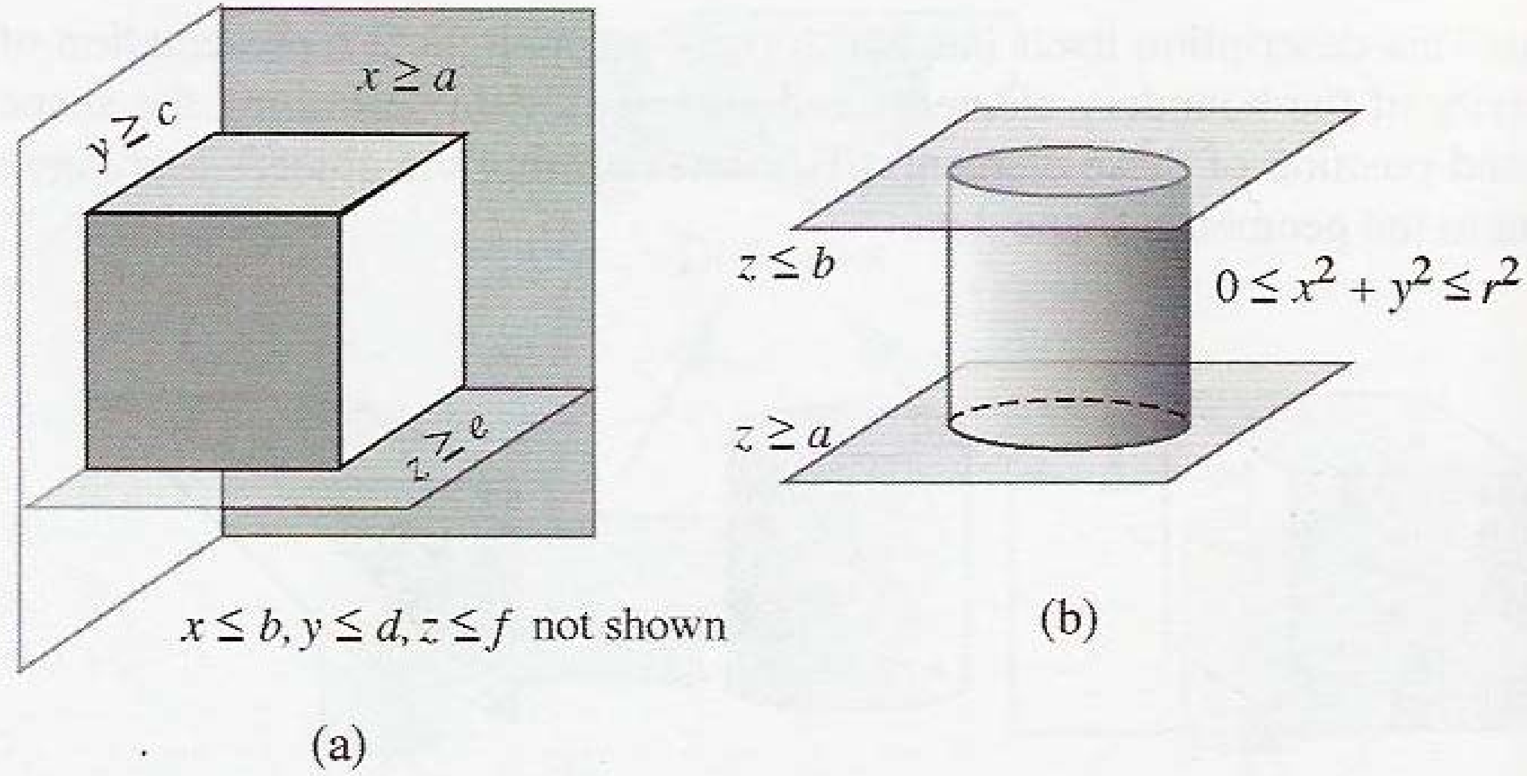
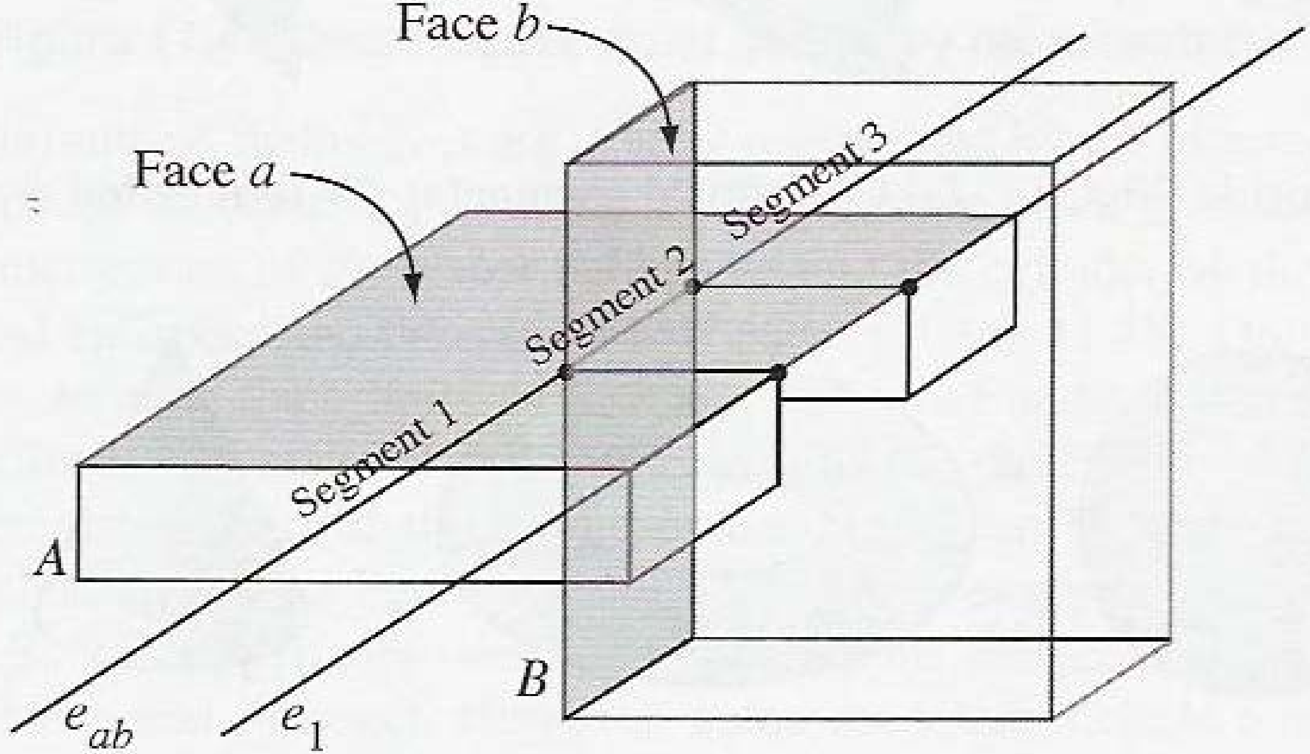


Figure 11.48 Primitives as intersections of halfspaces.

Constructive Solid Geometry (continued)



$$C = A - B$$

Figure 11.49 Boundary evaluation.

Constructive Solid Geometry (continued)

Refer to Figure 11.49 on previous slide.

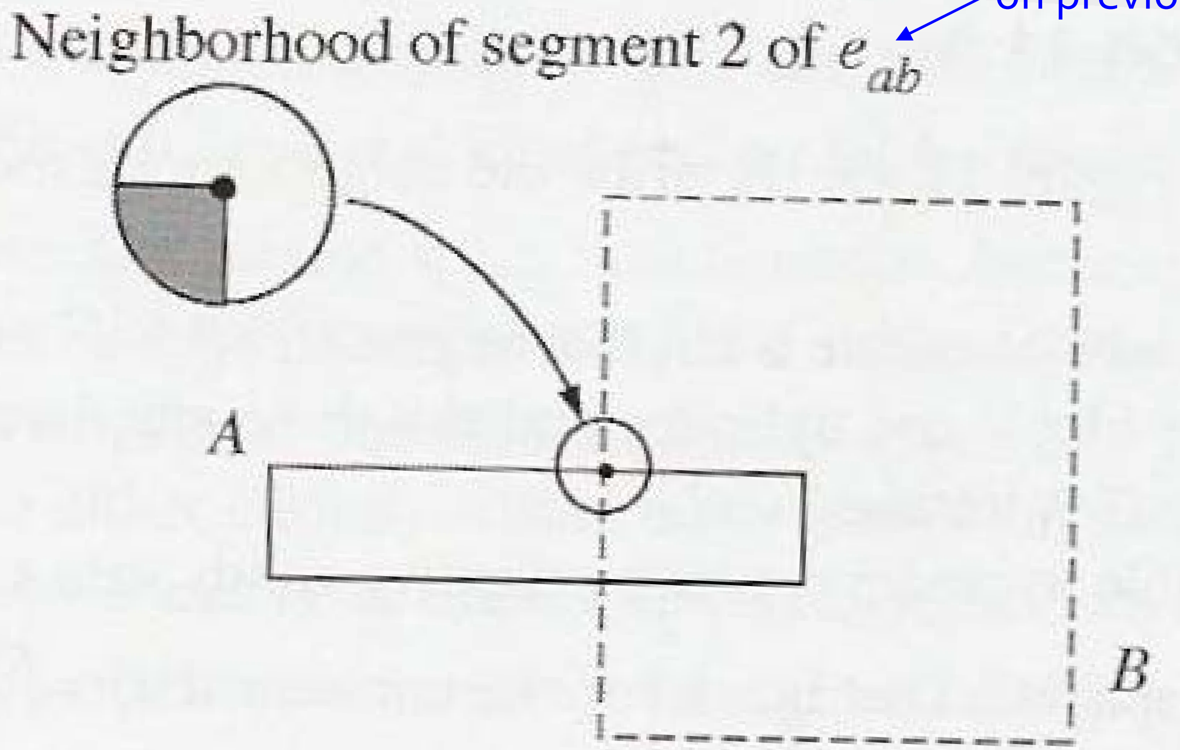


Figure 11.50 Neighborhood model.

Constructive Solid Geometry (continued)

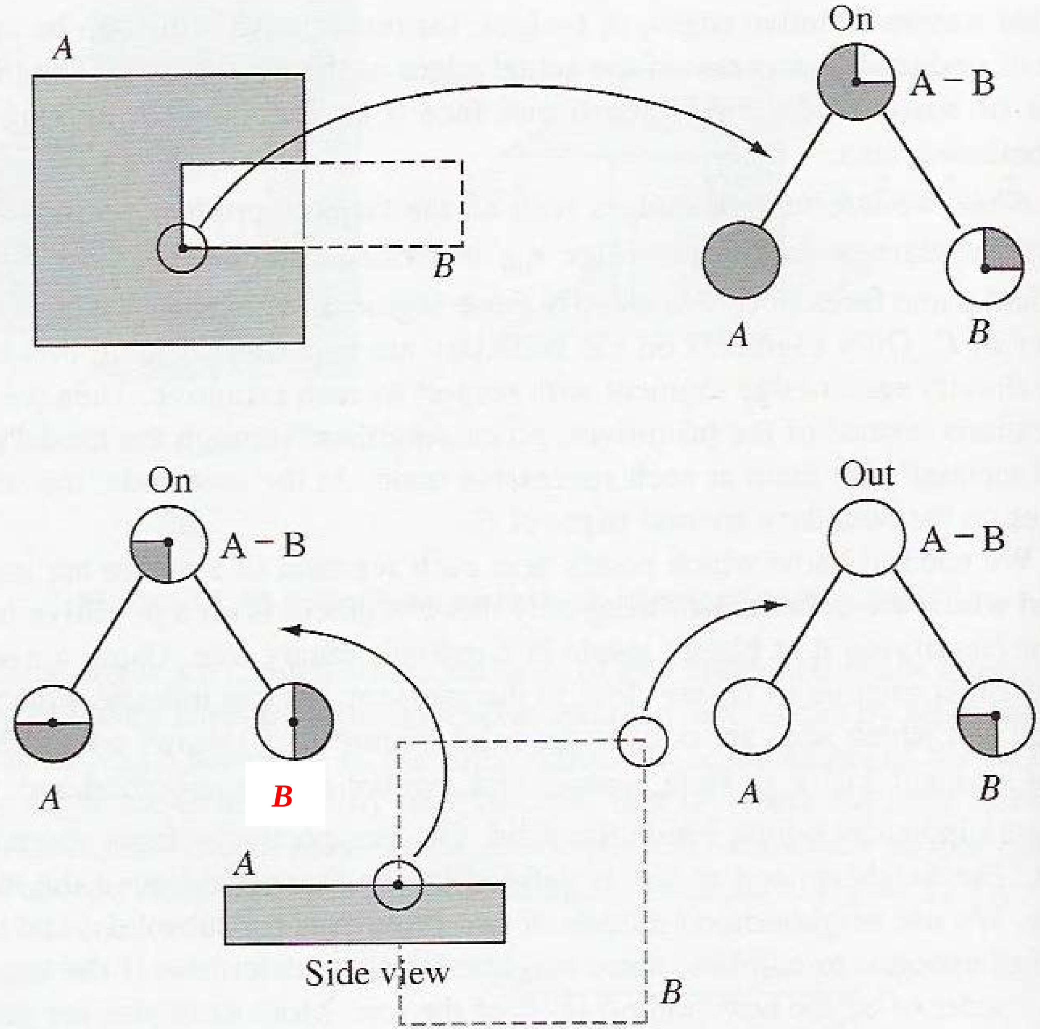


Figure 11.51 Combining neighborhood models.

Boundary Models

Boundary Model: complete representation of a solid as an organized collection of surfaces.

Boundary of a solid must be:

- closed
- orientable
- non-self-intersecting
- bounding
- connected

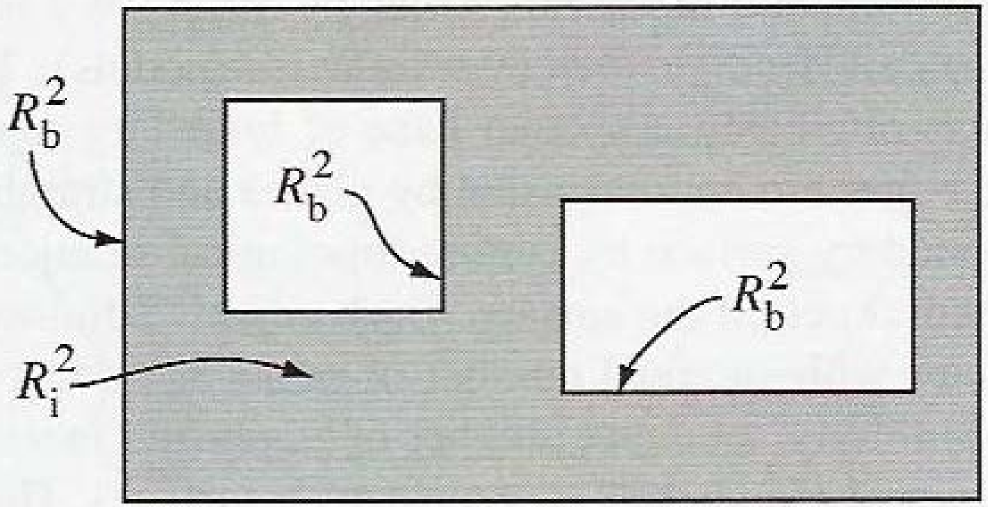


Figure 11.52 A plane figure and its boundaries.

Region R^n is finite, bounded portion of E^n.

$$R = [R_i, R_b]$$

Boundary Models (continued)

Boundary Representation (B-Rep)

- B-Rep minimal *face* conditions:
  - Number of faces is finite.
  - Face is subset of solid's boundary.
  - Union of faces defines boundary.
  - Face is subset of more extensive surface (e.g. plane).
  - Face has finite area.
  - Face is dimensionally homogeneous (regularized).

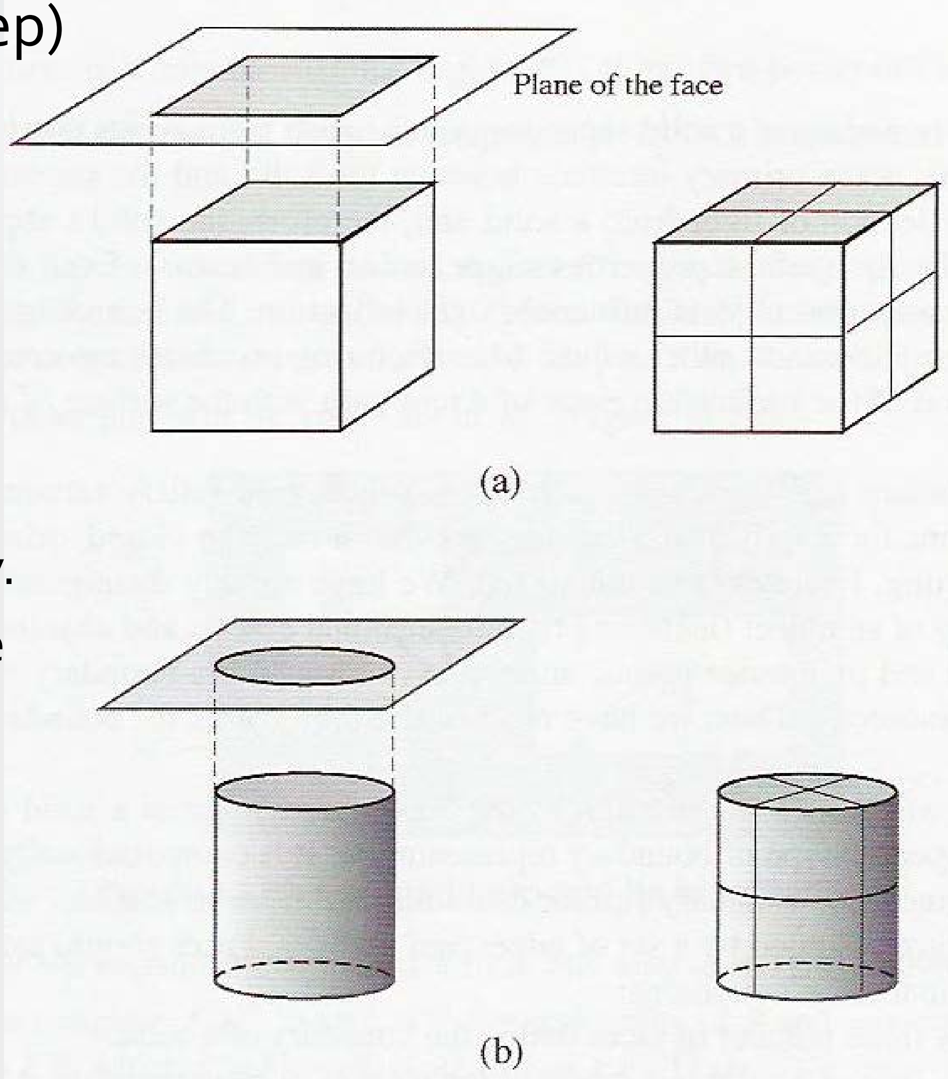


Figure 11.53 Faces defining the boundary of a solid.

Boundary Models (continued)

Boundary Representation (B-Rep)

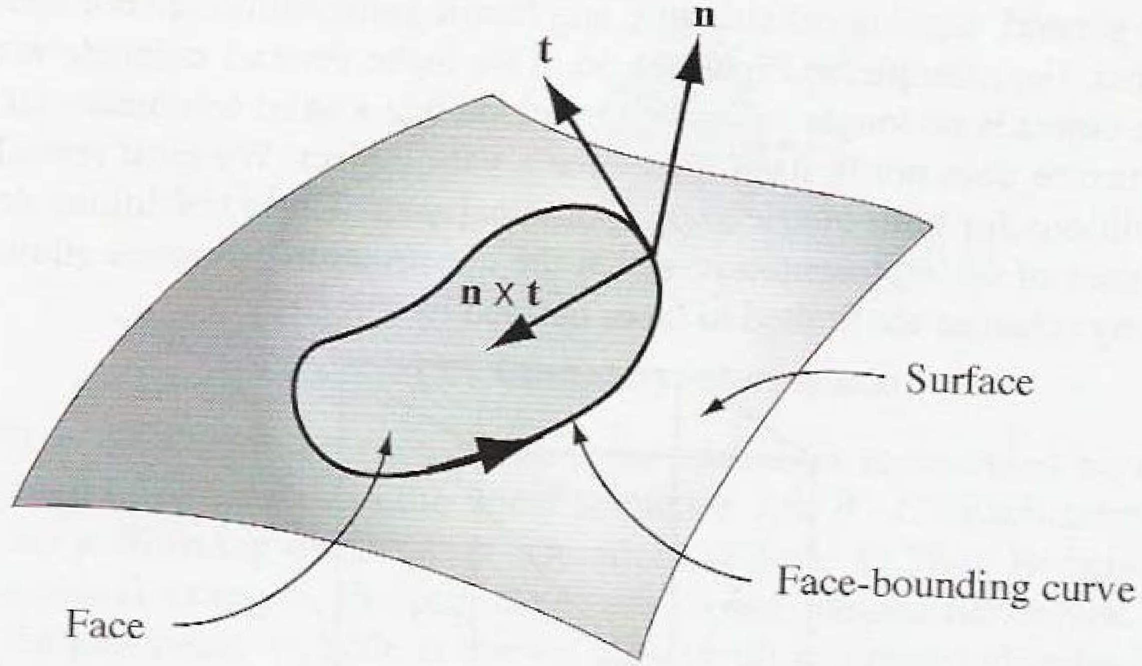
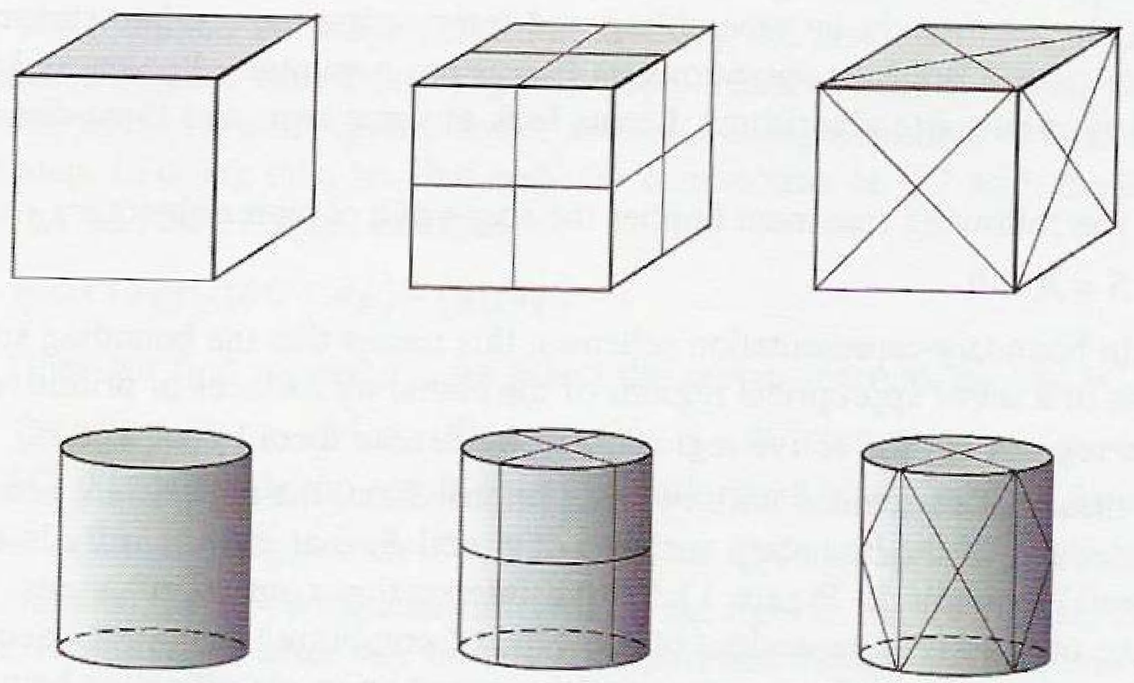


Figure 11.54 Face boundary convention.

Curved boundary faces require inside/outside convention.

Boundary Models (continued)

Boundary Representation (B-Rep)



Boundary representations are not unique.

Boundary Models (continued)

Boundary Representation (B-Rep)

Merging vertex 1 with vertex 2 makes object invalid.

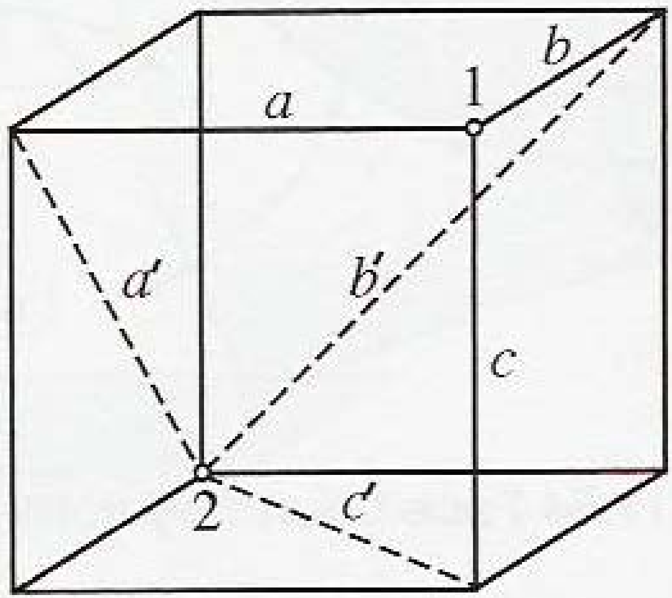


Figure 11.56 Interdependence of topology and geometry.

Boundary Models (continued)

Boundary Representation (B-Rep)

Powerful B-rep systems view solid as union of general faces (e.g. parametric curves).

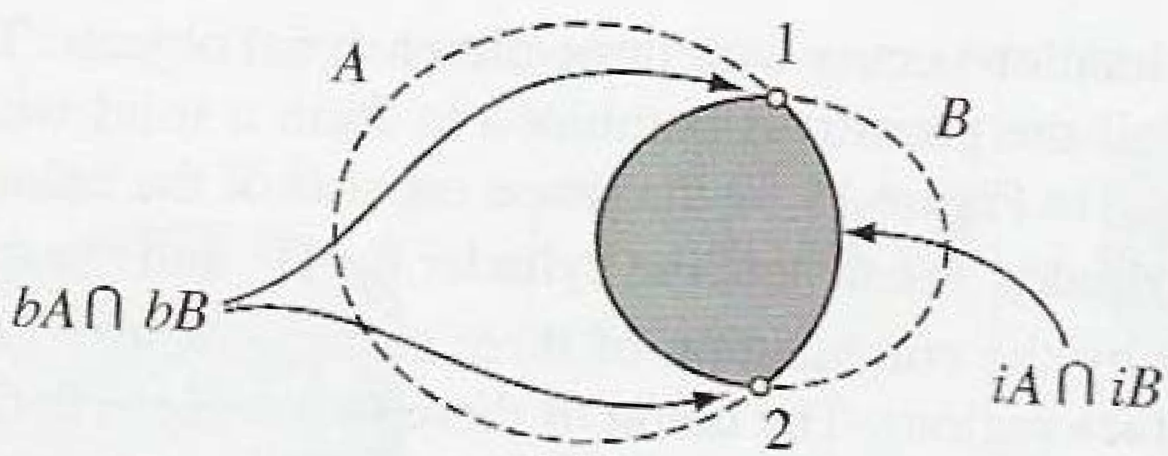


Figure 11.57 Boundary intersection.

Boundary Models (continued)

Boundary Representation (B-Rep)

0 value of parametric variable →

<u>2-step</u> *A* U *B*:

-Locate u_1, u_2

-Identify active parametric regions.

Include *C*.

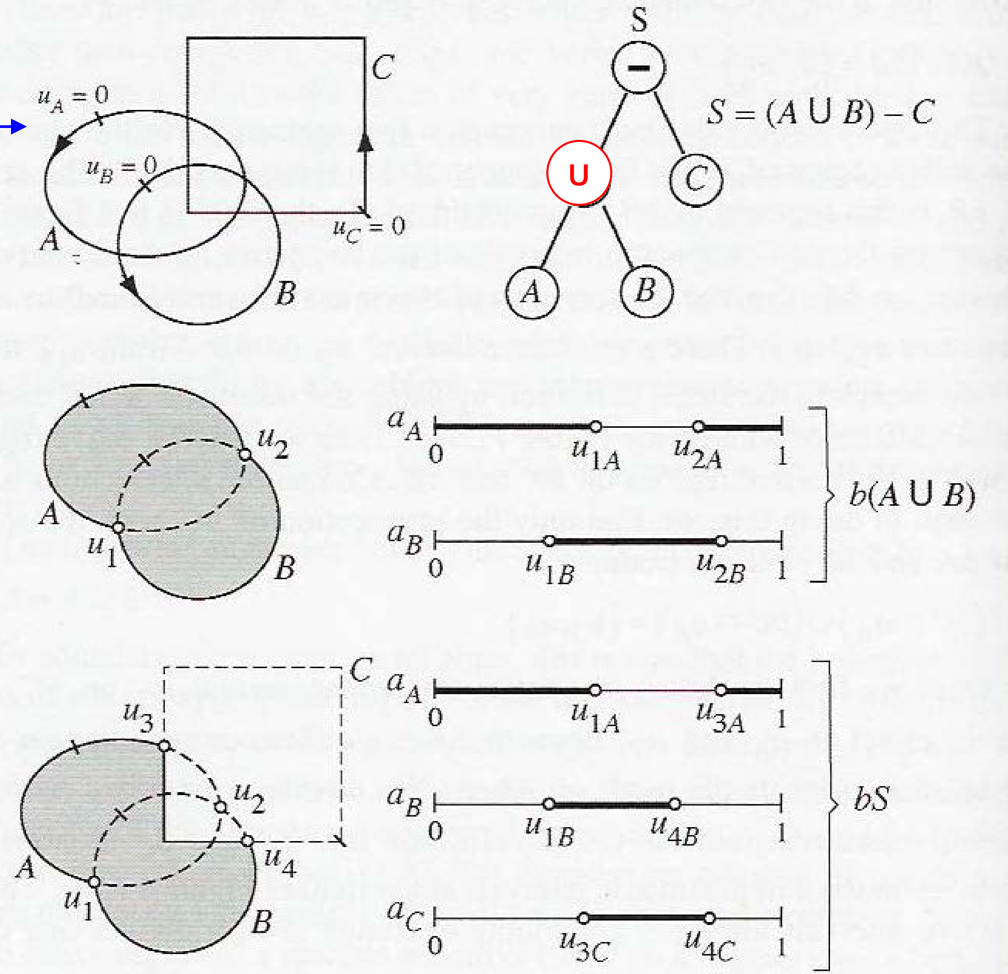
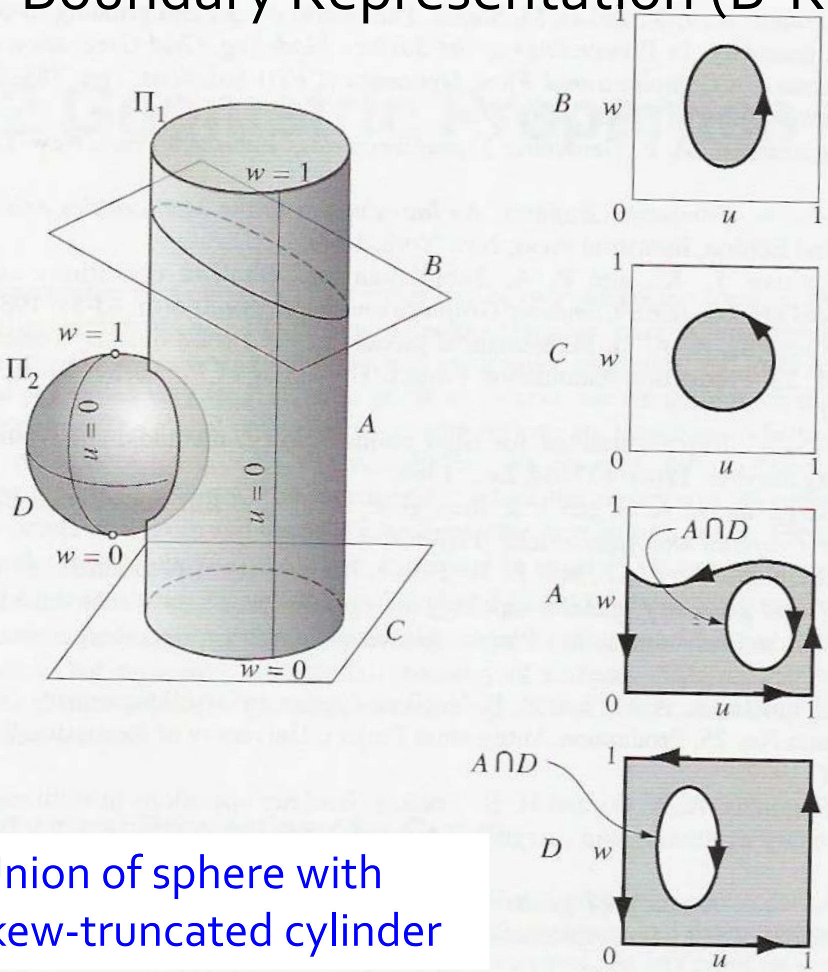


Figure 11.58 Two-dimensional boundary representation.

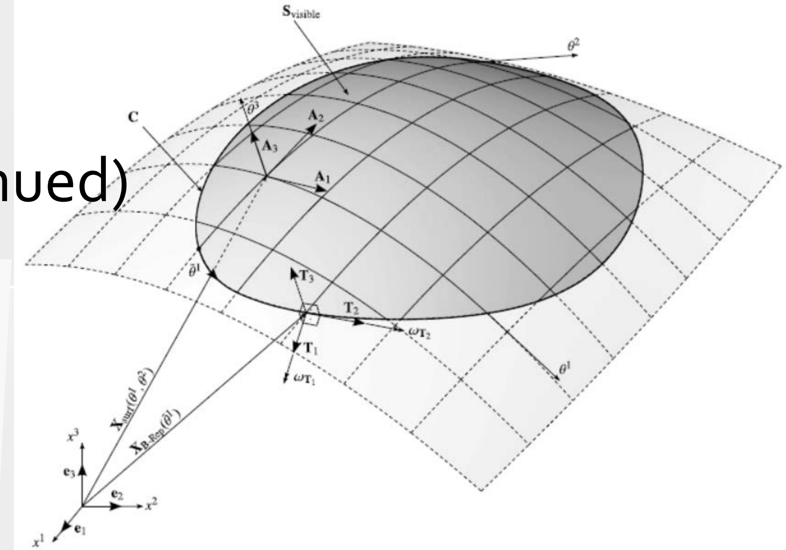
Boundary Models (continued)

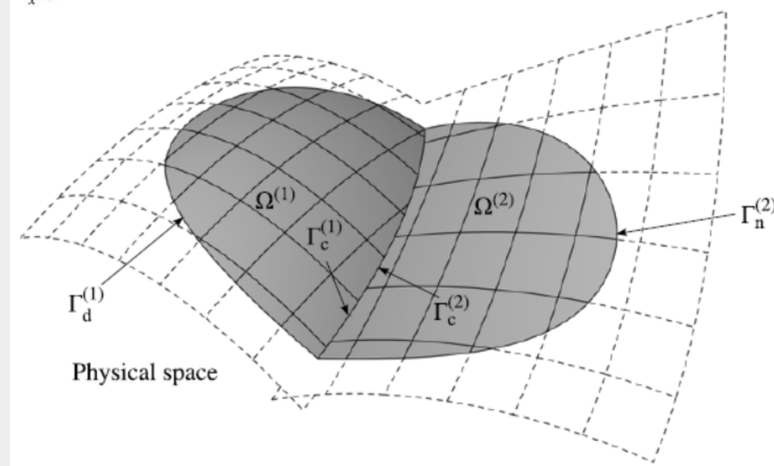
Boundary Representation (B-Rep)

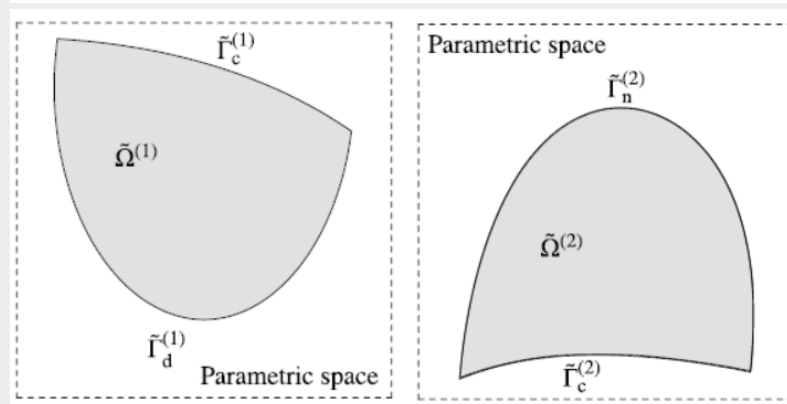


Union of sphere with skew-truncated cylinder

Figure 11.59 Three-dimensional boundary representation.







Introductory Notes on Geometric Aspects of Topology

PART I: Experiments in Topology

1964

Stephen Barr

(with some additional material from *Elementary Topology by Gemignani)*

PART II: Geometry and Topology for Mesh Generation

Combinatorial Topology

2006

Herbert Edelsbrunner

PART I: Experiments in Topology

What is Topology?

- Rooted in:
  - Geometry (our focus)
    - Topology here involves properties preserved by transformations called *homeomorphisms*.
  - Analysis: study of real and complex functions
    - Topology here involves abstractions of concepts generalized from analysis
      - Open sets, continuity, metric spaces, etc.
- Types of Topologists:
  - Point set topologists
  - Differential topologists
  - Algebraic topologists…

Source: Gemignani

Towards Topological Invariants

- Geometrical topologists work with properties of an object that survive distortion and stretching.
  - e.g. ordering of beads on a string is preserved
    - Substituting elastic for string
    - Tying string in knots

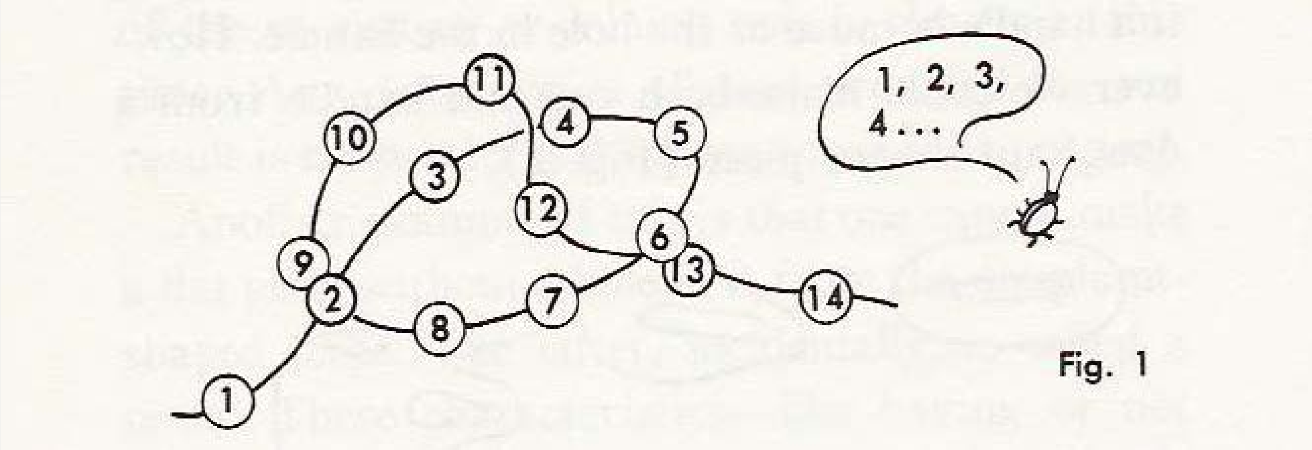
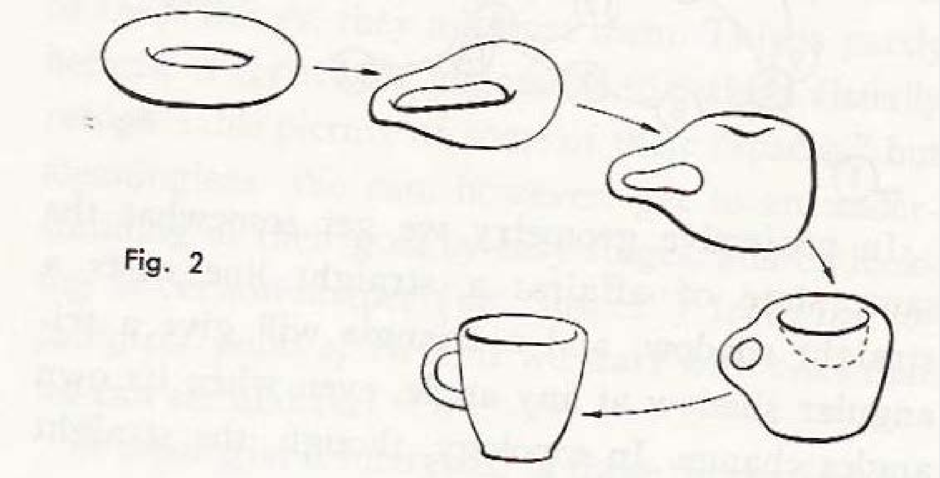


Fig. 1

Source: Barr

Towards Topological Invariants

- Distortions are allowed if you don't* *See caveat on next slide.*
  - disconnect what was connected
    - e.g. make a cut or a **hole** (or a "**handle**")
  - connect what was not connected
    - e.g. joining ends of previously unjoined string or filling in a hole



Legal **continuous bending and stretching** transformations of torus into cup.

Torus and cup are ***homeomorphic*** to each other.

Source: Barr

Towards Topological Invariants

Can make a break if we rejoin it afterwards in the same way as before.

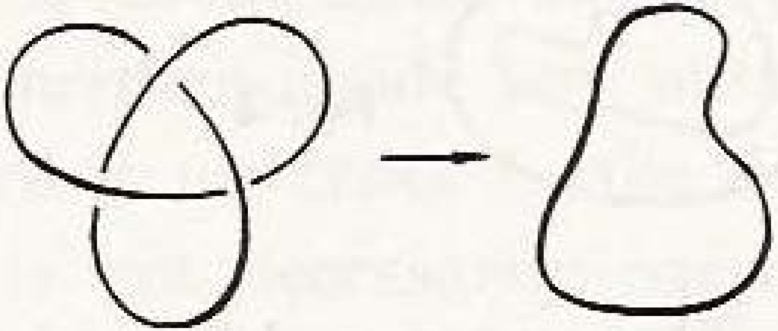


Fig. 3

Trefoil knot and curve are **homeomorphic** to each other.
They can be continuously deformed, via bending and stretching, into each other in 4-dimensional space*.

- *Barr states this as a conjecture; another source states is as a fact.*

Source: Barr

Connectivity

- Lump of clay is **simply connected**.
  - One piece
  - No holes
  - Any closed curve on it divides the whole surface into 2 parts*:
    - inside
    - outside

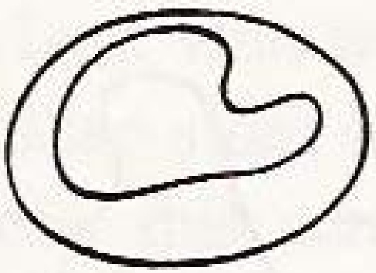


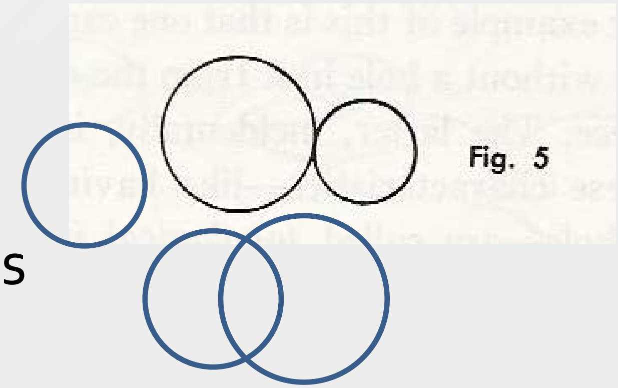
Fig. 4

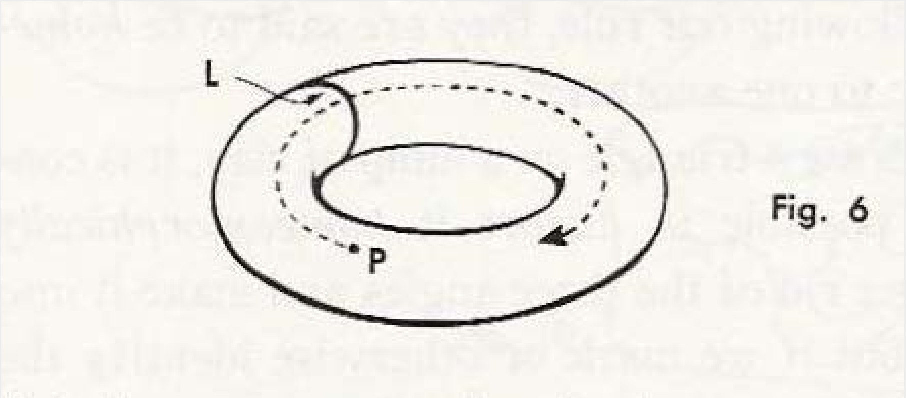
*_**Jordan Curve Theorem**_ is difficult to prove.

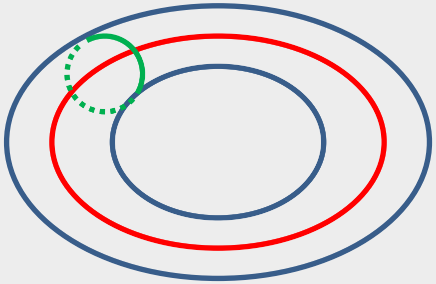
Source: Barr

Connectivity (continued)

- For 2 circles on simply connected surface, second circle is either
  - tangent to first circle
  - is disjoint from first circle
  - intersects first circle in 2 places
- For 2 circles on torus
  - line need not divide surface into 2 pieces
  - 2 circles can *cross* each other at <u>one</u> point







Source: Barr

Connectivity (continued)

- On a “lump of clay”, given a closed curve joined at two distinct points to another closed curve
  - Homeomorphism cannot change the fact that there are two joints.
    - No new joints can appear.
    - Neither joint can be removed.

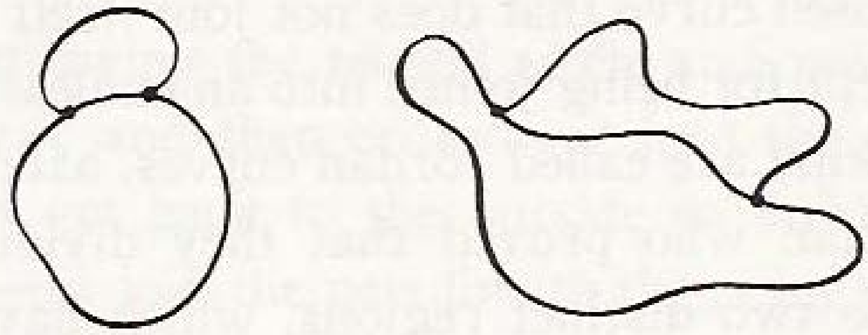
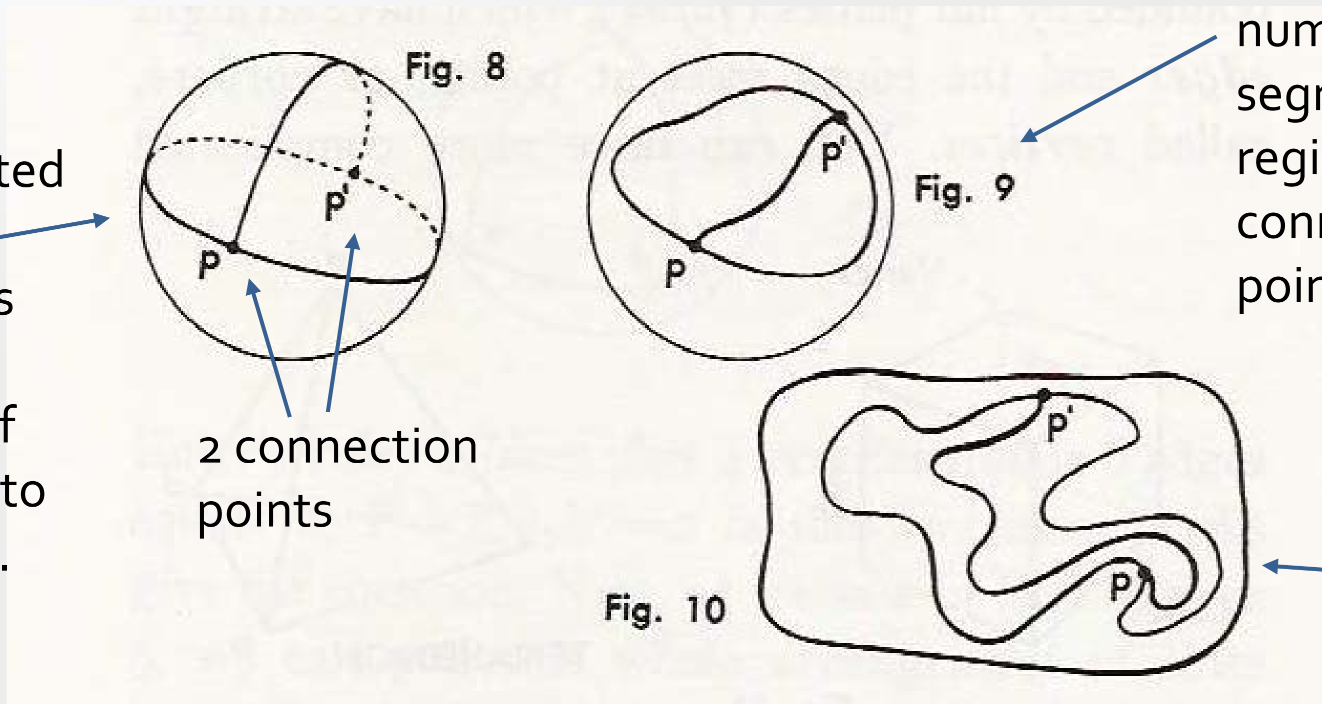


Fig. 7

Source: Barr

Connectivity (continued)

- Preserving topological entities:



3 connected curve segments partition surface of sphere into 3 regions.

2 connection points

"pulling" the curves onto this side preserves number of curve segments, regions, and connection points

further distortion preserves topological entities

Source: Barr

Revisiting Euler's Formula for Polyhedra

- $V - E + F = 2$

- Proof generalizes formula and shows it remains true under certain operations.
- Before the proof, verify formula for distorted embedding of **tetrahedron** onto **sphere**, which is a *simply connected surface*.

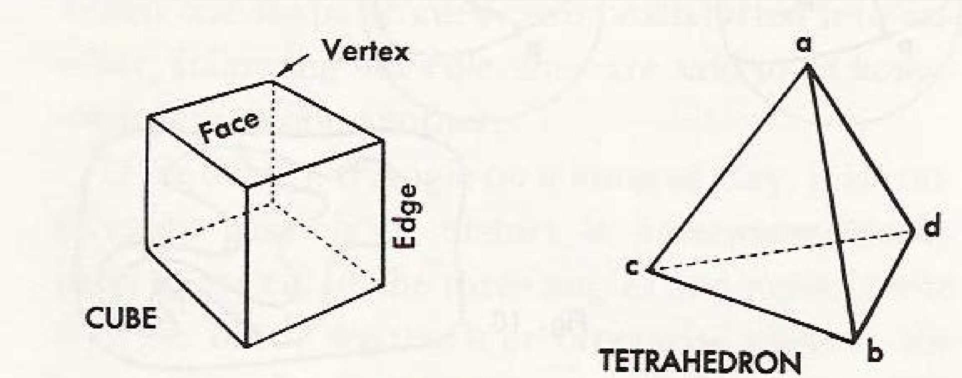


Fig. 11

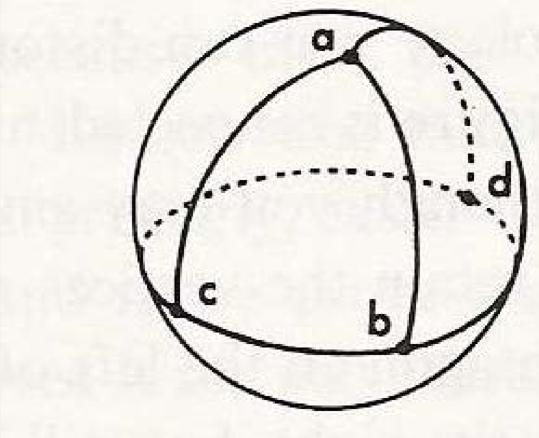


Fig. 12

Source: Barr

Revisiting Euler's Formula for Polyhedra
(continued)

- "Pull" arrangement of line segments around to front and verify formula.
- This gives us a vehicle for discussing operations on a drawing on a simply connected surface.
- Explore operations before giving the proof…

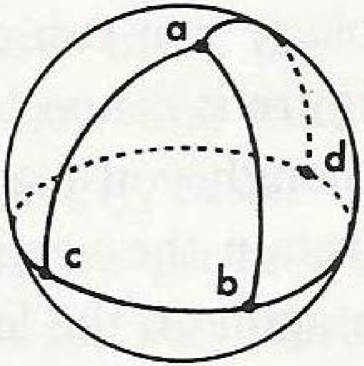


Fig. 12

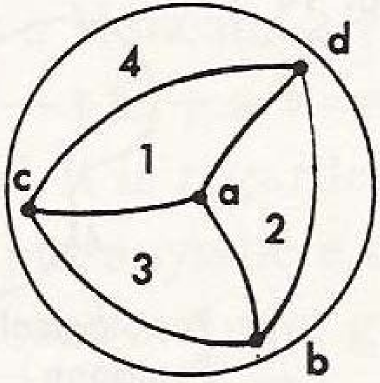
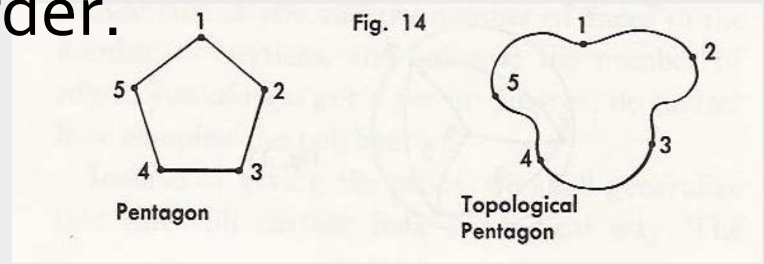


Fig. 13

Source: Barr

Revisiting Euler's Formula for Polyhedra
(continued)

- Operations must abide by rules:
  - Vertices must retain identity as marked points in same order.
  - **C^0** connectivity is preserved



  - Figure is drawn on a simply connected surface.
  - Every curve segment has a vertex
    - at its free end if there are any free ends
    - where it touches or crosses another curve segment
  - Any enclosure counts as a face.

Source: Barr

Revisiting Euler’s Formula for Polyhedra
(continued)

- For a single curve segment:
  - 1 unbounded face
  - 2 vertices
  - $V - E + F = 2 - 1 + 1 = 2$

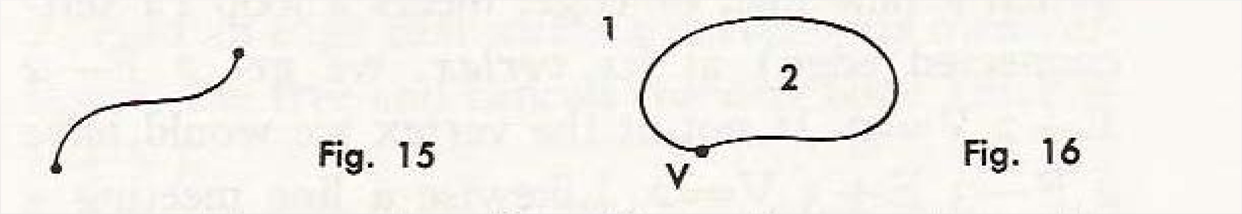


Fig. 15

Fig. 16

- Connecting the 2 ends preserves formula.

Source: Barr

Revisiting Euler's Formula for Polyhedra

(continued)

Also we can put any number of arbitrary vertices on an edge: and each would divide the line into new edges, giving, in Fig. 17, 1 F—4 E+5 V=2.

Alternatively, cross first line with another.

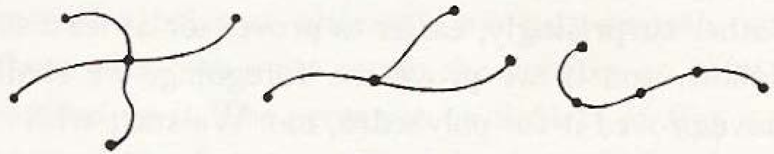


Fig. 17

When a new line, or edge, meets a loop (a self-connected edge) at its *vertex*, we get 2 F—2 E+2 V=2. If not at the vertex we would have 2 F—3 E+3 V=2. Likewise a line meeting a loop at 2 points gives 3 F—3 E+2 V=2 (Fig. 18).

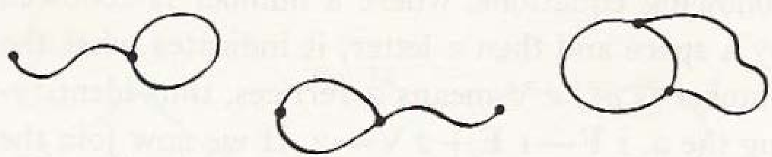


Fig. 18

The only way to obtain a new face is by adding at least one edge. Edge must either connect with both its ends or be itself a loop.

Source: Barr

Revisiting Euler's Formula for Polyhedra
(continued)

- Proof claims that the following 8 cases are exhaustive:

1. If we add a vertex to an edge between vertices, it divides it: making 1 edge into 2, thus it adds 1 E, canceling the new V, in the expression F—E+V.

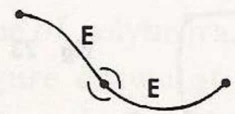


Fig. 19

2. Add an edge that meets a vertex—its own vertex on the free end cancels the new edge (in F—E+V).

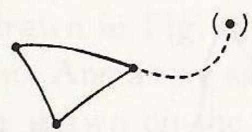


Fig. 20

3. Add an edge that meets an edge between vertices: it adds 2 E and 2 V (having divided the old edge). These cancel as before.

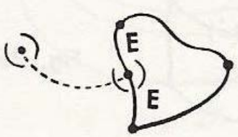


Fig. 21

Source: Barr

Revisiting Euler's Formula for Polyhedra
(continued)

4. Add an edge with each end meeting a vertex: it adds 1 F and 1 E (but no V) and they cancel.

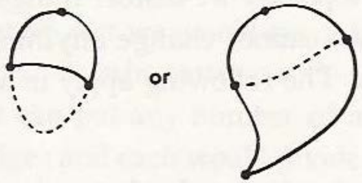


Fig. 22

5. Add an edge with both ends meeting the same V: it adds 1 F and 1 E, which cancel.

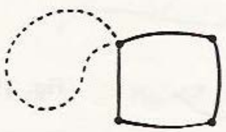


Fig. 23

6. Add an edge that meets 1 V and 1 E: it adds 1 F, 2 E, and 1 V, which cancel (1 F—2 E+1 V=0).

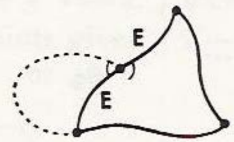


Fig. 24

7. Add an edge that meets 2 edges: it adds 1 F, 3 E, and 2 V, which cancel (1 F—3 E+2 V=0).

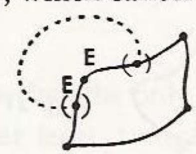


Fig. 25

8. Add an edge with both ends meeting at one V in one edge: it adds 1 F, 2 E, and 1 V, which cancel.

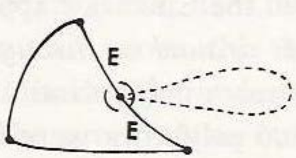


Fig. 26

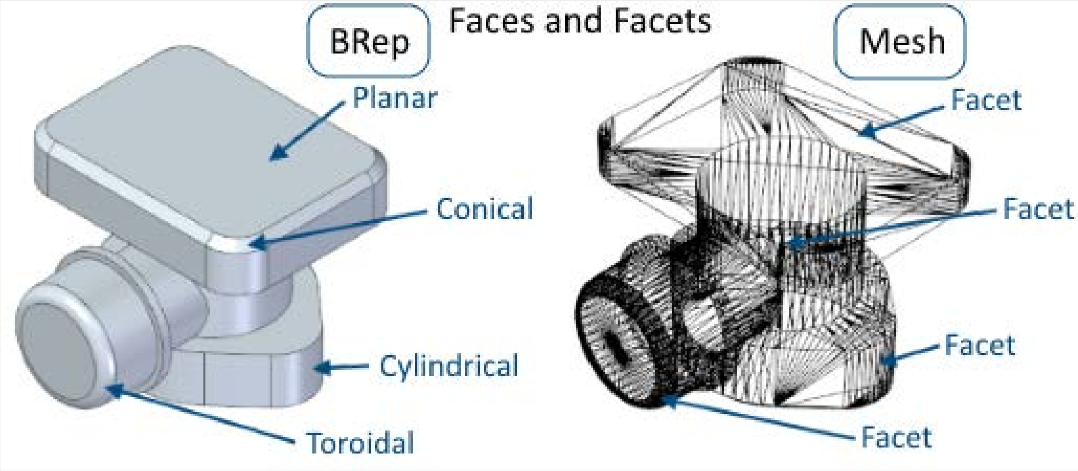
-These are all the legal ways of adding edges and vertices.
-Thus we can draw any such connected figure on a simply connected surface while preserving Euler's formula.
-Must also apply to polyhedra.

Source: Barr

PART II: Geometry and Topology for Mesh Generation

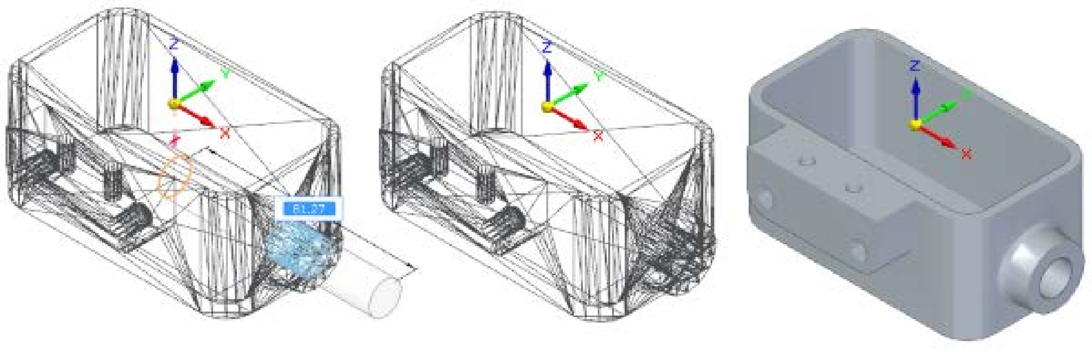
Combinatorial Topology

2006 *Herbert Edelsbrunner*



Goals

- Introduce standard topological language to facilitate triangulation and mesh dialogue.
- Understand space:
  - how it is connected;
  - how we can decompose it.
- Form bridge between continuous and discrete geometric concepts.
  - Discrete context is convenient for computation.



Parametric Hole in Mesh Geometry

Source: Edelsbrunner

Simplicial Complexes: Simplices

- Fundamental discrete representation of continuous space.
  - Generalize triangulation.
- Definitions:
  - Points are ***affinely independent*** if no affine space of dimension i contains more than i +1 of the points.
  - ***k-simplex*** is convex hull of a collection of k +1 affinely independent points.
  - Face of σ: $\sigma = \text{conv} S$

    $\tau \leq \sigma$

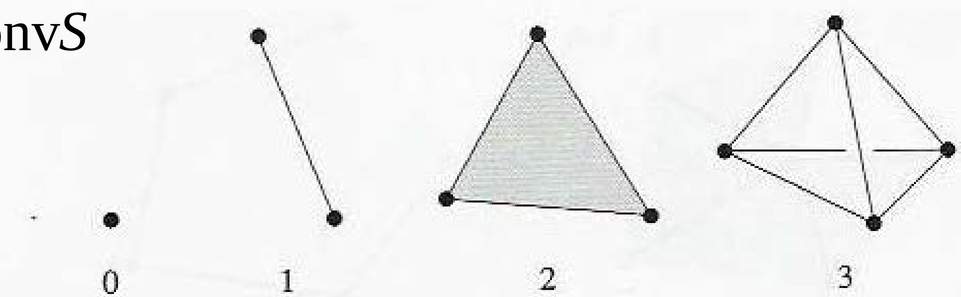


Figure 3.1. A 0-simplex is a point or vertex, a 1-simplex is an edge, a 2-simplex is a triangle, and a 3-simplex is a tetrahedron.

The 4 types of nonempty simplices in $\boldsymbol{R}^3$.

Source: Edelsbrunner

Simplicial Complexes

- Definition: A *simplicial complex* is collection of faces of a finite number of simplices, any 2 of which are either disjoint or meet in a common face.

$$i)\left(\sigma \in K\right)\wedge\left(\tau \le \sigma\right)\Rightarrow\left(\tau \in K\right)\text{and}$$
$$ii)\,\sigma,\upsilon \in K \Rightarrow \left(\sigma \cap \upsilon\right)\le \sigma,\upsilon$$

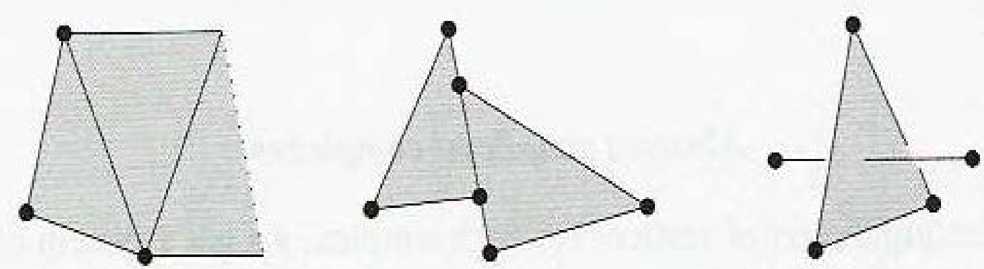


Figure 3.2. To the left, we are missing an edge and two vertices. In the middle, the triangles meet along a segment that is not an edge of either triangle. To the right, the edge crosses the triangle at an interior point.

Violations of the definition.

Source: Edelsbrunner

Simplicial Complexes: Stars and Links

- Use special subsets to discuss local structure of a simplicial complex.
- Definitions:
  - *Star* of a simplex τ consists of all simplices that contain τ.
  - *Link* consists of all faces of simplices in the star that don't intersect τ.

$$\mathrm{St}\,\tau = \{\sigma \in K \mid (\tau \le \sigma)\},$$
$$\mathrm{Lk}\,\tau = \{\sigma \in (\mathrm{ClSt}\,\tau) \mid \sigma \cap \tau \ne \emptyset\}$$

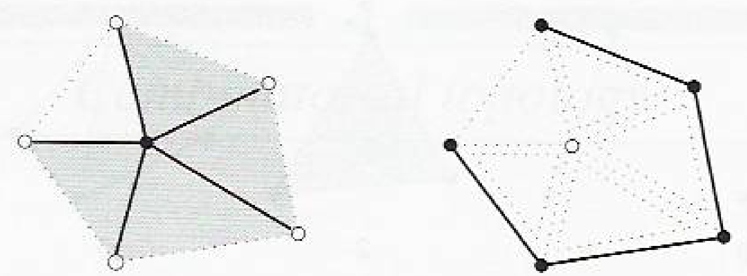


Figure 3.3. Star and link of a vertex. To the left, the solid edges and shaded triangles belong to the star of the solid vertex. To the right, the solid edges and vertices belong to the link of the hollow vertex.

Star is generally not closed. Link is always a simplicial complex.

Source: Edelsbrunner

Simplicial Complexes: Abstract Simplicial Complexes

- Eliminate geometry by substituting *set* of vertices for each simplex.
  - Focus on combinatorial structure.
- Definition: A finite system A of finite sets is an *abstract simplicial complex* if:

$$(\alpha \in A \text{ and } \beta \subseteq \alpha) \Rightarrow \beta \in A$$

Vert A is union of vertex sets.

A is subsystem of power set of Vert A.

A is a subcomplex of an n-simplex, where n+1 = card Vert A.

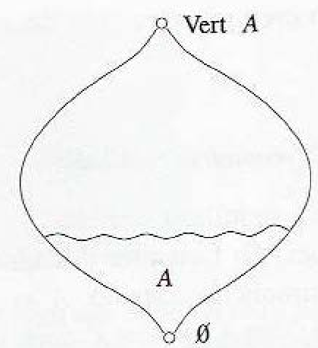


Figure 3.4. The onion is the power set of Vert A. The area below the waterline is an abstract simplicial complex.

Source: Edelsbrunner

Simplicial Complexes: Posets

- Definition: Set system with inclusion relation forms partially ordered set (poset), denoted: $(A, \subseteq)$
- Hasse diagram:
  - Sets are notes
  - Smaller sets are below larger ones
  - Inclusions are edges (implied includes not shown)

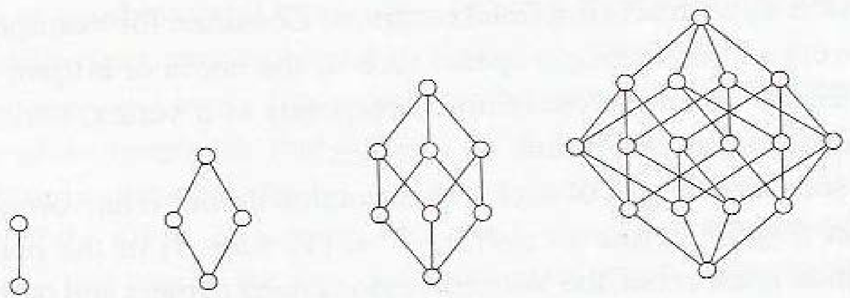


Figure 3.5. From left to right, the poset of a vertex, an edge, a triangle, and a tetrahedron.

Source: Edelsbrunner

Simplicial Complexes: Nerves

- One way to construct abstract simplicial complex uses <u>*nerve*</u> of arbitrary finite set C:

$$\mathrm{Nrv}\, C = \left\{ \alpha \subseteq C \mid \bigcap \alpha \neq \emptyset \right\}$$

$$\text{If } C = \beta \subseteq \alpha \text{ then } \bigcap \alpha \subseteq \bigcap \beta. \text{ Hence } \left(\alpha \in \mathrm{Nrv}\, C \right) \Rightarrow \left(\beta \in \mathrm{Nrv}\, C \right)$$

Nerve is therefore an abstract simplicial complex.

<u>*Example:*</u>
C is union of elliptical regions.
Each set in covering corresponds to a vertex.
k+1 sets with nonempty intersection define a k-simplex.

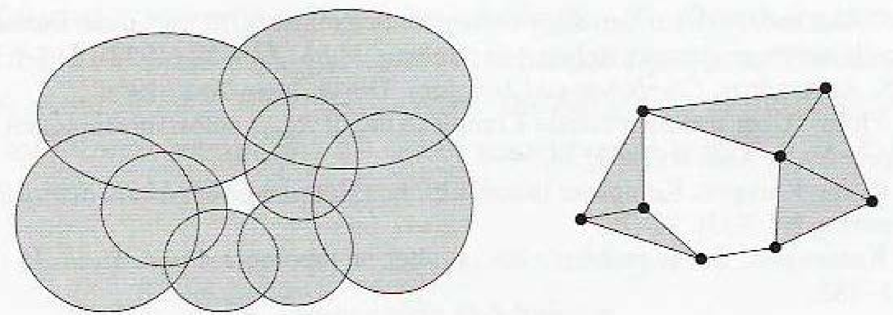


Figure 3.6. A covering with eight sets to the left and a geometric realization of its nerve to the right. The sets meet in triplets but not in quadruplets, which implies that the nerve is two dimensional.

Source: Edelsbrunner

Subdivision: Barycentric Coordinates

- Two ways to refine complexes by decomposing simplices into smaller pieces are introduced later.
- Both ways rely on ***barycentric coordinates***.
  - Non-negative coefficients γ_i such that $x = \Sigma_i\, \gamma_i p_i$. $\Sigma_i\, \gamma_i = 1$

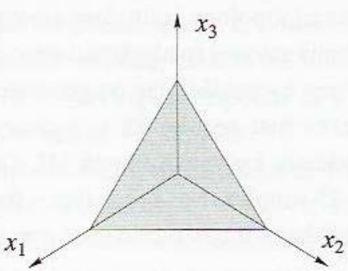


Figure 3.7. The standard triangle connects points (1, 0, 0), (0, 1, 0), and (0, 0, 1).

Standard k-simplex = convex hull of endpoints of k+1 unit vectors.

Barycenter (centroid) : all barycentric coordinates = 1/(k+1)

Source: Edelsbrunner

Subdivision:
Barycentric Subdivision

- Subdivision connecting barycenters of simplices.
- Example:

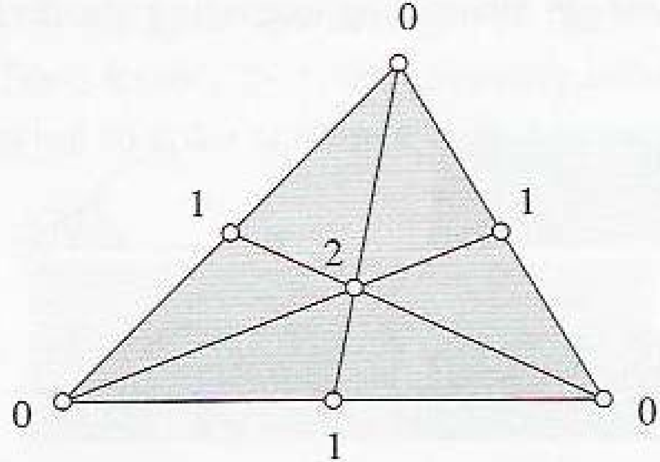


Figure 3.8. Barycentric subdivision of a triangle. Each barycenter is labeled with the dimension of the corresponding face of the triangle.

Source: Edelsbrunner

Subdivision: Dividing an Interval

- Barycentric subdivision can have unattractive numerical behavior.
- Alternative: try to preserve angles.
  - Distinguish different ways to divide [0,1]:
    - (k+1)-division associates point x with division of [0,1] into pieces of lengths γ_1, γ_2, …, γ_k

Cut [0,1] into 2 halves:

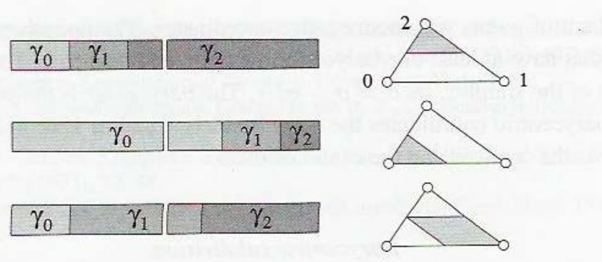


Figure 3.9. Three generic 3-divisions.

$$\gamma_2 \geq \frac{1}{2}$$

$$\gamma_0 \geq \frac{1}{2}$$

$$\gamma_0, \gamma_2 \leq \frac{1}{2}$$

Subdividing the rhombus: 2 cases for dividing line of γ_2 with respect to separator of γ_0 from γ_1.

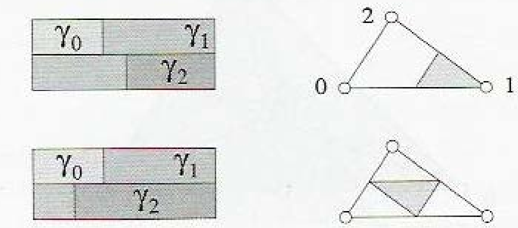


Figure 3.10. Two pairs of generic 2-divisions.

Source: Edelsbrunner

Subdivision:
Edgewise Subdivision

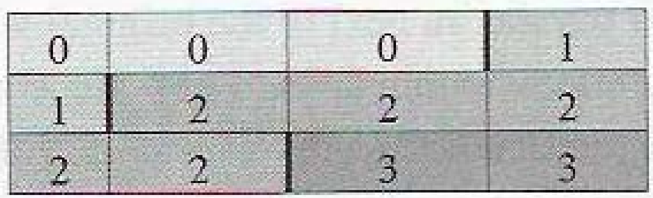


Figure 3.11. Stack of 4-division, cut into three equal intervals.

Source: Edelsbrunner

Subdivision: Edgewise Subdivision

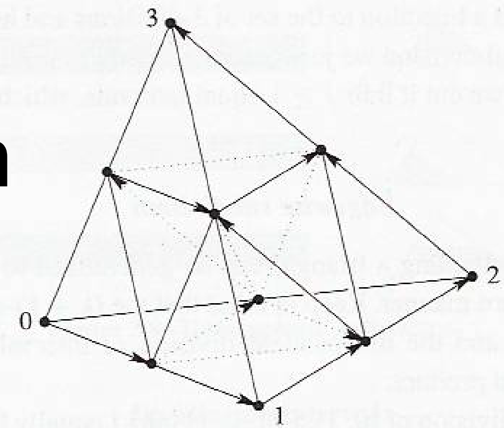


Figure 3.12. 8-division of a tetrahedron with shape vectors indicated by arrowheads.

Example

Consider the edgewise subdivision of a tetrahedron for $j = 2$. There are eight generic color schemes, namely

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 3 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 2 & 2 & 3 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & 1 & 2 \\ 2 & 3 & 3 & 3 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 1 & 2 & 2 \\ 2 & 2 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 2 & 3 \\ 3 & 3 & 3 & 3 \end{bmatrix}.$$

They divide the tetrahedron into four tetrahedra near the vertices and four tetrahedra dividing the remaining octahedron, as shown in Figure 3.12. Note that the way the tetrahedron is subdivided depends on the ordering of the four original vertices. The distinguishing feature is the diagonal of the octahedron used in the subdivision. It corresponds to the two-by-two color scheme with colors 0, 1, 2, 3. The diagonal is therefore the edge connecting the midpoints of $p_0 p_2$ and $p_1 p_3$.

Source: Edelsbrunner

Topological Spaces: Topology

- Topological notion of space (from *point set topology*)
  - and important special case of manifolds
- Definition: A *topological space* is a point set $\mathbf{X}$ together with a system X of subsets $A \subseteq \mathbf{X}$ that satisfies:

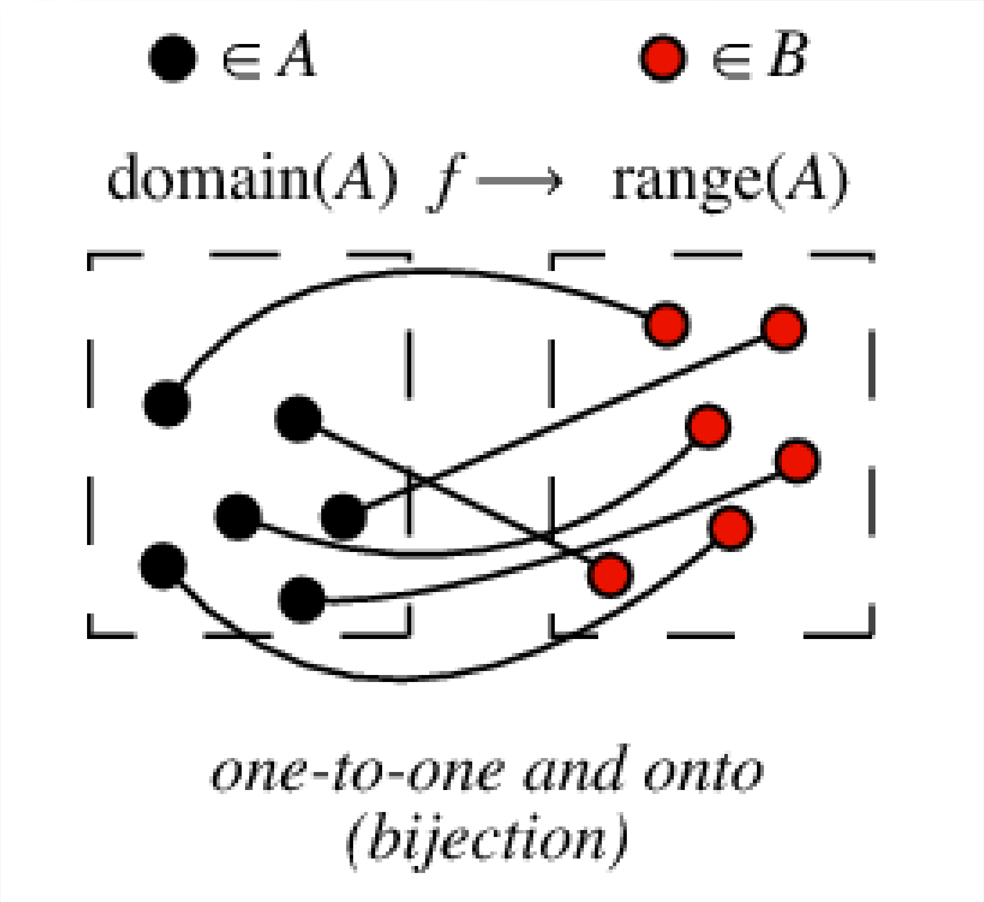
i.
ii. $\emptyset, \mathbf{X} \subseteq X$
iii. $Z \subseteq X \Rightarrow \bigcup Z \in X$

$$Z \subseteq X \text{ and } Z \text{ finite} \Rightarrow \bigcap Z \in X$$

- System X is a *topology*.
  - Its sets are the *open sets* in $\mathbf{X}$.
- Example: d-dimensional Euclidean space: $\mathbf{R}^d$.
  - Use Euclidean distance to define open ball as set of all points closer than some given distance from a given point.
  - Topology of $\mathbf{R}^d$ is the system of open sets, where each open set is a union of open balls.

Source: Edelsbrunner

Bijection (review)



Source: Wolfram MathWorld

Topological Spaces: Homeomorphisms

- Topological spaces are considered same or of same type if they are connected in same way.
- Homeomorphism is a function $f : \mathbf{X} \rightarrow \mathbf{Y}$ that is bijective, continuous, and has a continuous inverse.
  - "Continuous" in this context: preimage of every open set is open.
- If homeomorphism exists, then $\mathbf{X}$ and $\mathbf{Y}$ are homeomorphic:
  - Equivalence relation: $\mathbf{X}$ and $\mathbf{Y}$ are topologically equivalent:

$$\mathbf{X} \approx \mathbf{Y}$$

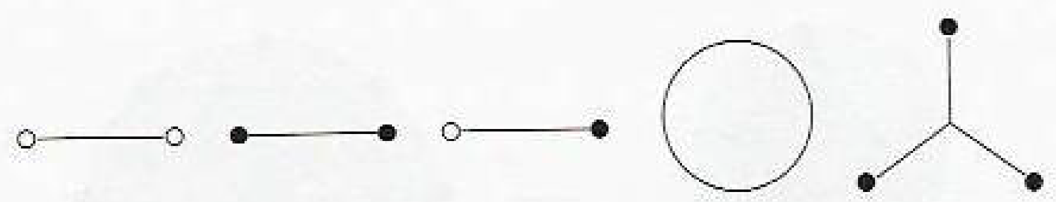


Figure 3.13. From left to right, the open interval, the closed interval, the half-open interval, the circle, a bifurcation.

Source: Edelsbrunner

Topological Spaces: Triangulation

- Typically a simplicial complex
- Polyhedron in $\mathbf{R}^d$ is the *underlying space* of a simplicial complex.
- *Triangulation* of a topological space $\mathbf{X}$ is a simplicial complex whose underlying space is homeomorphic to $\mathbf{X}$.

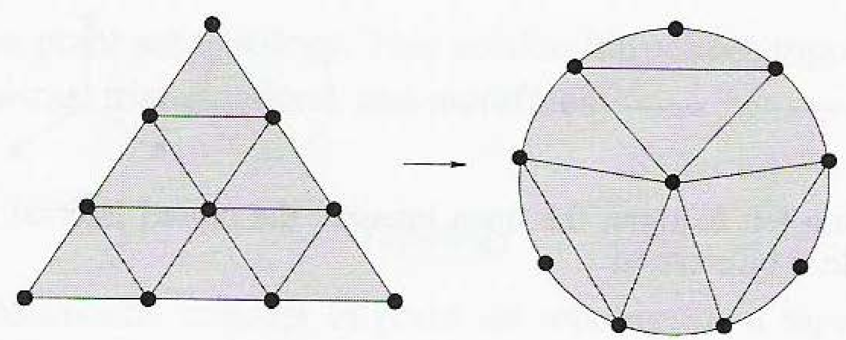


Figure 3.14. Triangulation of the closed disk. The homeomorphism maps each vertex, edge, and triangle to a homeomorphic subset of the disk.

Source: Edelsbrunner

Topological Spaces: Manifolds

- Defined locally:
  - Neighborhood of point $x \in X$ is an open set containing *x*.
  - Topological space $\mathbf{X}$ is a *k*-manifold if every $x \in X$ has a neighborhood homeomorphic to $\mathbf{R}^k$.
- Examples:
  - *k*-sphere: $\mathbf{S}^k = \{x \in \mathbf{R}^{k+1} \mid \|x\| = 1\}$

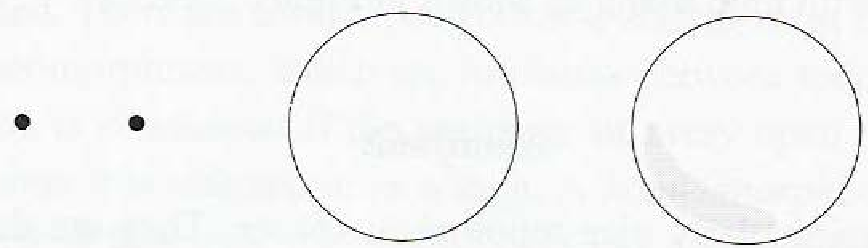


Figure 3.15. The 0-sphere is a pair of points, the 1-sphere is a circle, and the 2-sphere is what we usually call a sphere.

Source: Edelsbrunner

Topological Spaces: Manifolds with Boundary

- Now allow 2 types of neighborhoods to obtain more general class of spaces:
  - 2nd type is half an open ball: $\mathbf{H}^k = \{x = (x_1, x_2, \ldots x_k) \in \mathbf{R}^k \mid x_1 \geq 0\}$
- Space $\mathbf{X}$ is a k-manifold with boundary if every point $x \in X$ has a neighborhood homeomorphic to $\mathbf{R}^k$ or to $\mathbf{H}^k$.
  - Boundary is set of points with a neighborhood homeomorphic to $\mathbf{H}^k$.
- Examples:
  - k-ball:

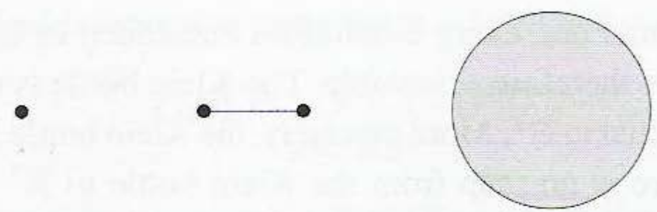


Figure 3.16. The 0-ball is a point, the 1-ball is a closed interval, and the 2-ball is a closed disk.

Source: Edelsbrunner

Topological Spaces: Orientability

- Global property.
- Envision (k+1)-dimensional ant walking on k-manifold.
  - At each moment ant is on one side of local neighborhood it is in contact with.
  - Manifold is nonorientable if there's a walk that brings ant back to same neighborhood, but on the other side.
  - It is orientable if no such path exists.
- Orientable examples:
  - Manifold: k-sphere
  - Manifold with boundary: k-ball
- Nonorientable examples

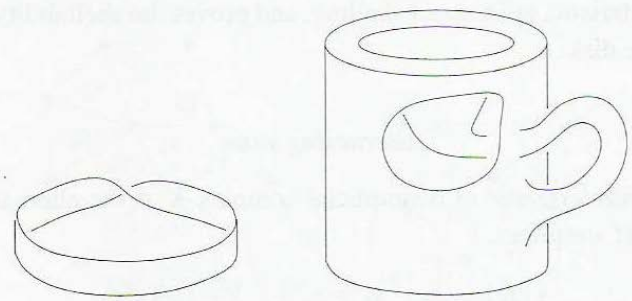


Figure 3.17. The Möbius strip to the left is bounded by a single circle. The Klein mug to the right is drawn with a cutaway view to show a piece of the handle after it passes through the surface of the mug.

Source: Edelsbrunner

Euler Characteristic: Alternating Sums

Euler characteristic of a triangulated space.

Euler Characteristic: Shelling

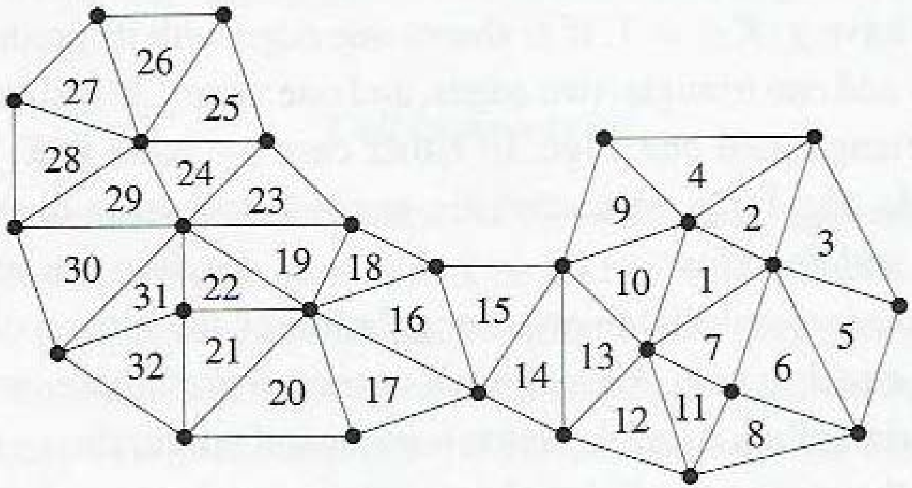


Figure 3.18. The numbers specify a shelling of the triangulation.

Source: Edelsbrunner

Euler Characteristic: Shelling

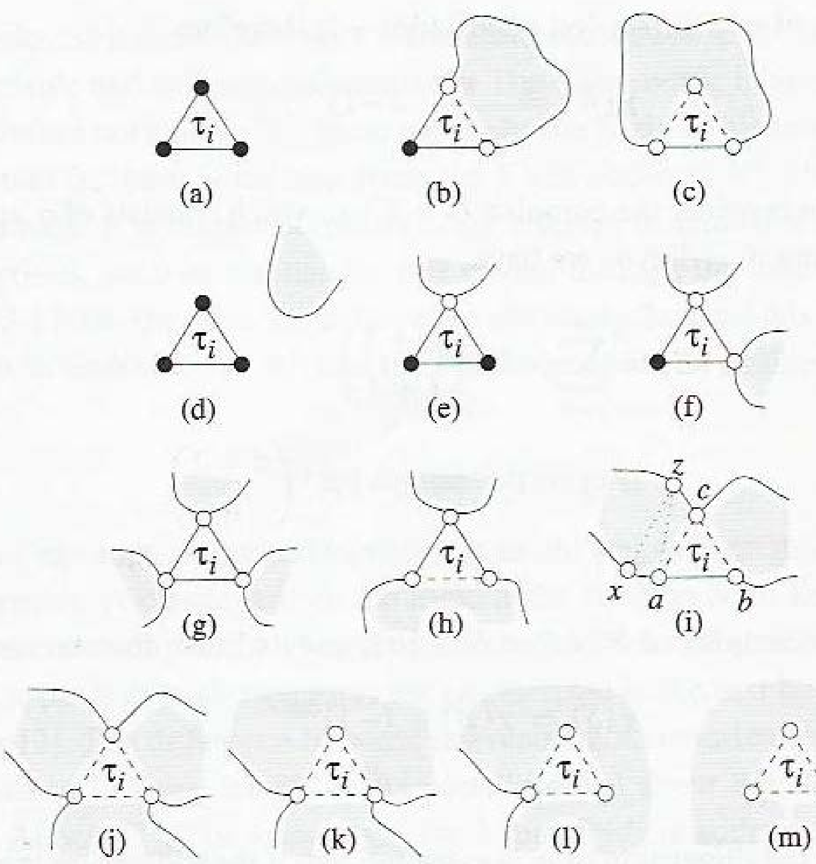


Figure 3.19. The 13 ways a triangle can intersect with the complex of its predecessors. Only cases (a), (b), and (c) occur in a shelling.

Source: Edelsbrunner

Euler Characteristic: Cell Complexes

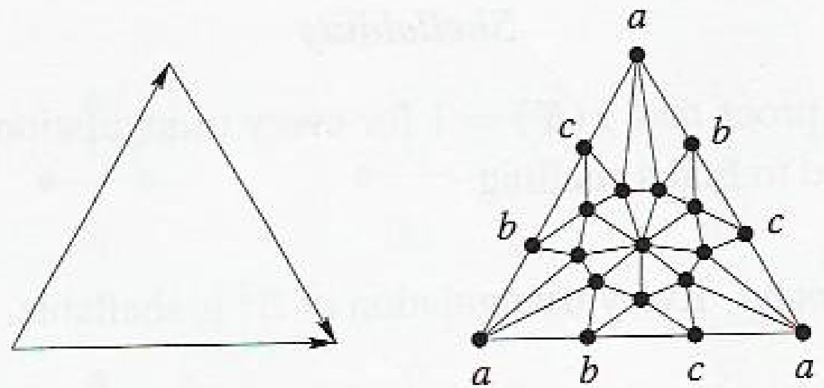


Figure 3.20. The dunce cap to the left consists of one 2-cell, one edge, and one vertex. Its triangulation to the right consists of 27 triangles, 39 edges, and 13 vertices.

Source: Edelsbrunner

Euler Characteristic: 2-Manifolds

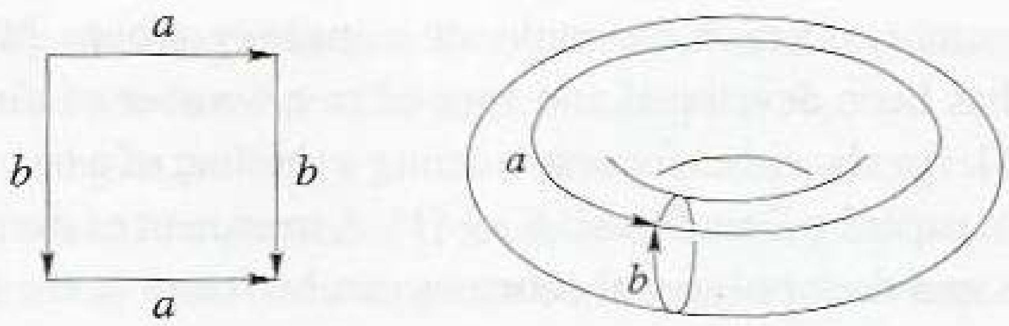


Figure 3.21. Edges with the same label are glued so their arrows agree. After gluing we have two edges and one vertex.

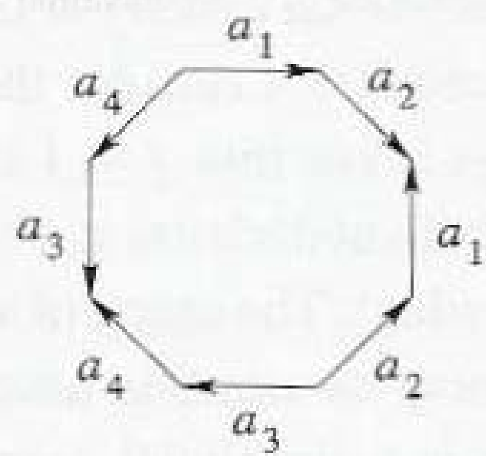


Figure 3.22. The polygonal schema of the double torus.

Source: Edelsbrunner

Parasolid 3D Geometric Modeling

3D geometric modeling needs continuous innovation
to meet the requirements of
additive manufacturing,
generative design,
and other cutting-edge design
and manufacturing techniques.

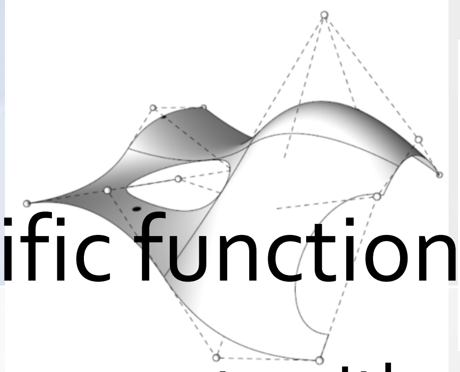
At the same time, geometric modelers should take advantage of new and improved computing environments.

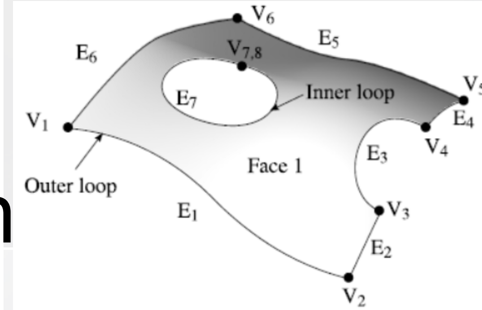
Parasolid v30.0
3D Geometric Modeling Engine

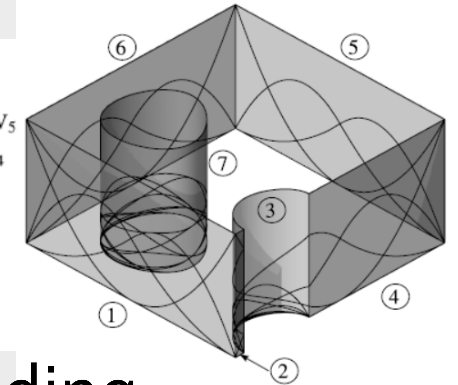
New version extends **classic B-rep** and **facet B-rep modeling** towards realizing the full power of Convergent Modeling Parasolid v30.0 delivers enhancements to classic B-rep to enable application developers to deliver sophisticated functionality more effectively to their end-users.

Deformation of Mesh Faces

Mesh-specific function



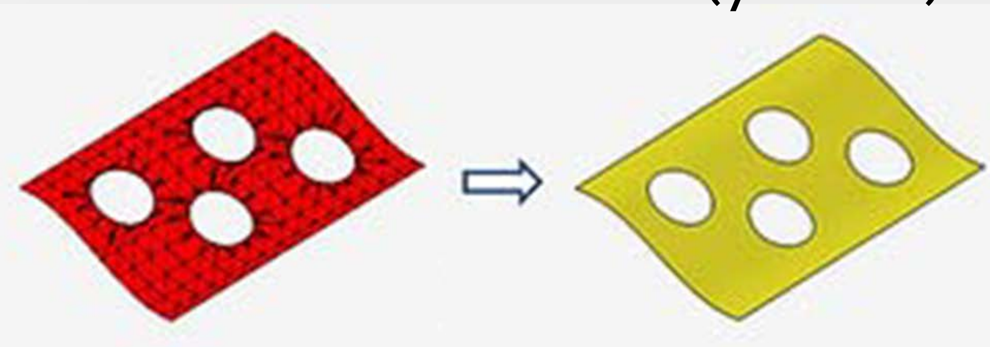


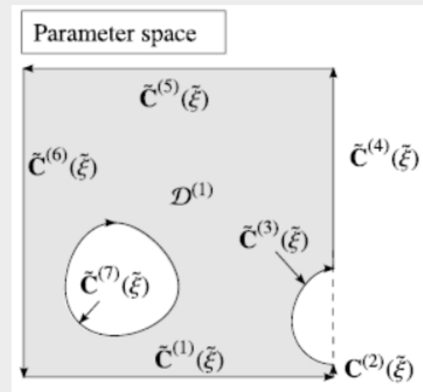


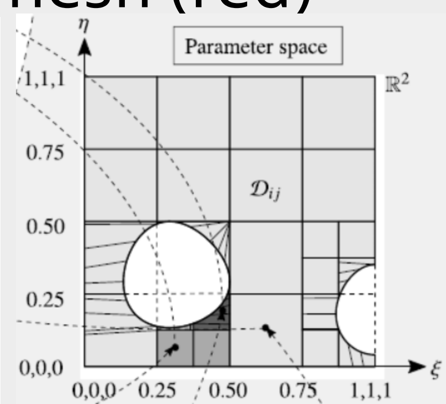
Several enhancements with mesh data including :

- Added mesh enquiry functions and identification of subsets of a mesh.
- Creation of trimmed surfaces from a mesh and generation of polylines from isoclines.
- Improved control over repair of mesh foldovers.
- Improved performance of mesh-based operations.

A trimmed surface (yellow) created from a mesh (red)







Facet B-rep modeling

Facet related tools enhancements have been provided.

- Creation of edge blends for facet models.
- Addition of direct modeling operations for deform, offset and replace of mesh faces.
- Creation of B-curves from polylines and finding chains of smoothly connected edges.
- Identification and deletion of redundant topologies and copying of construction and orphan geometry.
- Calculation of the minimum distance between classic B-rep models and facet B-rep models.

Parasolid v30.0

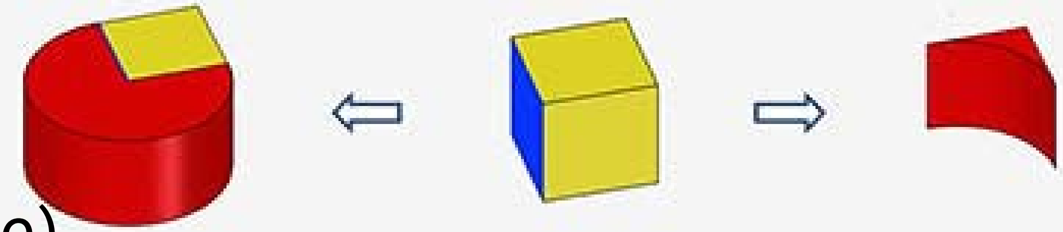
Facet B-rep enhancements cover modeling with facets and imported facet data repair, model editing.

All Parasolid operations in future releases will support models containing arbitrary combinations of **classic B-rep geometry** and **facet B-rep geometry**.

Enhancements have been added to classic B-rep blending and Boolean operations

Rotational transform:

- Improved control over the direction of rotational transforms in order to add or remove material.
- Increased body tapering operations
- Improved the accuracy of minimum radii calculations on B-surfaces.
- Improved detection of clashes in mirror transforms of topologies.



Rotating a face (Blue)
to either add material (Left) or remove material (Right)

B-rep blending and Boolean operations

- Trimmed solution on a periodic surface blend.
- Identification of underlying surfaces that have curvature similar to an edge blend being applied.
- Improved behavior when topology tolerances are involved in Boolean auto-matching operations.
- Imprinting and merging on complex grid-like faces.