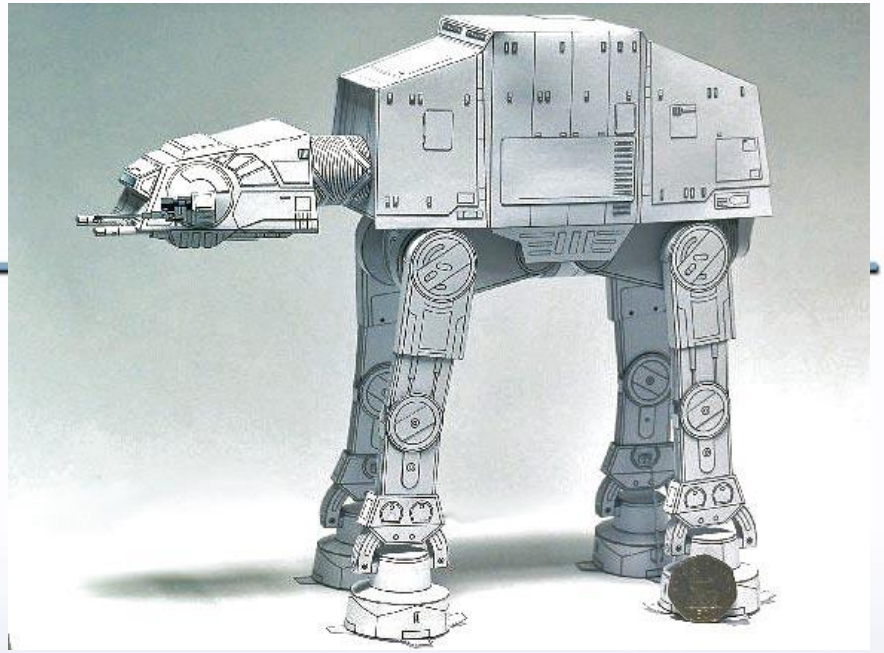
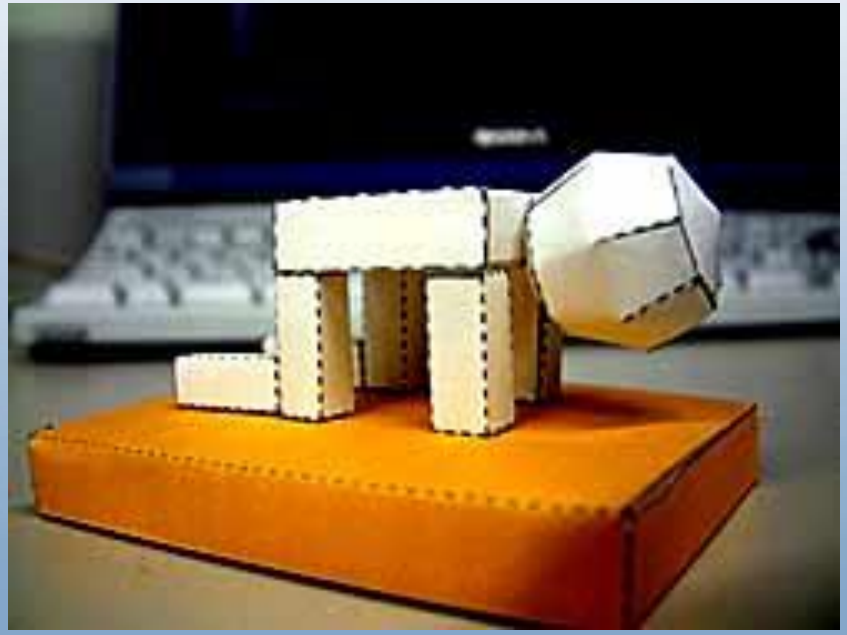
BRep and CSG





Contents

- Boundary Representation BRep
- Euler's formula
- Winged-edge structure
- Euler operators
- CSG tree
- Point-membership classification algorithm
- References

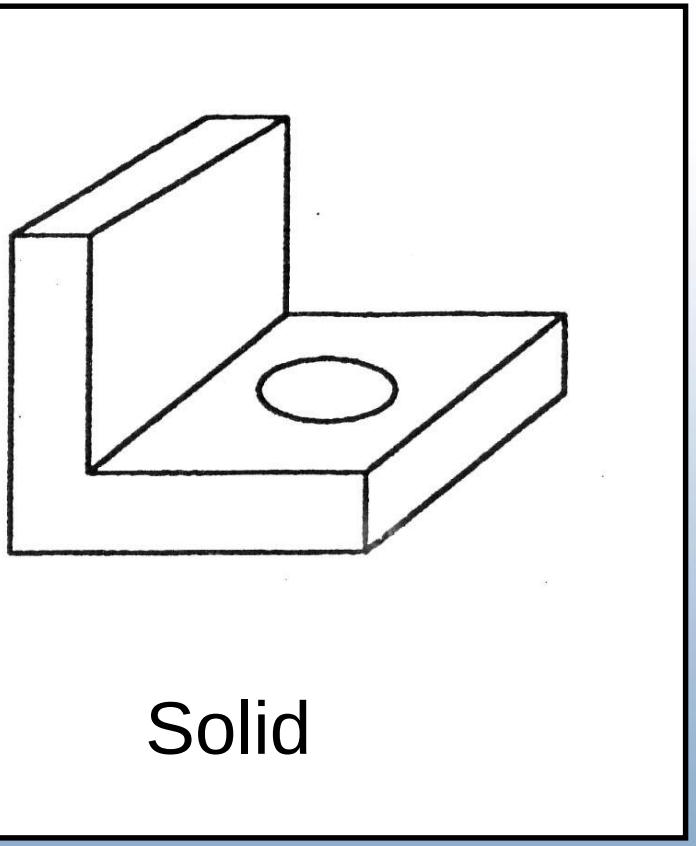
BRep notions

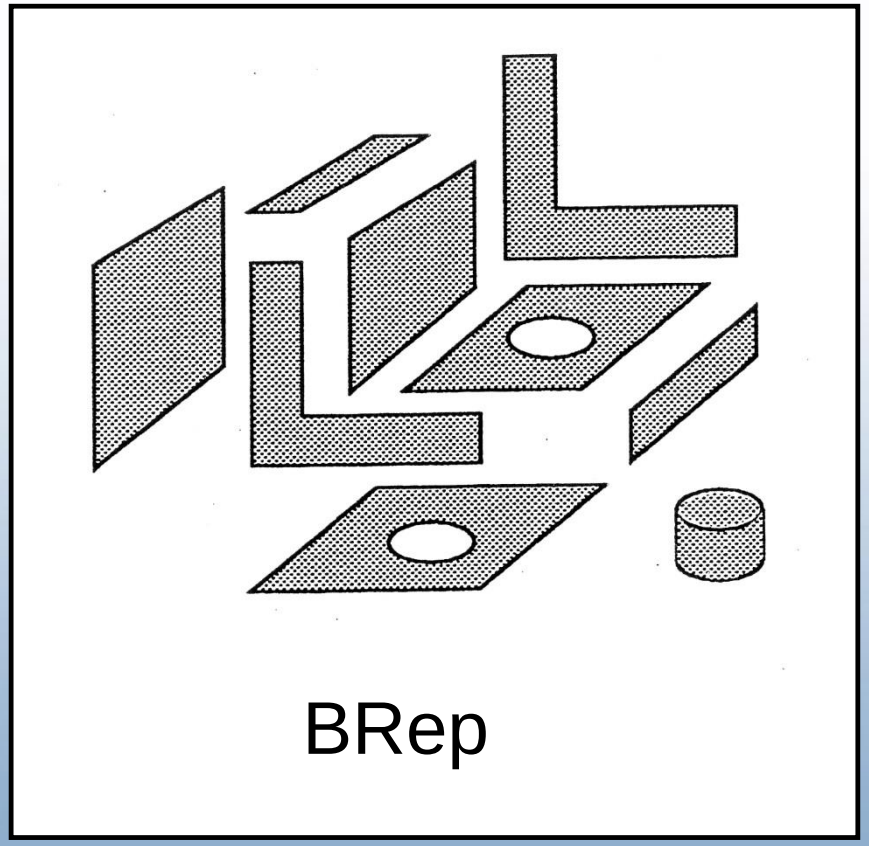
In the Boundary Representation BRep, a solid is represented by segmenting its boundary into a finite number of bounded subsets usually called "*faces*" or "*patches*", and representing each face by its bounding *edges* and *vertices*.

BRep notions

- This description has two parts, a topological description of the connectivity and orientation of vertices, edges, and faces in the form of a graph, and a geometric description for embedding these surface elements in space.
- The topological description specifies vertices, edges, and faces abstractly, and indicates their incidences and adjacencies.
- The geometric description specifies, for example, the coordinates of vertices or the equations of the surfaces containing the faces.

BRep example





Requicha, IEEE CG&A, 1983, p.26

BRep structure

Historically, BRrep evolved from a description of polyhedra in computer graphics:

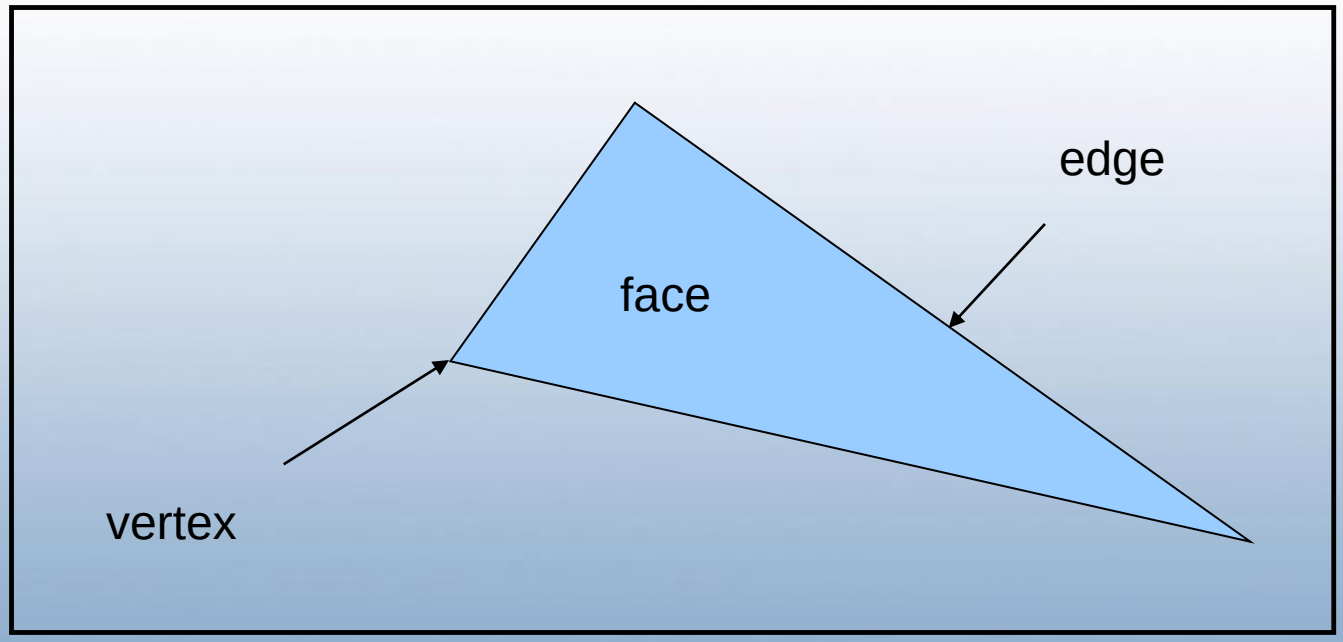
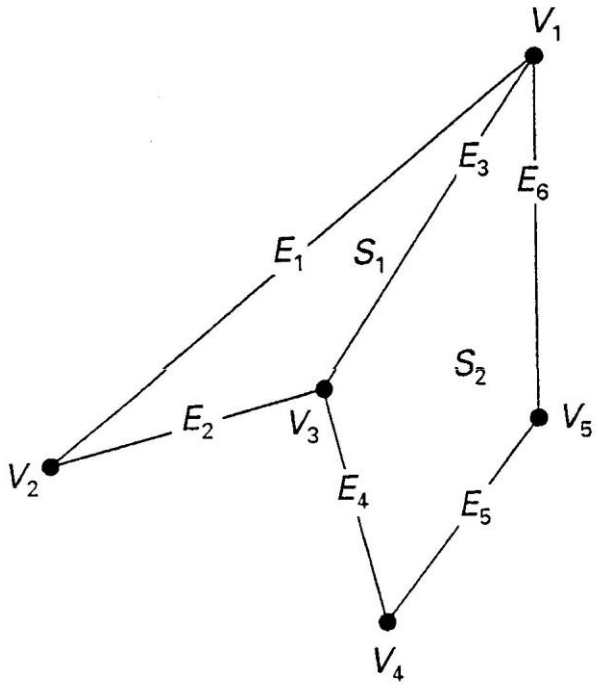


Fig. vertex-edge-face from CG course

BRep structure



EDGE TABLE	
E_1:	V_1, V_2
E_2:	V_2, V_3
E_3:	V_3, V_1
E_4:	V_3, V_4
E_5:	V_4, V_5
E_6:	V_5, V_1

POLYGON-SURFACE TABLE	
S_1:	E_1, E_2, E_3
S_2:	E_3, E_4, E_5, E_6

VERTEX TABLE	
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

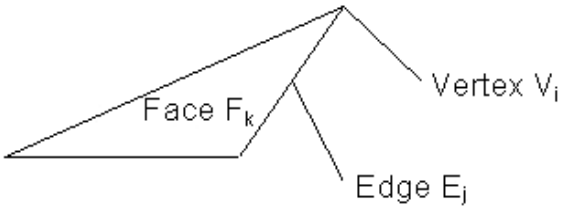
Hearn/ Baker, p.307

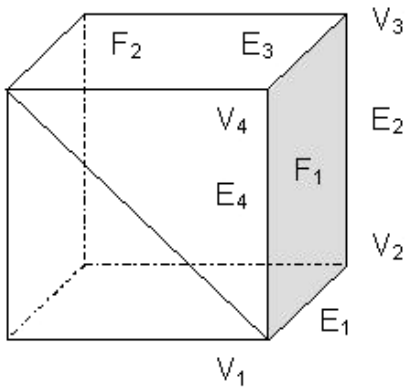
BRep and Graphs

- Planar graph can be drawn on a plane with no intersection between edges
- Planar graph can be drawn (embedded in) on a sphere or any other genus 0 surface (without through holes) with no edge intersections
- Any graph can be embedded in some surface with genus G (number of through holes)
- BRep can be considered a graph embedded in a surface

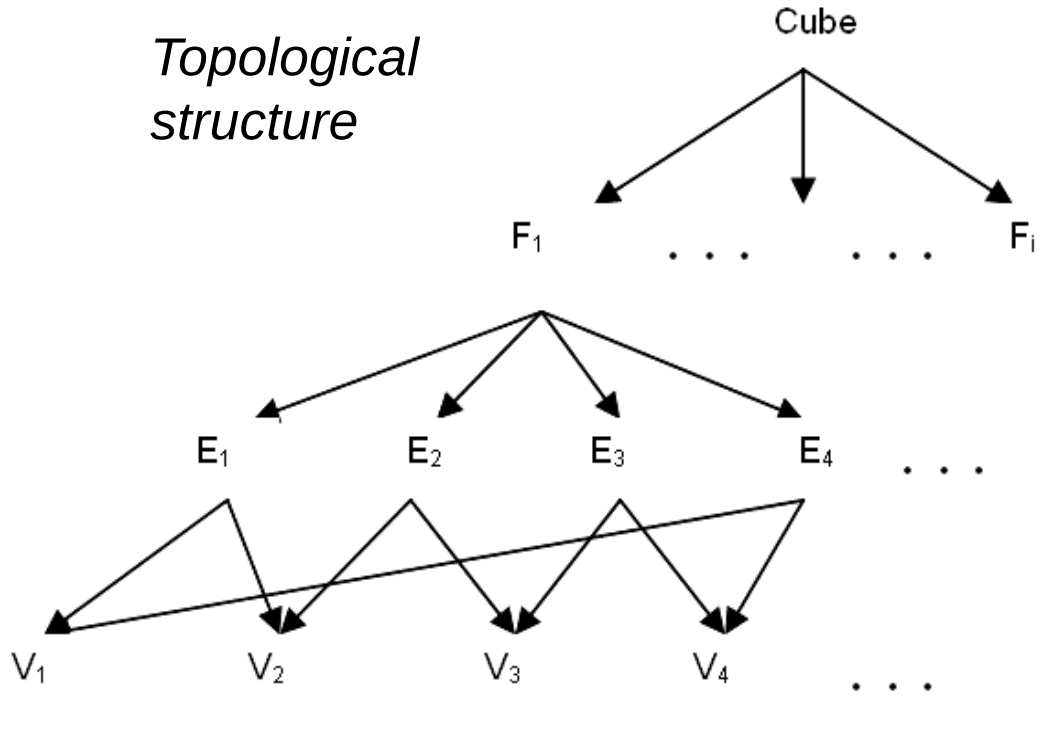
BRep of a Cube

Example: A boundary representation for a cube





Topological structure



Polyhedra and Euler's Formula

- A 3D polyhedron is a solid that is bounded by a set of polygons:

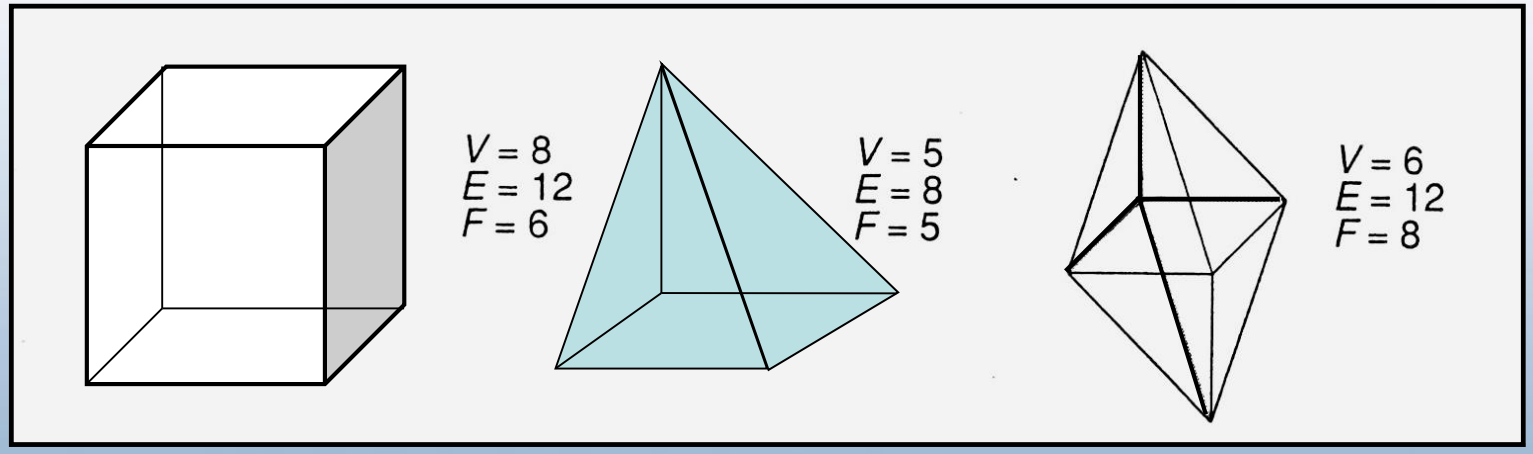
- each edge connects two vertices and is shared by exactly two faces

- at least three edges meet at each vertex

- faces do not interpenetrate.

Polyhedra and Euler's Formula

- A simple polyhedron can be deformed into a sphere (no through holes).



Some simple polyhedron with their V, E and F values.

Foley, p.545

Polyhedra and Euler's Formula

- The BRep of simple polyhedron satisfies Euler's formula:

$$\mathbf{V - E + F = 2}$$

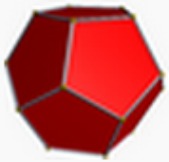
where

V is the number of vertices

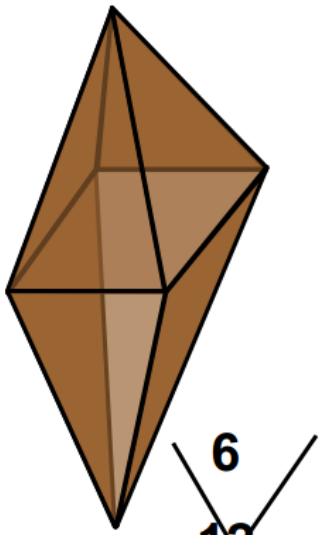
E is the number of edges

F is the number of faces.

Polyhedra and Euler's Formula

Name	Image	Vertices V	Edges E	Faces F	Euler characteristic: $V - E + F$
Tetrahedron		4	6	4	2
Hexahedron or cube		8	12	6	2
Octahedron		6	12	8	2
Dodecahedron		20	30	12	2
Icosahedron		12	30	20	2

Polyhedra and Euler's Formula



V = 6
E = 12
F = 8

V – E + F = 2

where

V is the number of vertices

E is the number of edges

F is the number of faces.

6 – 12 + 8 = 2

The internal face has to be removed as it violates the requirement that each edge is shared by exactly two faces.

Generalized Euler's Formula

- The BRep of solids that have faces with holes satisfies the generalized Euler's formula:

$$V - E + F - H = 2 (C - G)$$

where

V is the number of vertices

E is the number of edges

F is the number of faces

H is the number of holes in the faces

C is the number of separate components (parts)

G is the genus – number of holes passing though the solid (for sphere G=0, torus G = 1)

If C > 1, G is the sum of the genera of all its components.

Generalized Euler's Formula

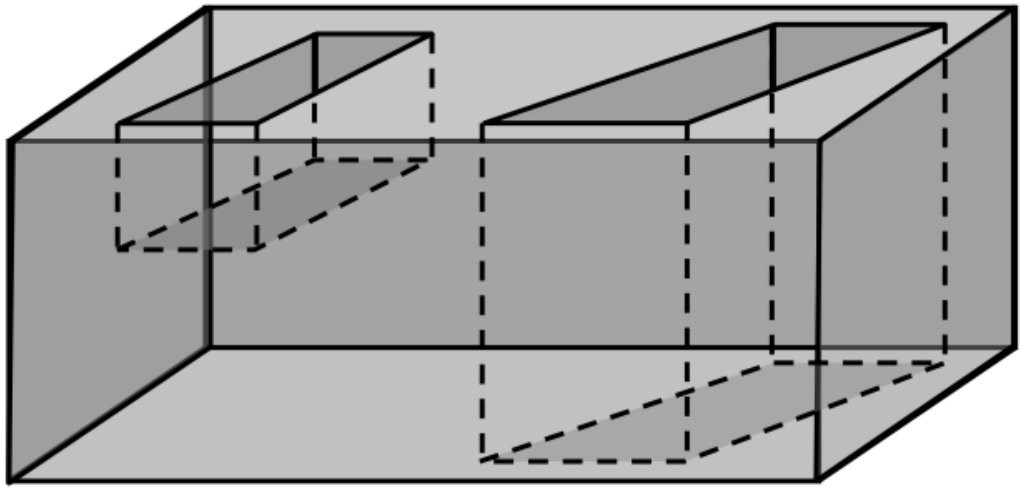
V = 24

E = 36

F = 15

H = 3

C = G = 1



$$V - E + F - H = 2(C - G)$$

V is the number of vertices

E is the number of edges

F is the number of faces

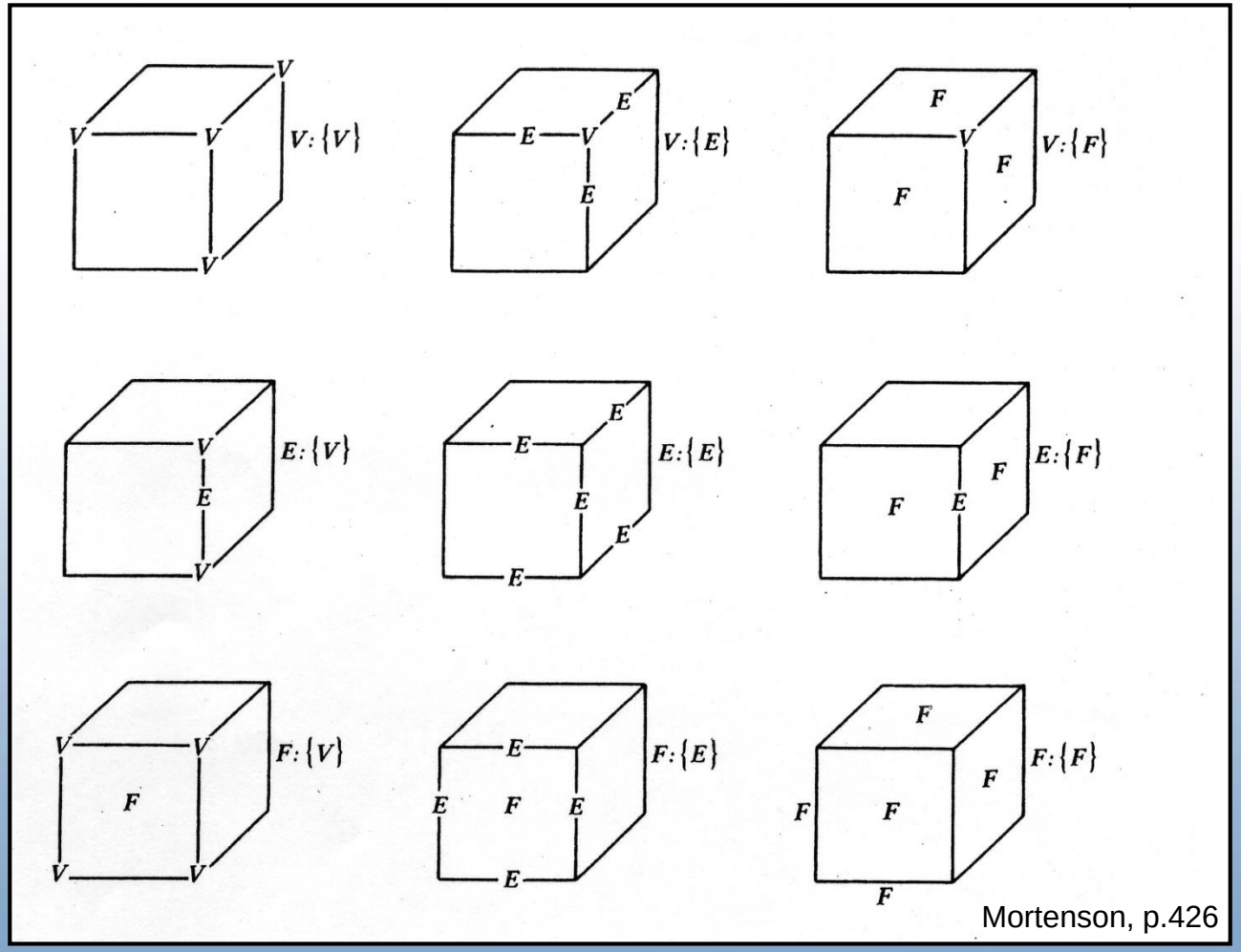
H is the number of holes in the faces

C is the number of separate components

G is the genus

Topological relationships in BRep

A polyhedron has nine classes of topological relationships between pairs of elements: vertices, edges, and faces.



Mortenson, p.426

Topological relationships in BRep

Different applications need different adjacency information:

- *V:* {*V*}, *E:* {*V*}, *F:* {*V*} in wireframe (vector) graphics to know how vertices are joined
- *V:* {*F*} in set operations to know the ring of faces around the vertex
- *F:* {*F*} adjacency among faces is needed in Euler operators.

Winged-edge structure

This data structure represents the boundary of a manifold polyhedral object. The topological information is as follows:

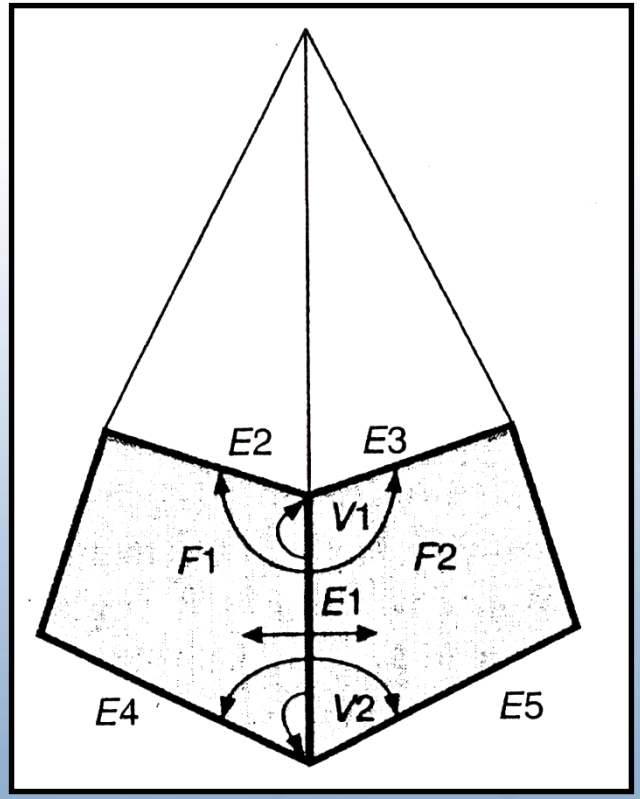
- Each face is bounded by a set of disjoint edge cycles. One cycle is the outside boundary of the face, the others bounding holes.
- Each vertex is adjacent to a circularly ordered set of edges, so the vertex table specifies one of these edges for each vertex.

Winged-edge structure

- For each edge the following information is given (see figure):

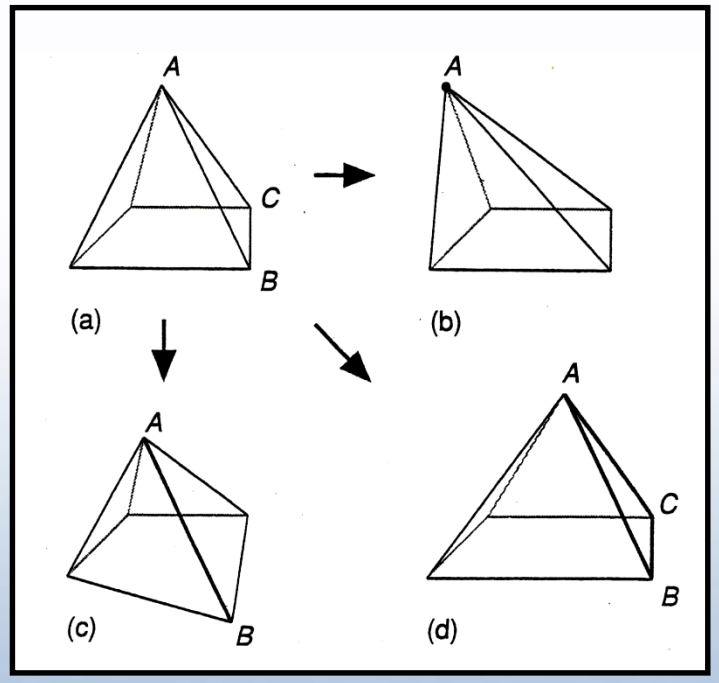
1) Incident vertices (V1, V2)
2) Left and right adjacent face (F2, F1)
3) Two edges that share V1 (E2, E3)
4) Two edges that share V2 (E4, E5)

This structure makes it possible to determine in constant time which vertices or faces are associated with an edge.

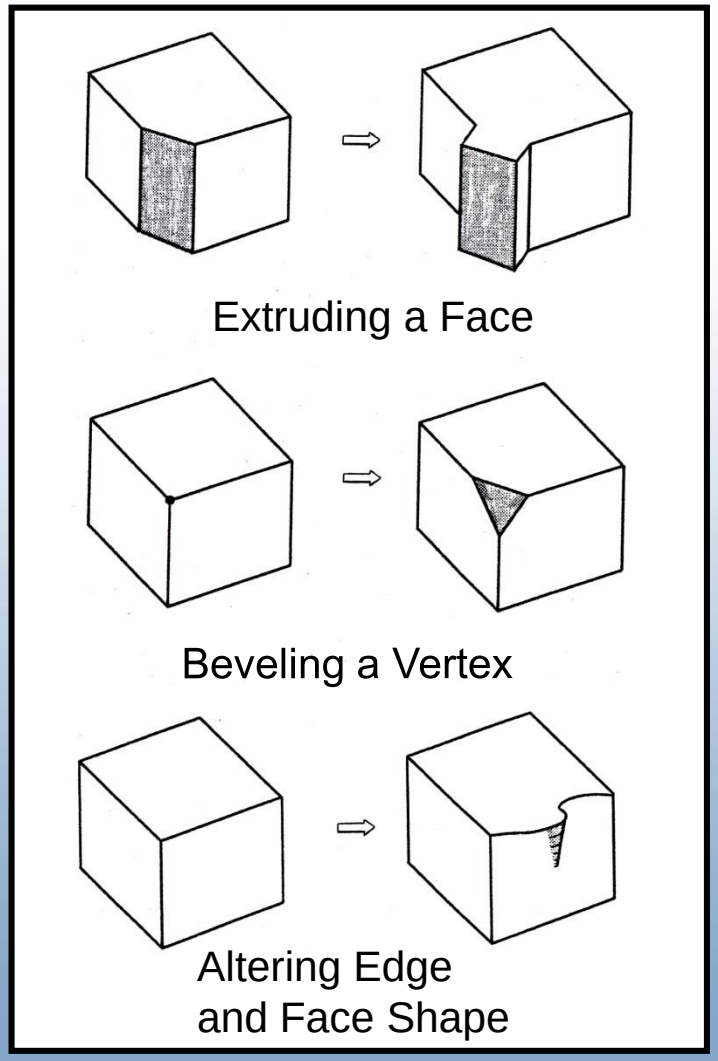


Foley, p.545

Local modifications



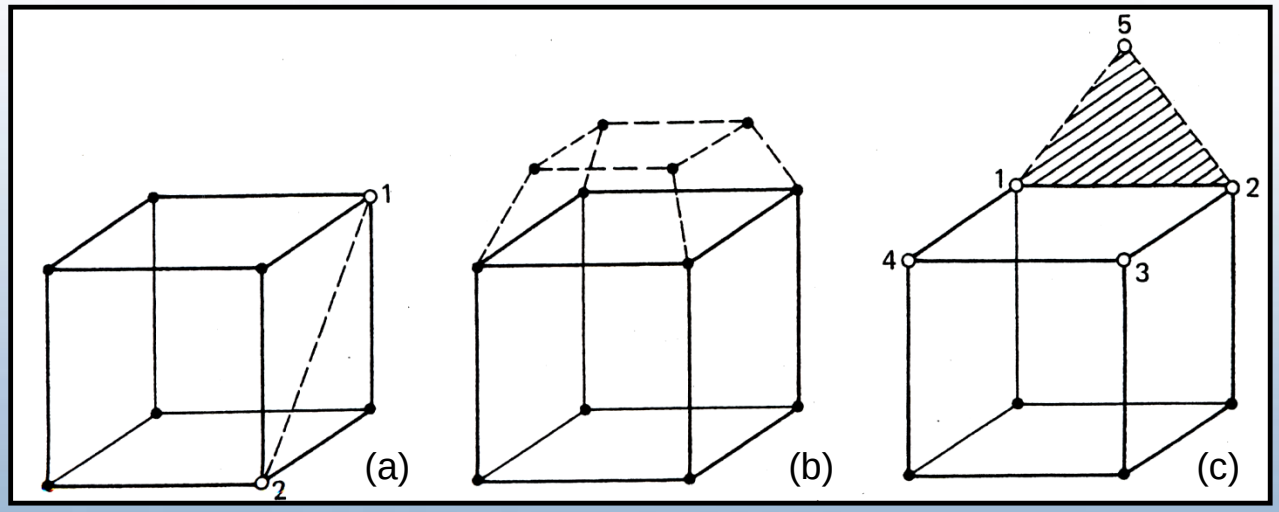
(a) An object on which tweaking operations are performed to move, (b) vertex A, (c) edge AB, (d) face ABC



Foley, p.544
Hoffmann, p.18

Euler operators

Euler operators transform the objects satisfying Euler's formula by adding and removing vertices, edges and faces.



(a, b) a cubical polyhedron is correctly modified;
(c) the result is not a polyhedron because edges (1,5) and (2,5) are not shared by two faces each.

Mortenson, p.421

Euler operators

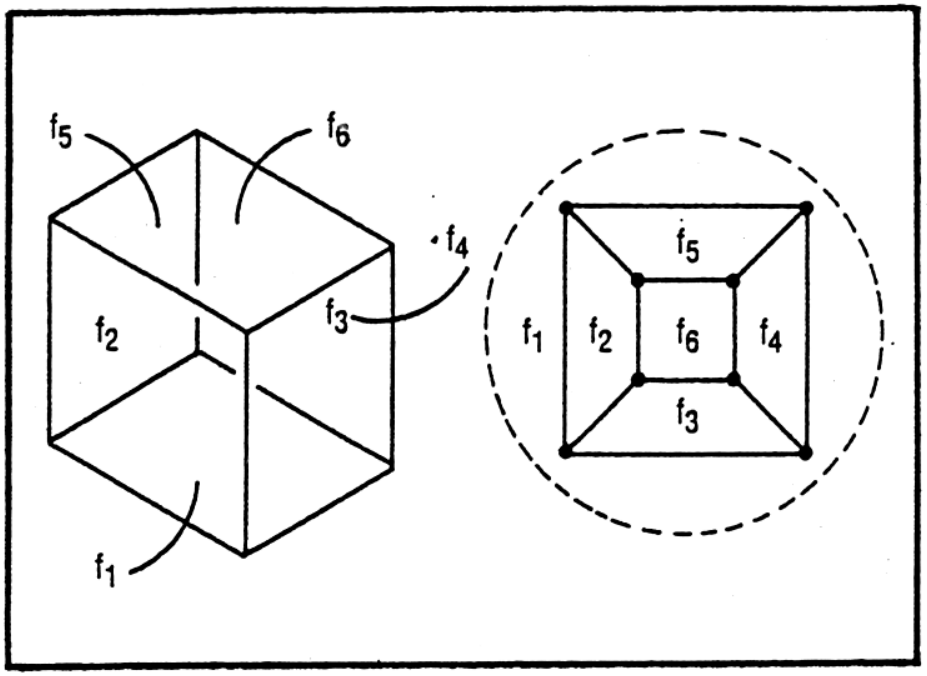
A linear combination of five primitive operators (with their inverses) can represent all objects satisfying Euler formula:

1. Make (kill) an edge and a vertex (***mev*** / ***kev***)
2. Make a face and an edge (***mfe / kfe***)
3. Make a body, a face, and a vertex (***mbfv / kbfv***);
4. Make a cavity, or passage, and a body (***mrb / krb***);
5. Make an edge and kill a hole (***me-kh***).

Advantages of Euler operators:

- Ensured topological validity of the resulting solids
- Intermediate language isolating high-level operations from the underlying data structures

Euler operators in the GWB system

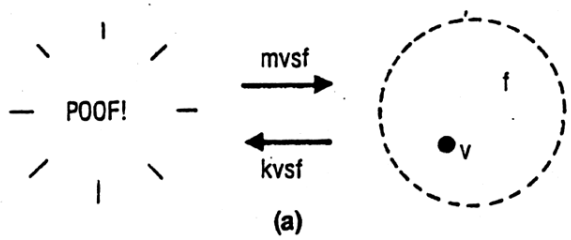


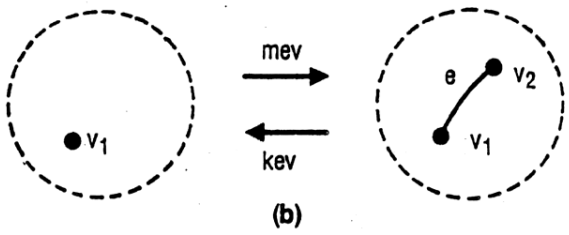
Cube topology: a plane model

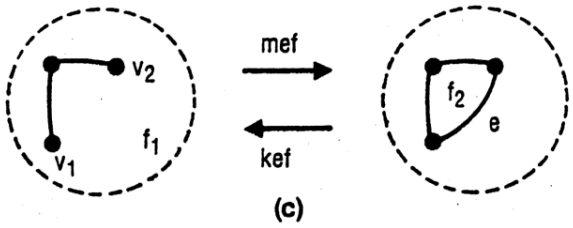
Euler operators

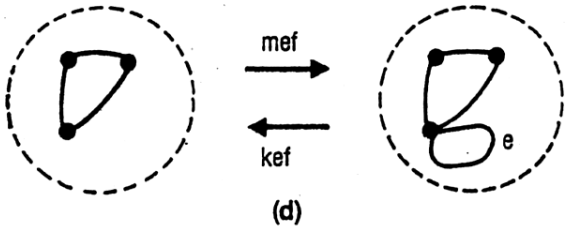
OPERATOR	EXPLANATION
mvsf(f,v)	MAKE VERTEX, SOLID, FACE
kvsf()	KILL VERTEX, SOLID, FACE
mev(v_1,v_2,e)	MAKE EDGE, VERTEX
kev(e,v)	KILL EDGE, VERTEX
mef(v_1,v_2,f_1,f_2,e)	MAKE EDGE, FACE
kef(e)	KILL EDGE, FACE
kemr(e)	KILL EDGE, MAKE RING
mekr(v_1,v_2,e)	MAKE EDGE, KILL RING
kfmrh(f_1,f_2)	KILL FACE, MAKE RING, HOLE
mfkrh(f_1,f_2)	MAKE FACE, KILL RING, HOLE
semv(e_1,v,e_2)	SPLIT EDGE, MAKE VERTEX
jekv(e_1,e_2)	JOIN EDGES, KILL VERTEX

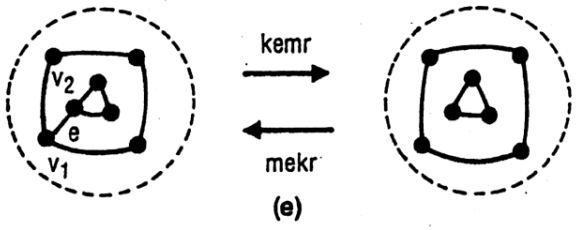
Euler operators in the GWB system

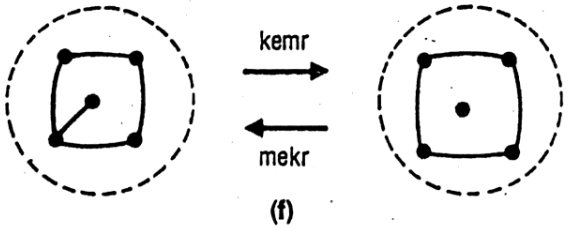


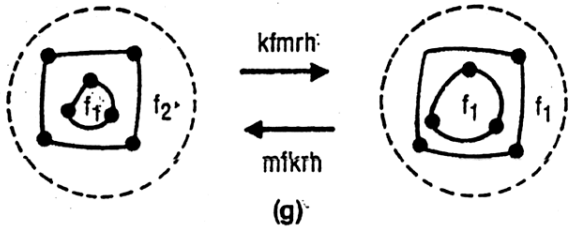


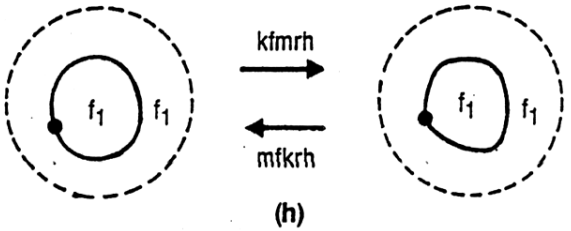


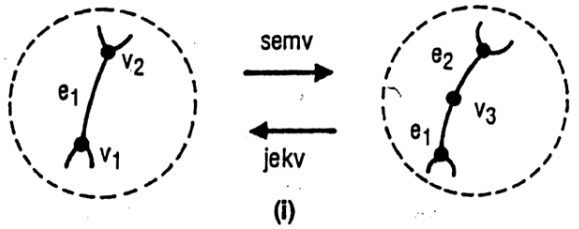


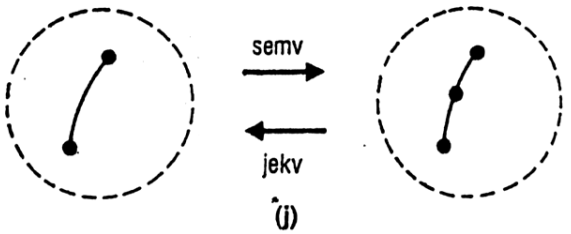












Mantila, CG&A, 1982, p.20

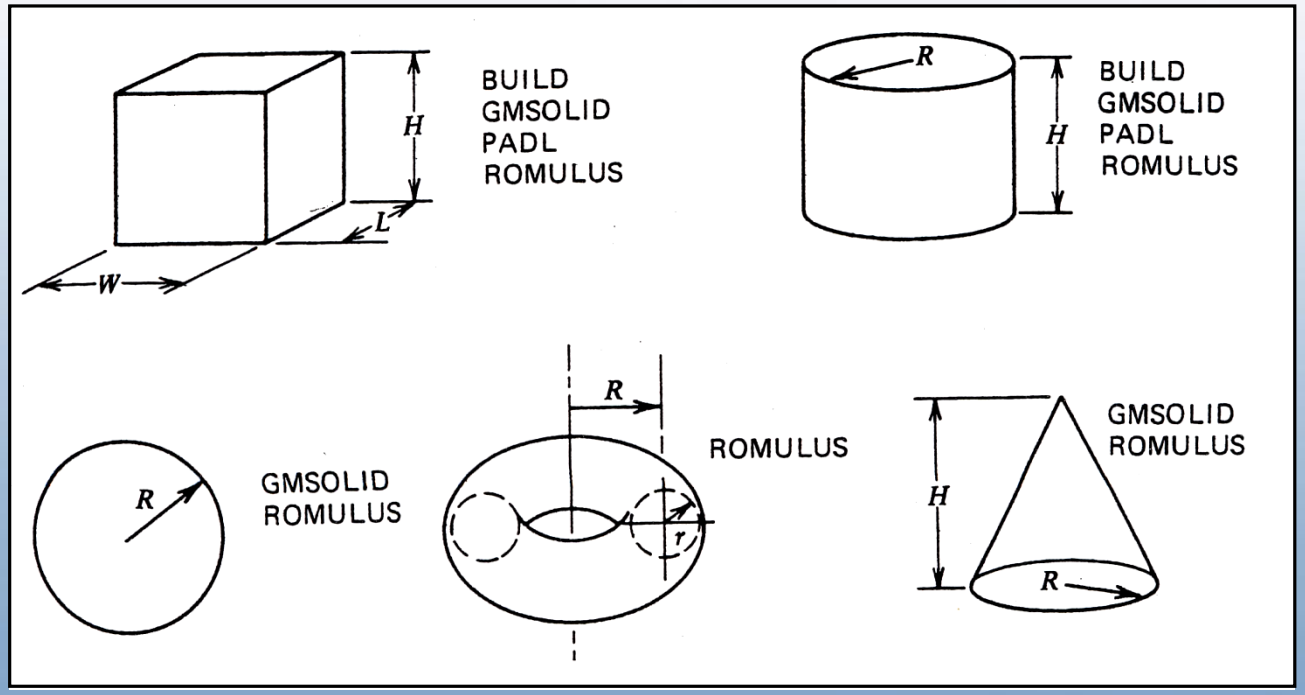
Contents

- Boundary Representation BRep
- Euler's formula
- Winged-edge structure
- Euler operators
- CSG tree
- Point-membership classification algorithm
- References

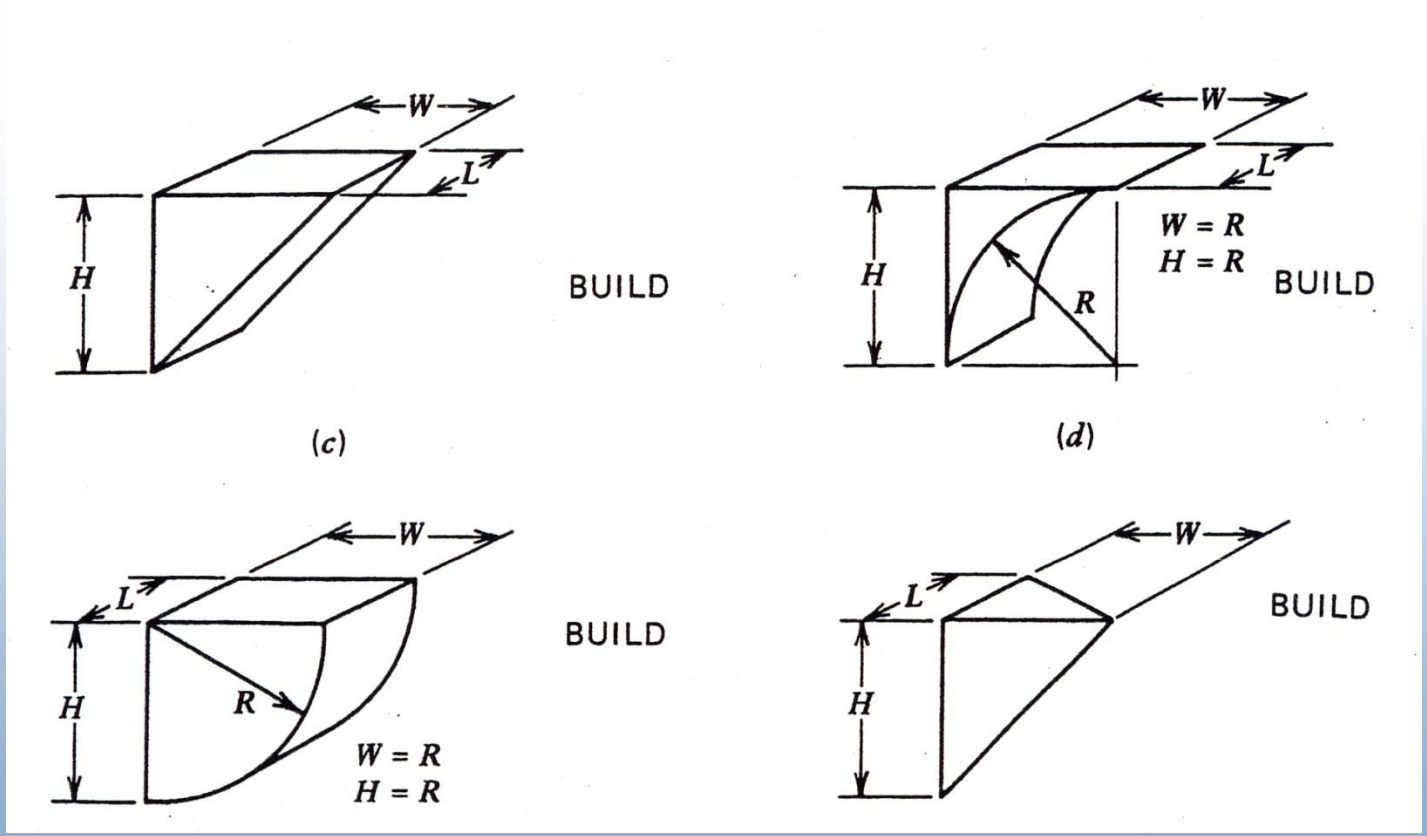
Constructive Solid Geometry
CSG: primitives

In CSG, simple primitives are combined by means of regularized set operations and rigid motions.
Standard CSG primitives:

block, cylinder, sphere, cone, and torus

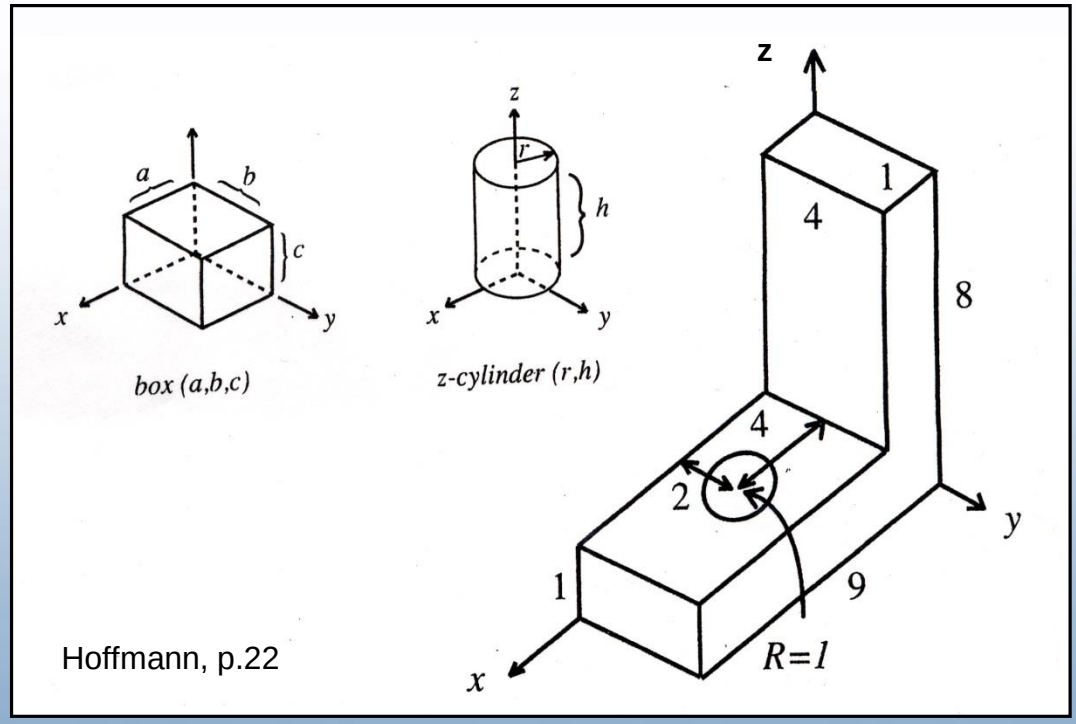


CSG representation: primitives



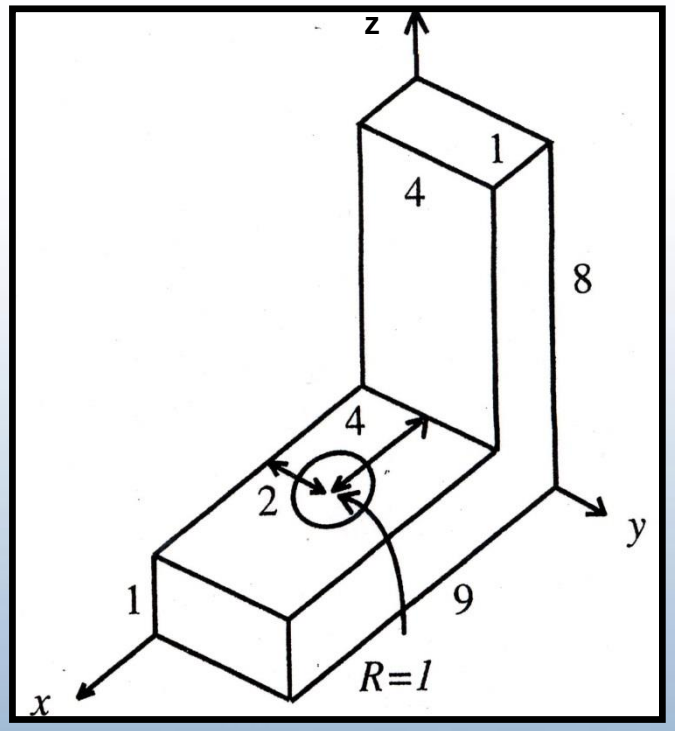
CSG Tree

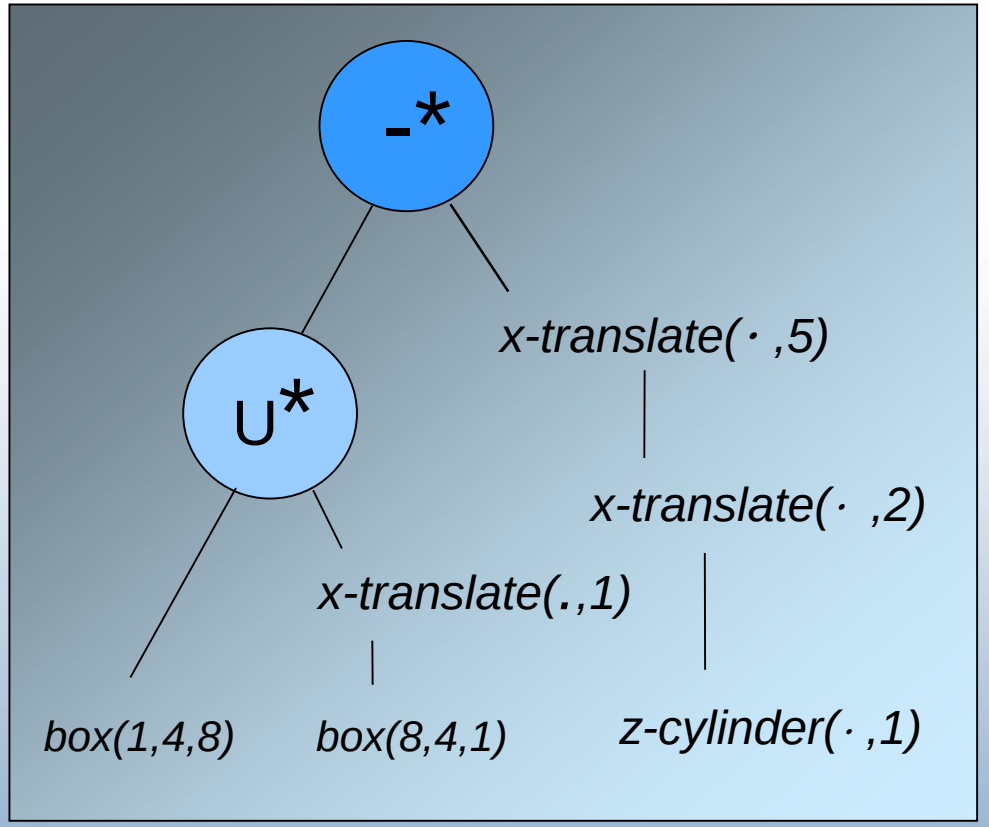
- An object is represented as a ***binary tree*** with operations at the internal nodes and primitives at the leaves
- Nodes: regularized set operations or rigid motions (translation, rotation)



Hoffmann, p.22

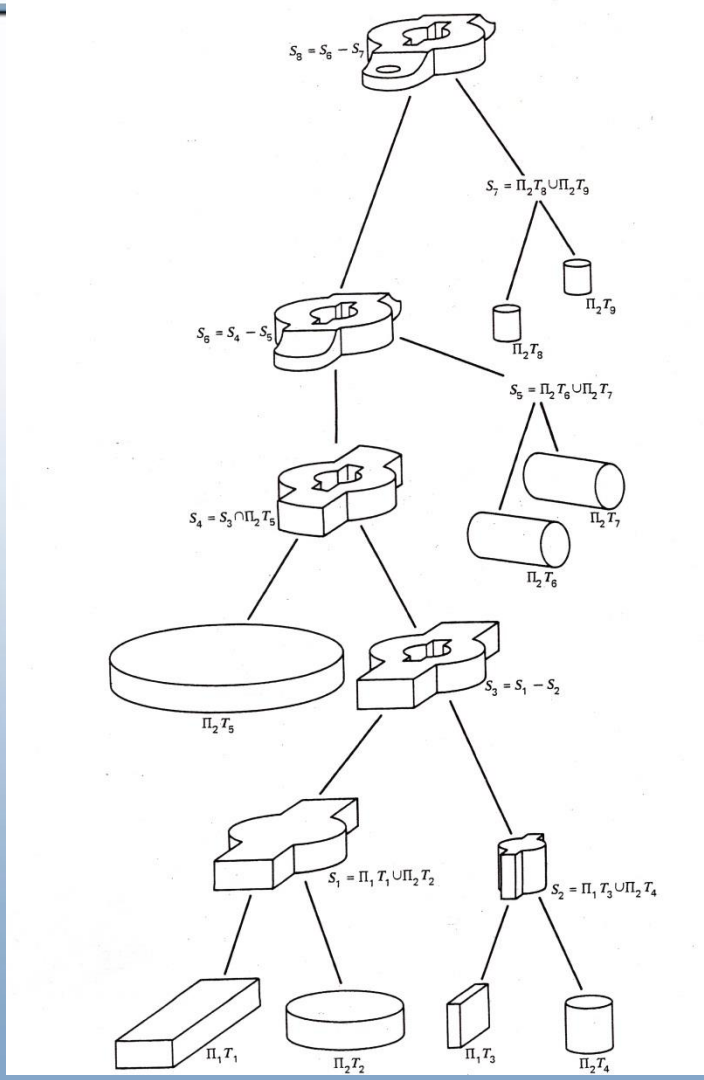
CSG Tree

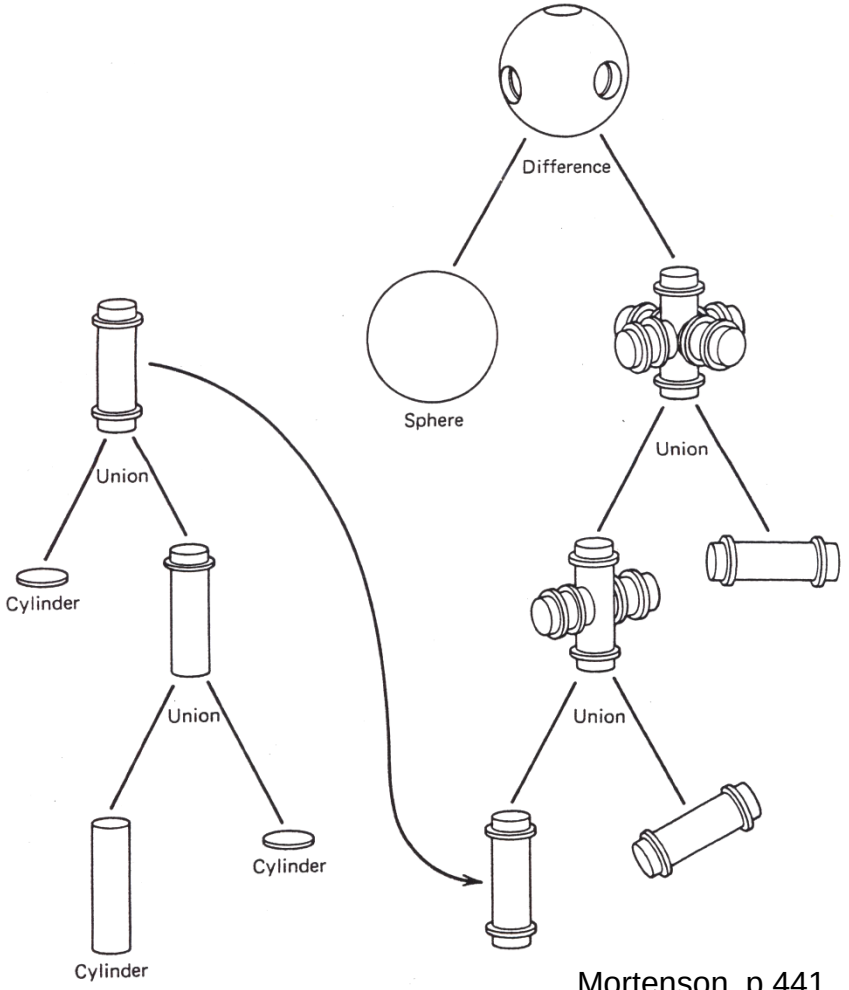




(block(1,4,8) U* x-translate(block(8,4,1),1) -* x-translate(y-translate(z-cylinder(1,1),2),5)

CSG Tree: Examples





Mortenson, p.441

Point membership classification

A solid *S* and a point *X* are given.

Query (binary predicate):

Is *X* inside or outside of *S* ?

Point membership classification (PMC) function

M[X,S] = (XinS, XoutS)

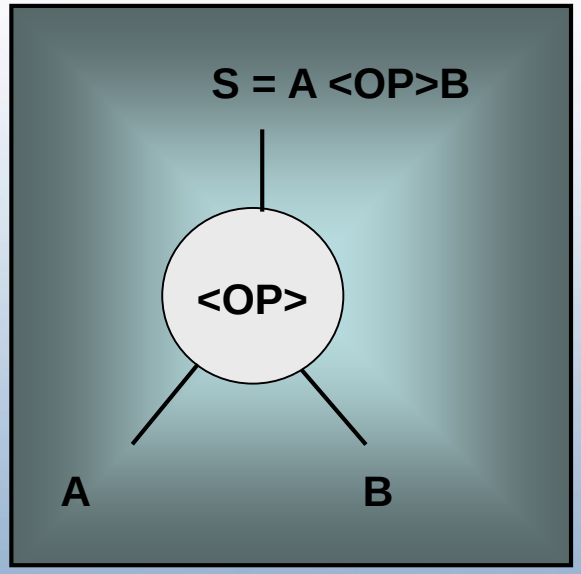
Query (ternary predicate):

Is *X* inside, outside or on the boundary of *S* ?

Point membership classification (PMC) function

M[X,S] = (XinS, XonS, XoutS)

Divide-and-conquer paradigm



Tilove IEEE Tr. on Comp. 1980, p.877

F(S) ← IF (S is a primitive)
THEN prim - f(S) ELSE combine
(f(A), f(B), <OP>)

The divide and conquer paradigm.
<OP> is a regularized operator
($\cup^*$, $\cap^*$, or $-^*$)

Point membership classification

M [X, S] ← IF (S is a primitive)
THEN prim–M (X,S)
ELSE combine (M [X, left–subtree(S),
M [X, right–subtree (S)], root <S>))

- *prim-M* is a primitive classification procedure and must produce “in”, “on” or “out” answers for a given point
- *combine* applies set operations (three-valued logic!) to its arguments
- postorder tree traversal

Recursive PMC algorithm structure

1. Point coordinates *(x,y,z)* are sent to the root of the CSG tree.

2. Downward propagation

- Coordinates are propagated into the tree down to the leaves, possibly altered at the motion nodes.

- At each leaf, the final point coordinates describe the same point, but in the local coordinate frame of the primitive solid.

Downward propagation cases:

1) If *(x,y,z)* arrives at a set-theoretic operation node, it is passed unchanged to the two subtrees

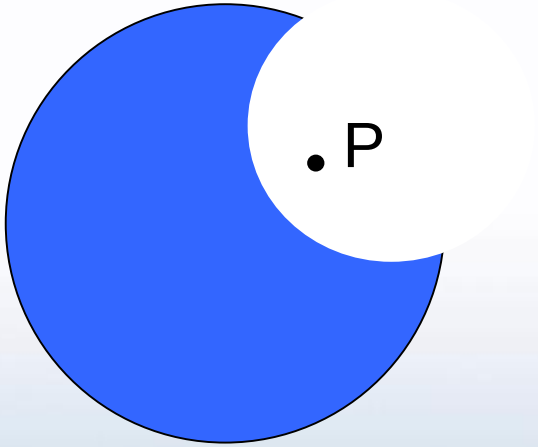
2) If *(x,y,z)* arrives at a motion node, the inverse transformation is applied to *(x,y,z)*, resulting in new local coordinates *(x',y',z')*, which are sent to the two subtrees

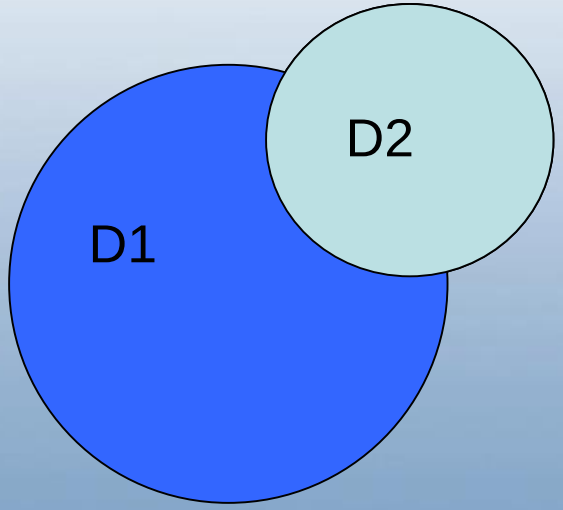
3) If *(x,y,z)* arrives at a leaf, the point is classified against the primitive, and the classification is returned to the parent of the leaf

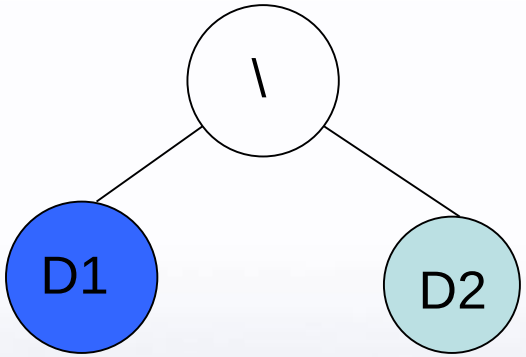
3. Upward propagation

- Classifications from the subtrees are combined in the set-theoretic operation nodes
- No work is done at motion nodes.

PMC Example







M [X, S] ← IF (S is a primitive)
THEN prim–M (X,S)
ELSE combine (M [X, left–subtree(S),
M [X, right–subtree (S)], root <S>))

Postorder tree traversal:

D1-D2-\

PMC steps:

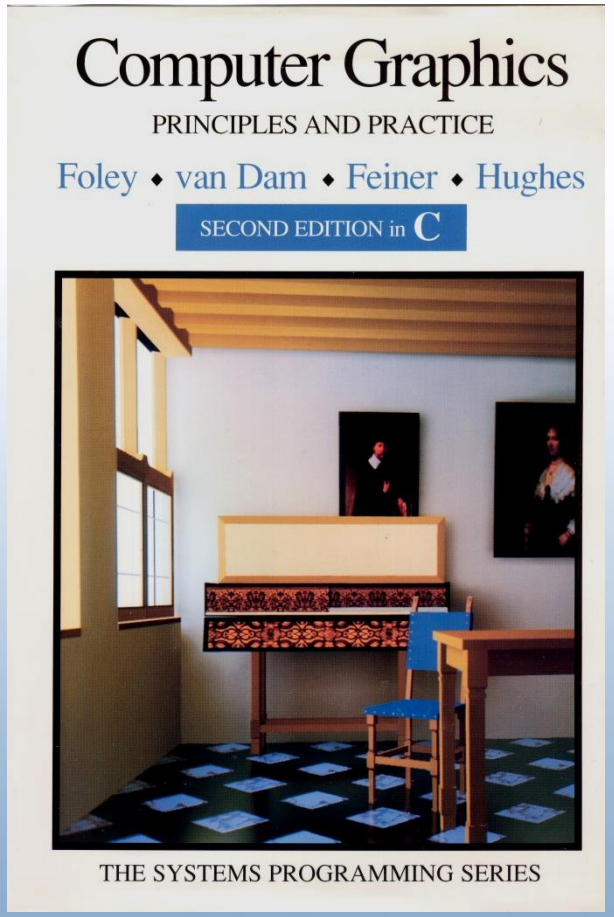
1)

2)

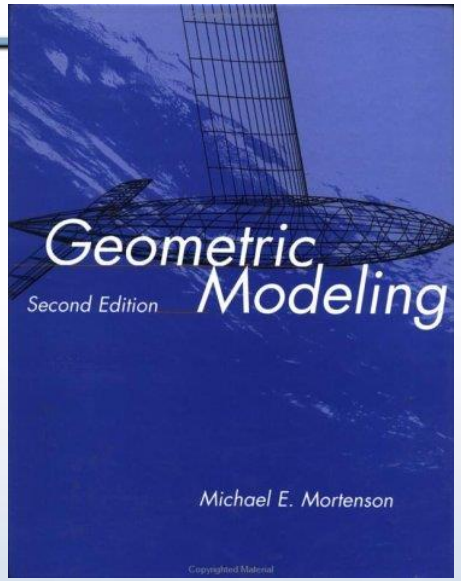
3)

References

- James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics: Principles and Practice (2nd Edition in C), Addison-Wesley, Reading, MA, 1997.



References



- Michael E. Mortenson, Geometric Modeling, John Wiley and Sons, 1985.
  Second edition, 1997.
  Third edition, 2006.
- Christoph M. Hoffmann
  Geometric and Solid Modeling. An Introduction, Morgan Kaufmann Publishers, 1989, 338 p.