

004
B78

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

ПЕРЕОБЛІК 2022р.

V.V. Borysevych

SOLID MODELING

Synopsis of lectures

ТВЕРДОТІЛЬНЕ МОДЕЛЮВАННЯ

Конспект лекцій

Научно-техническая
библиотека
"ХАИ"



mt0112537

**НАУКОВО-ТЕХНІЧНА
БІБЛІОТЕКА**
Національного аерокосмічного
університету ім. М.Є.Жуковського
«Харківський авіаційний інститут»

Харків «ХАИ» 2008

Борисович В.В. Твердотільне моделювання: консп. лекцій / В.В. Борисевич. – Х.: Нац. аерокосм. ун-т «Харк. авіац. ін-т», 2008. – 72 с.

Наведено загальні відомості, необхідні для вивчення основ конструювання твердотільних моделей деталей і механізмів, що використовуються у багатьох додатках САПР. Описано основні особливості використання твердотільного моделювання в таких галузях, як конструювання деталей, скінченно-елементний аналіз, генерація програм для верстатів із ЧПУ, кінематичний і динамічний аналіз, швидке прототипування й створення складань.

Для студентів, що вивчають дисципліни, пов'язані з літако- та вертольотобудуванням, виробництвом літальних апаратів, із системами автоматизованого проектування, а також для студентів будь-яких спеціальностей, пов'язаних з вивченням створення тривимірних зображень об'єктів і моделювання процесів.

The general information for study of the basic of the solid modeling of the part and of assemblies for different solid modeling systems is given. The main particularities of the solid modeling applications for such fields as part design, finite element analysis, generation of the program for computer aided machining, kynematic and dynamic analysis, rapid prototyping and assembling.

For those students who study discipline relating to aircraft and helicopter manufacturing, to CAD/CAM/CAE systems and for any special subject relating to study of the solid modeling and modeling of processes.

Іл. 37. Табл. 3. Бібліогр.: 20 назв

Рецензенти: канд. техн. наук, доц. Г.К. Крижний,
канд. техн. наук, проф. А.М. Краснокутський



© Національний аерокосмічний університет ім. М.С. Жуковського
«Харківський авіаційний інститут», 2008

1. TYPES OF SOLID MODELS

Integration of functions within the factory requires a product definition that is unique and consistent throughout the design and manufacturing process; it is computer graphics that makes possible a practical implementation of this dictum. At the heart of this definition is the geometry of the product – its shape. The world has three spatial dimensions, so computer models must be three-dimensional.

Three-dimensional CAD models can take three forms:

1. Wireframe. Includes only points in space and the lines connecting them. Objects are represented by their edges.
2. Surface. The edges of the wireframe are spanned by mathematically defined areas.
3. Solid. The space enclosed by surfaces is defined and forms a closed volume.

Wireframe representations of complex objects are very difficult to understand visually, because computer displays and paper plots seldom give an indication of depth and invisible lines. Seeing all the edges at once leads to perceptual confusion because of ambiguities (Fig. 1.1).

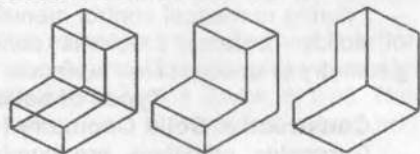


Fig. 1.1. Ambiguity of the wireframe model

Advantages of solid models (SM) are:

- SM are the least abstract and most realistic of the three forms;
- a complex object can be decomposed into a much smaller number of solid components (primitives) than surfaces, lines, or points;
- SM and surface models allow the generation of images with hidden lines and hidden surfaces removed, which are much more realistic than wireframe models (Fig. 1.2);

– the most important aspect of solids is that their integrity can be computationally determined. In other words, SM software with the aid of the computer can tell if a given object is a legitimate solid or not. Objects designed in SM systems will be manufacturable, at least from the point of view of spatial integrity. It is advised to keep in mind that although a given solid may be valid or geometrically complete (forming a closed volume), this by no means guarantees it to be manufacturable;

- SM systems can be automatically tested for interferences among parts;
- SM system makes analysis tools accessible to designers who do not understand the mathematical details;

This edges are not identify

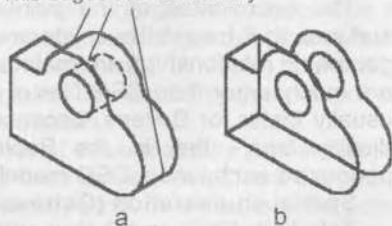


Fig. 1.2. Real and wireframe view:
a – real 3D view; b – wireframe view

- SM makes it possible to build "software" prototypes. These are models that exist only within the memory of the computer. These models can be subjected to computer-based simulations of the prototype tests, and the results can be used to build a real prototype. Computer-based simulations can often be better representations of realworld conditions than those to which physical prototypes are subjected;
- solids are described by topology in addition to geometry. Topology is the properties of the shape that are unaltered by deformation;
- SM allow to use parametric/relational (P/R), feature-based and variational approach in modeling;
- database of SM includes information which can determine the mass properties of the physical objects being modeled so it may be possible to predict object weight or dynamic behavior of a complex mechanism;
- SM may be used for kinematic and dynamic analysis;
- during numerical control manufacturing the computer-verifiable nature of solids obviates problems concerned with flaws such as missing geometry or unconnected surfaces.

Types of solid modeling system

Constructive Solid Geometry (CSG, C-Rep).

Geometric primitives are combined in this approach by means of Boolean operations. CSG systems store a record of the primitives and the operations used with them; each time the part is to be displayed, it is computed from the tree of primitives and operations. Consequently, CSG databases are usually quite small relative to other representations, but display times are sometimes longer.

Boundary Representation (B-Rep).

Making use of the fact that higher-dimension geometries are bounded by objects of lower dimensions, B-Rep systems maintain an explicit "tree" of boundaries: a solid is bounded by surfaces; a surface is bounded by lines; a line is bounded by points.

The coordinates of the points that bound the lines that bound the surfaces that bound the solids are explicitly stored in B-Rep systems; the geometric relationships are maintained by pointers. As a result, B-Rep files are much larger than CSG files of similar parts. But display performance is usually faster for B-Reps, because the model need not be "evaluated" at display time - that is, the Boolean operations on primitives that are performed each time a CSG model is displayed do not have to be done.

Spatial enumeration (Octrees, O-Rep)

While both CSG and B-Rep systems approach solid geometry from the point of view of the part, it is also possible to consider the part in the context of the space in which it resides. Space can be divided by a progressively refined grid of cubes, using cubes that stay completely within or completely outside the model, and smaller and smaller cubical partitions to approximate the boundaries of the model. O-Rep system performances degrade much more slowly with the complexity of the model than do B-Rep or CSG.

Hybrids

Most of the SM systems on the market today are some form of hybrid of CSG and B-Rep. CSG is generally used for internal representation and B-Rep for the generation of views. Since CSG models are stored unevaluated, they are resolution-independent. B-Rep polygonization of models can be geared to the desired trade-off between resolution and display speed.

2. SOLID MODELING CONCEPT

While 2D drawings can be created manually or electronically, solid models must be created in an electronic "drawing universe". Solid models themselves are not physically accessible. CAD workstations are used to create, edit, and display 2D representations of the electronic solid model.

3D coordinate systems

Solid models are located in an electronic space that is defined in terms of 3D Cartesian coordinates. This is known as the 3D work space or model space. Some 3D systems provide an auxiliary 2D work space as well. This is sometimes referred to as draw space and is where 2D text and conventional details are located.

Three-dimensional coordinates are used to specify the location of points in space, the distances between pairs of points, and displacements between consecutive positions of a point. A coordinate system consists of an origin and a system of reference planes or axes. A coordinate system is sometimes referred to as a coordinate frame.

The location of a point is specified by giving its distance and direction from a given coordinate frame's origin. The location of an object is determined by assigning 3D coordinates to an origin point on the object. The orientation of the object is determined by assigning 3D coordinates to additional features, or to coordinate system local to the object's definition.

3D model space has at least one permanent coordinate system. This is called the global, or world coordinate system, and is typically the default coordinate system. For specific tasks, temporary or local coordinate systems can be defined relative to the global coordinate system, to points on an entity, or to an entity's coordinate system.

A Cartesian coordinate system consists of three mutually perpendicular axes which intersect at the origin. These axes extend away from the origin indefinitely in both a positive and negative direction. The location of a point is specified by listing three coordinates, one measured along each of the three axes. The cartesian is the most common coordinate system used in SM systems. Another way to specify the location of an object is to use cylindrical or spherical coordinates. Both forms can be used within the Cartesian coordinate system. The choice of coordinate system format depends largely upon the nature of the task at hand.

The cylindrical coordinate of a given point consists of the following information:

1. A radial distance r from a chosen reference frame's origin to the projection of the point of interest on that frame's X - Y plane.
2. The angle α subtended by the X axis and the radial distance vector.
3. The projected distance d of the point from the X - Y plane.

The spherical coordinates of a given point consists of the following information:

1. A vector r of radial distance from the origin to the point of interest.
2. The angle α_1 between the X axis and the projection of the radial distance vector on the X - Z plane.
3. The angle α_2 between the X - Z plane and the radial distance vector.

Reference Planes

Any two of the three axes in a Cartesian coordinate system form a reference plane. There are three generic reference planes: the X - Y plane, the Y - Z plane, and the X - Z plane. A single coordinate value measures the distance from the reference plane formed by the other two axes. Thus, a Z coordinate specifies the distance from the X - Y plane to a point, measured in a direction parallel to the Z axis.

Construction concept

Fundamental construction concepts are key to producing basic and complex solid models. Understanding these concepts and their applications will increase one's ability to grasp more advanced concepts, thus increasing the overall capabilities of an SM system.

Construction Planes

Although SM systems allow the construction of objects in 3D model space. The designer's interface consists of the monitor screen, pointing device, and keyboard, which are typically two-dimensional. Movement of the pointing device on a digitizer tablet or desktop is tracked by cross hairs, aperture boxes, or other screen cursors. These movements are two-dimensional.

During constructive operations the pointing device and screen cursor can be made to correspond to the default or active construction plane, a reference plane in 3D model space. When points and entities are entered with the pointing device and/or keyboard, they are typically assigned the elevation and orientation of the active construction plane.

If the global or world coordinate system is active, points and entities appear on the X - Y plane of that coordinate system. If a temporary or local coordinate system is active, its X - Y plane is typically the construction plane. In some systems, an elevation above or below the X - Y plane can be defined as a default value, so that the construction plane is parallel but not coplanar to the X - Y plane of the current coordinate system.

When a new construction plane is created, a complete local Cartesian coordinate system is established with its X - Y plane typically defined as the new active construction plane. The graphic representation of construction planes may vary from system to system. Their display attributes can sometimes be modified to include a rectangular border, snap grids, to suppress the positive Z axis, or to assign color to individual

components. There may be many auxiliary construction planes defined in model space as needed while only one is current and active at any given time. The construction and modification of solid model is facilitated by creating and activating construction planes as needed, but not all SM constructive operations require the use of a construction plane.

Building Blocks

A solid model of a complex object typically consists of several simple objects, just as a 2D drawing consists of simpler geometric entities, such as lines, circles, and arcs. Because lines, circles, and arcs are not 3D entities, they cannot enclose space and so cannot be assigned mass properties. Also, lines, circles, and arcs cannot be combined directly to yield a solid model, although they are used in the creation of a solid model's visual display.

Basic solid models are constructed from elementary solid objects or primitives (spheres, cylinders, prisms, slabs, etc.). Each of these solid primitives is a shape that can be described simply. Solid primitives can be combined like building blocks to construct more complex objects. Thus, a model of a nail can be assembled from a cone and two cylinders.

These building blocks are typically combined with the use of the three Boolean operations (sometimes referred to as set operations): UNION, SUBTRACTION, and INTERSECTION (Fig. 2.1).


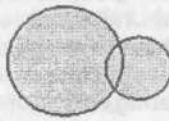
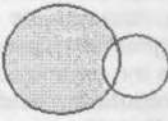
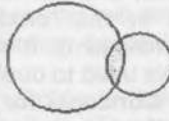
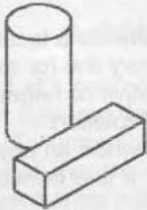
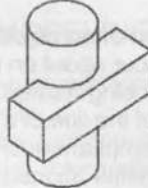


INITIAL OBJECTS	UNION $A \cup B$	SUBTRACTION $A - B$	INTERSECTION $A \cap B$
<p>Set</p>  <p>A B</p>	 <p>A B</p>	 <p>A B</p>	 <p>A B</p>
<p>Solid bodies</p> 			

Fig. 2.1. Boolean operations

Advanced 3D surfacing techniques may be required to define very complex portions of a solid model.

Creation Solids from 2D Geometry

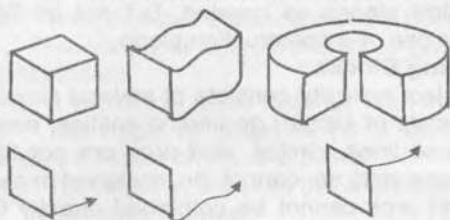


Fig. 2.2. Creation solids from 2D geometry

3D solids can be generated from 2D geometry by extruding, sweeping, or revolving a profile along an axis. Typically, the active reference plane is used to construct a 2D profile. When generating a 3D solid, the 2D profile must form a closed area. In conventional 2D wireframe and surface modeling, this closed condition is not required. The

closed curve may consist of several linear entities connected end to end, depending upon the SM system used (Fig. 2.2).

Group and Assemblies

Physical products typically consist of parts and components organized into groups and/or assemblies. To reflect this hierarchical structure, solid models must also be organized into groups and assemblies.

The term group implies that the components of the group share a common level in the assembly process. The term assembly implies that the assembly is composed of components from a lower level in the assembly process. For example, a bolt, nut, and washer used to bolt two parts together may be treated as a group for ease of manipulation. Two parts, along with several bolt-nut-washer groups may form an assembly. As in real products, assemblies and individual components can be combined to form higher-level assemblies. Groups and assemblies are tools used to duplicate the organizational structure of the physical product.

Concepts for manipulation and management of 3D objects.

Translations and Rotations

The location of an entity in a solid model can be modified by either translation or rotation. Translation shifts an object along one, two, or three axes without changing its angular orientation relative to any of the three axes.

Rotation changes the angular orientation of an object relative to a local or global coordinate system. Rotation may occur about an arbitrary line (or axis) that is part of or separate from the object being rotated. Positive or negative rotation may occur based on the direction of the line or axis of rotation.

Translation and rotation often occurs simultaneously, as when an object is translated along a curved path but remains aligned with a line between the origin and a fixed point on the object.

Some workstations provide auxiliary controls to enable a designer to translate and/or rotate the solid model relative to the global coordinate system, active reference plane, or a reference line in the case of rotation, by turning knobs or repositioning other physical control devices. Such

translations and rotation are hardware-controlled and are independent of the SM software.

Copying, Scaling, and Mirroring

Components of physical products are often duplicated. Bolts, nuts, and washers are examples. Some components occur in identical shapes, but varying sizes. If a specific component is used several times in a product, it need be modeled only once. Additional instances can be copied from the first.

Similarly, if a component is used in several different sizes, only one instance of the component need be modeled. Other instances of the component can be created by simultaneously copying and scaling the original component.

Physical objects often occur in left- and right-hand versions, or mirrored images. Automotive design, for example, utilizes this concept extensively. Once the right-hand version of a SM component is defined, the left-hand version can be created with a mirroring operation. In SM, the mirrored objects are symmetrical across a selected construction plane known as the mirror plane.

Layers

Layers, sometimes referred to as levels, are techniques used to organize and manage the solid model data for display. The term layer is derived from the overlay or pinbar methods used in graphic design. Sheets of punched Mylar are registered on a pinbar, so that a drawing can be separated into objects, colors, classes of objects, systems, etc. Each sheet of Mylar corresponds to a layer of the composite drawing. In SM, layers are expanded into the third dimension and are typically assigned portions of the entire composite model. Layers are convenient for storing reference geometry, design, and modeling notes, as well as system-dependent information.

In an SM system, layers are attributes or properties of objects. When an object is defined, a value is assigned to the layer attribute. The layer attribute can then be used to select objects for operations by specifying the layer name or value. Additional common uses for layers are to control the visibility, color, and linetype of a class of objects.

3. SOLID MODELING DATABASE STRUCTURES

Let's consider representation and modeling methods used in modern SM systems databases.

To understand the function of various parts of a CAD database, it is useful to consider the information represented at different "levels" of an engineering effort. The global characteristics of a design are considered "high-level" information, while local details make up the "low level" of a design. Often a "top-down" design strategy is employed where the low-level design is derived from high-level design information, using various tools and methodologies. In computer programming, for example, a

program is designed and written in a high-level language and then translated by a compiler into a low-level "machine language" for the actual control of the computer.

Early CAD systems were developed to model and manipulate the lower levels of a design, primarily drawings. Their databases performed mostly a bookkeeping function, keeping track of large numbers of points, lines, curves, and other items. With SM, however, CAD systems extend into the higher levels of design information, placing increased demands on the database.

The primary function of any CAD database is to symbolically represent certain characteristics of real or imagined 3D objects. To classify its progression, CAD systems can be divided into four generations based on the characteristics modeled. These are as follows:

First generation – Drawings. Objects are represented by the projections of selected edges on a 2D plane. Lines and text can be interactively entered and updated.

Second generation – Wireframes. Selected edges are modeled in 3D. This supports 3D line display and the automatic generation of 2D views from any viewpoint.

Third generation – Surface. Objects are represented by modeling selected surfaces. Realistic shaded images can be produced.

Fourth generation – Solids. Objects are represented by the 3D space they occupy. Advanced features such as mass property calculations and interference detection can be performed automatically.

Basic Database Concepts

To a designer, the database is the part of a CAD system. With solid models achieving a high level of understanding and interaction can be much more difficult than with a drawing. Here are a few reasons why:

1. Important details are often visually hidden or obscured in 3D views, hindering visualization and control.

2. With solids, the elements of a model are interrelated and must adhere to various rules needed to maintain model integrity.

3. SM operations are very powerful, but can have side effects that are not well understood (objects can become invalid).

4. Users do not view and interact with a single model in the database but with a collection of models that represent various aspects of the object.

Because of these difficulties, an understanding of the SM database can help to reduce wasted effort, disappointment, and erroneous results when using SM systems. It should be noted that modern CAD databases contain a wealth of information not directly related to the shape of a model. This includes text notes, unit and scale information, material properties, part numbers, modification history, and countless other pieces of information.

Representing Geometric Elements

Solid models are represented by sets of geometric elements that are somehow connected. Most all geometric elements in CAD are defined by polynomials (equations where the exponents are positive integers and

coefficients are constants such as $ax^2+bx+c=0$). It is very convenient because of the vast array of mathematical tools that can be applied to them (differentiation to compute surface normals)

Degree of Equations

Robustness of a geometric element is limited by the largest exponent used in its defining polynomial or set of polynomials. This is referred to as the degree of the equation. With linear equations the largest exponent is 1 ($ax+by=c$). They can be used to represent straight lines and flat elements. Quadratic equations have squared variables and can represent quadratic curves (arcs, parabolas, ellipses). $x^2+y^2+z^2=R^2$ - defines the surface of a sphere where r is the radius.

Sculptured shapes

While many useful solid models can be constructed from geometric elements defined by quadric surfaces, there has been an increasing need for shapes with free-form or "sculptured" surfaces to fulfill aesthetic, functional, efficiency, or other requirements, usually a sculptured shape (curve, surface, or solid) is defined by a formula and the location of a number of control points that may lie on, inside or outside the shape. Such schemes include NURBS (Non-Uniform Rational B-Splines) and Bezier shapes. In addition to providing increased representational freedom, geometric transformations (translation, rotation, and scaling) can be easily performed by transforming the control points, and local shape changes can often be made simply by moving nearby control points.

SM system must have computer procedures that are able to mathematically handle the interaction of any combination of elements (e.g., determine the edge of intersection between a cylinder and a sphere, a cylinder and a cone). Traditional SM databases represent a wide variety of geometric elements and therefore require a large number of such procedures.

Since newer representation schemes such as NURBS are able to represent classical shapes as well as sculptured shapes, their use may mean that only one or small number of shape-representation formats are needed. This can simplify the data base and greatly reduce the number of geometric procedures. However, the required procedures are often complex, require more processing, and may be less reliable.

Bounded versus Unbounded objects

Geometric elements can form a boundary for elements of the next-higher dimensionality. Thus, a point forms a boundary on a line (two points bound a segment of a line). Lines can form boundaries on surfaces and surfaces can bound 3D space (enclose a volume).

In SM, an object is "bounded" if it has a complete set of enclosing surfaces that restrict it to a finite volume. An unbounded object can occupy an infinite volume. For example, a cylinder without end caps is an unbounded object and has infinite volume. Because of the mathematical problems unbounded objects are often restricted or prohibited (or automatically clipped at the edges of a finite volume).

A half-space is a solid object consisting of all of the space on one side of a surface. A half plane, for example, is the space on one side of a plane. A half-space may be bounded (spheres) or unbounded (half planes and unbounded cylinder).

Geometry versus Topology

There is a fundamental distinction between the geometry and topology of a solid model stored in a database. Geometric information is the shape, size, and location of geometric elements. Topological information is the relationships between them (e.g. two faces share a common edge or two solid elements are unioned).

Consider the cube. It is composed of three types of geometric elements: points, lines and planes. Its geometric definition starts with points. They define 12 lines which, in turn, define 6 planes. The topological definition specifies how the geometric elements are bounded (forming topological elements) and how they are connected. Thus, topologically, a cube is defined as a region of space bounded by six faces which are planed bounded by edges and connected to each other at edges. The edges are, in turn, lines bounded by vertices (points of intersection) and are connected to each other at the vertices. Note that the topological definition does not change if the geometric definition is modified (e.g. vertices moved to form a rectangular box or other shape). It should be noted that this distinction between geometry and topology exists in other areas of engineering. For example, in piping and electrical systems, geometric information is the "layout", while the topological connections are represented in the "schematic"

Evaluated Representation

Within an SM database, geometric elements can be represented explicitly (mathematically) or they may be defined implicitly by relationships between other geometric elements. A curve, for example, could be defined explicitly by an equation or implicitly as the intersection of two surfaces. An evaluated database is one in which the faces, edges, and vertices of solid objects are explicitly represented.

In practice, a database may be more or less evaluated. Often the actual mathematical definition of an element is computed from other elements in the database only when needed (saving overall processing and memory). For example, the equation of an edge may be computed from its two endpoints when needed for display and then discarded.

Manifold versus Nonmanifold objects

Mathematicians separate solids into manifold and nonmanifold objects. With a manifold object (with manifold topology), every surface point has a neighborhood (an infinitesimal sphere around it) that can be mathematically deformed into a locally planar surface. In other words, if the object was made of a perfectly elastic material, a small spherical region around any surface point could be flattened into a tiny flat disk without cutting or tearing the material. For example, one could imagine

that a point on an edge of a rubber cube could find itself on a locally planar surface if it were pressed by a flat instrument.

While it may seem that all solid objects are manifold objects, this is not the case when modeling solids mathematically. Self-intersection is an example. Suppose edges A and B of the object were moved together so they became one edge (Fig. 3.1). The common edge would now bound four faces rather than the normal two. Given a point on this combined edge, its neighborhood could not be deformed into a planar surface on the solid because there are more than two faces attached to it. Self-interaction has caused it to become a nonmanifold object.

Much more difficult nonmanifold situations can arise, especially when curved surfaces are allowed. Since some SM systems function properly only with manifold objects, such situations can cause problems. Often, checks are performed during or after operations in an attempt to detect illegal occurrences.

Various models

SM systems maintain several representations of objects. The solid model used by fundamental modeling algorithms is the primary or working model. Others are secondary models and may not be solid representations. Often, secondary models are derived from the primary model to facilitate a specific function such as display, analysis (FEM, mass properties, kinematic analysis), manufacturing, documentation or data exchange. Secondary files are stored along with the primary model to avoid time-consuming regeneration when they are needed.

The ability to establish a direct connection between models in a database is called associativity. For example, changes in a solid model might trigger an immediate update in related drawings (top-down model translation). In some systems, parameters embedded in drawings are used to define solid models. A modification to a drawing can thus result in changes to the associated solid.

Often the steps used in building and modifying an object are saved by an SM database in a journal or log file. This can be considered a secondary model. It can be used for documentation, to re-create a part at a later time and to support an "undo" capability (to correct errors and mistakes).

A secondary model that facilitates display is called a display model. This could, for example, be a list of lines and text used to regenerate a

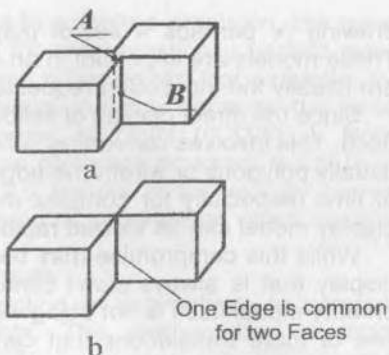


Fig. 3.1. Manifold object becoming nonmanifold: a – manifold object; b – nonmanifold object

drawing or perhaps a set of polygons for rendering a shaded image. These models are important in an SM database because display requests are usually the most often requested operations.

Since the direct display of solids is slow, a two-stage approach is often used. This involves converting solid objects into a nonsolid display model, usually polygons or wireframe edges, this may take a considerable period of time, especially for complex models. When completed, however, the display model can be viewed rapidly.

While this compromise may be more desirable than the alternative (a display that is always slow) confusion can result from the fact that the primary model itself is not being viewed. The display model is the result of one or more translations that can introduce various subtle (and not-to-subtle) changes. For example, the display model is usually an approximation using linear elements (lines or polygons). When viewed close up, curves appear straight and curved surfaces appear flat. Small features may not be represented at all.

Models portability

It is often desirable to move solid models between different SM systems. Even with the availability of standard conversion formats this may be impossible if the solid representations have incompatible types of elements (although it may be possible with lower-level models such as surfaces, wireframes and drawings). Sometimes only subsets of a model can be successfully translated, or subtle changes are introduced in certain geometric entities.

It is possible to change a model by simply moving it to a different type of computer (with the same SM system) because of differences in numeric representations. If such cross-platform transfers are contemplated, it would be advisable to obtain a written statement from the vendor specifying under what conditions this could occur, what steps should be taken to avoid it, how such changes could be detected and what problems could arise later.

While much has been written about advantages of particular SM representations, perhaps the factor most critical to ultimate success is the development of mathematical methods to implement fundamental SM operations.

Exact versus Inexact Mathematics

Computations of any kind can usually be classified as being exact or inexact depending on whether the result could differ from a mathematically perfect computation (having infinite precision). Integer addition and multiplication (e.g., $6 \times 20 + 30 = 150$), for example, are exact (assuming, of course, that the capabilities of the computer are not exceeded). With such operations the use of a computer with greater precision would not change the result. On the other hand, integer division (e.g., $1/3$) and most floating-point operations (those involving real numbers such as 2.531) are inexact.

Because numbers are represented to a limited precision, the result of inexact operations can differ from the mathematically perfect result by a small amount due to round-off and other errors. For example, the computation of 1 divided by 3 is always inexact, because the exact value would require an infinite number of digits (0.3333...). Note, however, that an inexact operation can generate an exact result (e.g., the floating-point division of 1.0 by 2.0 results in the exactly correct result 0.5). They just can't be guaranteed to generate an exact result in all cases.

Analytical versus Numerical Methods

An SM database relies on a collection of subroutines to compute information about geometric elements. The mathematical methods employed by such routines can usually be divided into two classes: analytical and numerical.

Analytical methods use mathematical formulas to directly compute a solution. For example, the edge of intersection between two planes (a straight line) is easily computed using analytical methods found in elementary geometry textbooks. Modern SM systems, however, use advanced modeling methods (e.g., NURBS) to represent sculptured surfaces. Unfortunately, there is often no known method (or no practical method) to analytically perform fundamental operations on such representations. For example, computing the curve of intersection of two bicubic patches is of great practical importance in SM. While it is possible, in theory, to use analytical methods, the resulting equation would be of degree 324! Since no formula is known for directly solving equations above degree 4, such methods are unusable.

In such cases, numerical methods must be used. This usually involves using repetitive mathematical procedures that attempt to converge on an approximate solution. In the case of intersecting bicubic patches, for example, numerical methods are employed to find approximations of points on the edge. This typically involves the execution of iterative procedures to generate points of increasing accuracy as the computations proceed. A series of such points can then be used to generate a spline curve representing the edge. This curve is an approximation and will not, in general, lie exactly on either patch. While this can cause errors, with sufficient processing the deviation can usually be made small enough to be essentially negligible for most purposes.

Unfortunately, numerical methods can fail to find a solution or fail to find all possible solutions. Also, they sometimes exhibit numerical instability. For example, instead of converging on a point of intersection, the procedure may try to iterate forever between two points on opposite sides of the edge. These problems are aggravated when geometric situations become complex.

4. GRAPHS AND TREES

Topological relationships are usually represented in a database with graphs. As shown in Fig. 4.1a, a graph is a set of *nodes* linked by *connections*. Connections are usually called *edges* by mathematicians. This terminology is not used here because of potential confusion with object edges. Nodes represent items, while connections represent relationships. A path is a sequence of nonrepeated connections between two nodes.

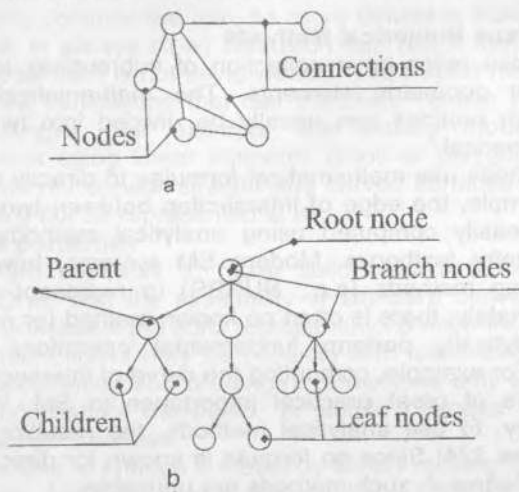


Fig. 4.1. Sample graph and tree: a – a graph is a collection of nodes which represent distinct items with any number of connections that represent relationships between items; b – a tree is a restricted form of graph

A tree is a restricted form of graph in which all nodes spread out from a single "root" node and there are no paths that start and end in the same node (no "loops"). As shown in Fig. 4.1, trees are traditionally drawn with the root node at the top. A node immediately below another node is a *child* (the upper node is *its parent*). Continuing this analogy, all the nodes above a node (back to and including the root) are its ancestors and all nodes below it are its descendants. A node and its descendants form a subtree (the node itself is the root of the subtree). Nodes with no children are *leaf* or *terminal* nodes while those with more than one child are *branch* nodes. There are no loops in a tree (a path of connections which leave and return to the same node without crossing the same connection twice).

Constructive Solid Geometry (CSG)

CSG is one of two major schemes currently used for representing solids. An object is constructed using boolean operations (UNION, INTERSECTION, and SUBTRACTION) to combine simple solid shapes

112537 u

(spheres, blocks, cylinders, etc.). Such geometric elements are called primitive solids or simply primitives. As shown in Fig. 4.2, a, CSG objects are usually stored using a tree database structure. Leaf nodes represent primitives, and branch nodes represent boolean operations. Note that each subtree also represents a legitimate solid.

This "classical" CSG tree can be extended by the use of transformation nodes. As illustrated in Fig. 4.5,b, they can be used to change the location and orientation of an object or some part of an object represented by a subtree. This provides for the independent design of a part and its later incorporation into a larger object or assembly (as a subtree). Rather than using separate nodes, often geometric transformations are simply incorporated into each node. Nodes T_2 and T_3 each transform a single primitive, while T_1 transforms the entire object. Auxiliary information such as material type or previously computed mass-properties data can also be attached to nodes.

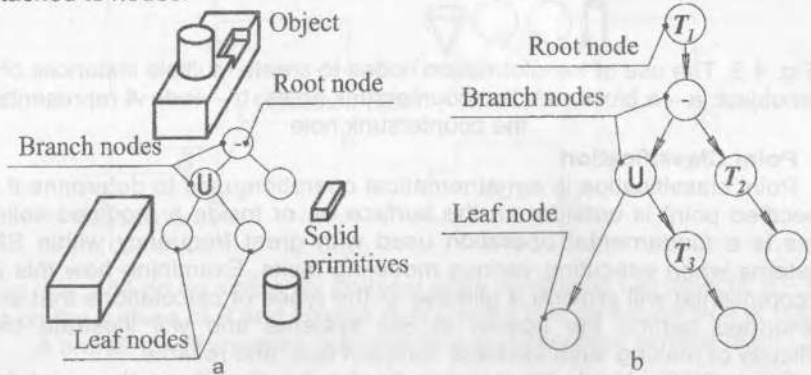


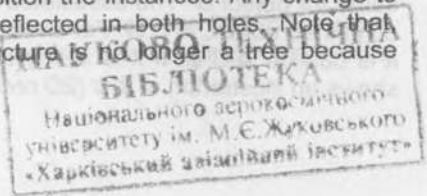
Fig. 4.2. CSG data structure: a – a tree structure is used to represent a three-primitive CSG object; b – the CSG tree can be further extended with transformation nodes

Supported CSG Primitives

The types of solid geometric elements that can be used as primitives in a particular SM system is an important consideration. Sometimes they are restricted to planar and quadric shapes to simplify the underlying mathematics. More advanced SM systems allow the use of primitives with sculptured surfaces (usually requiring cubic or higher-degree equations).

Multiple Use of Subtrees

A powerful CSG feature is the use of multiple instances of a part. Fig. 4.3 shows a subtree (node A and its two children) used to represent a countersunk hole. It is subtracted from a block in two places using transformation nodes (T_1 and T_2) to position the instances. Any change to the subtree definition is immediately reflected in both holes. Note that, technically, the overall CSG graph structure is no longer a tree because node A has two parent nodes.



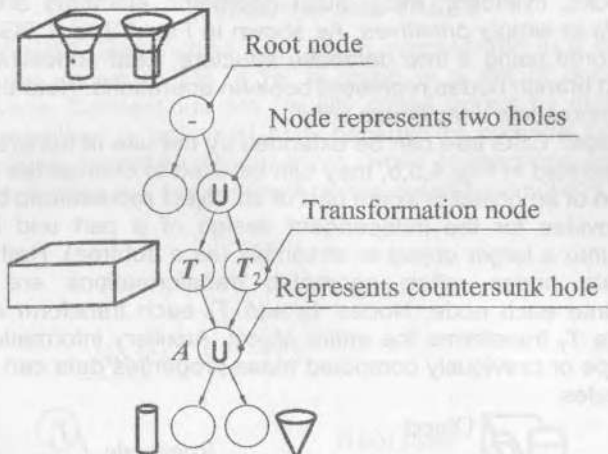


Fig. 4.3. The use of transformation nodes to create multiple instances of an object: a – a block with two countersunk holes; b – node **A** represents the countersunk hole

Point Classification

Point classification is a mathematical operation used to determine if a specified point is outside, on the surface of, or inside a modeled solid. This is a fundamental operation used with great frequency within SM systems when executing various modeling tasks. Examining how this is accomplished will provide a glimpse at the types of calculations that are performed behind the screen in SM systems and will illustrate the difficulty of making such systems compact fast, and reliable.

For a CSG object, the process begins by classifying the point for each primitive – a simple computation for most shapes. A new classification is then computed at each branch node until the final classification is determined at the root. For example, a point classification for the result of a union between two parts (subtrees), **A** and **B**, would be

	Out	on	in
Out:	out	on	in
On:	on	?	in
In:	in	in	in

Point classification for $A \cup B$

The columns, left to right, indicate whether the point has been found to be outside, on, or inside object **A**. The rows, top to bottom, indicate the same for object **B**. The associated value indicates the classification for the combined object, $A \cup B$. For example, if the point is outside both **A** and **B**, it is outside $A \cup B$. If it is inside either object, it is inside $A \cup B$. Fig. 4.4 shows an illustration of this (2D rectangles are used for simplicity).

In Fig. 4.4,a, point P is on the surface of one object (B) but outside the other (A). It is therefore on the surface of the resulting object. A problem called the "on-on ambiguity" arises when the point is on the surface of both objects (the "on on' case). In Fig 4.4b, the objects are on opposite sides of the common face (edge in this 2D example) and P finds itself inside the combined object. In Fig 4.4,c, the two objects are on the same side of the common face and P becomes a surface point. The new classification cannot be resolved with just subtree classifications (thus the "?" in the preceding case).

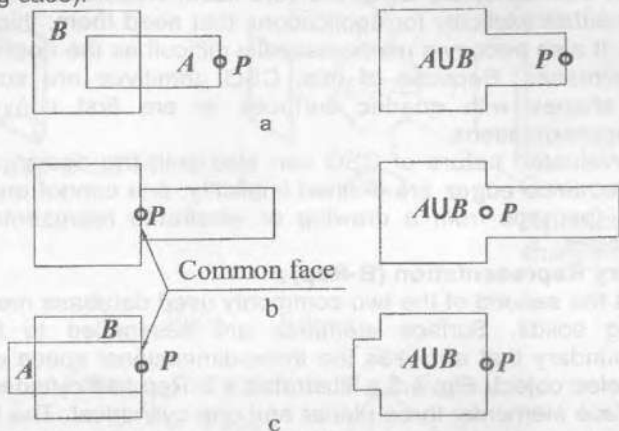


Fig. 4.4. The on on ambiguity problem while combining two objects: a – P is on the surface of B and outside A ; b – when P is on the surface of both A and B ; c – P shares a common face and becomes surface point

A solution is to have the computer analyze a spherical neighborhood of the point. The region of the sphere that is solid is computed for each primitive. This information is then combined at every CSG branch node to calculate the region or regions in the neighborhood that are solid. When combined at the root, if the entire neighborhood is solid, the point is classified as *interior*. If the neighborhood is completely empty, it is determined to be *exterior*. If it is partly solid, the point is determined to be *on the surface*. While conceptually simple, implementing this scheme is quite complex. One can appreciate the difficulty of maintaining and combining these "neighborhood balls" if, for example, the vertex points of many randomly oriented cones were to meet at the point being tested.

CSG Pros

1. CSG is a powerful, high-level representation scheme.
2. CSG models can be created with a minimum of steps, are compact, and are always valid (since they're built from solid elements).
3. Boolean operations are simple to implement and take little processing time (a new branch node for the resulting tree is simply generated and connected to the two CSG trees).

4. Many designers find the addition and subtraction of solid primitives to be an intuitive design paradigm that parallels manufacturing operations (welding, drilling, etc.). This lends itself to top-down design approaches.

CSG Cons

A pure CSG representation is severely limited in most SM situations, however, because it is unevaluated. An object's faces, edges, and vertices are not available in explicit form (mathematical equations). They are implicitly defined by the CSG structure itself. While such information can be computed explicitly for applications that need them, this is time-consuming. It also becomes mathematically difficult as the degree of the surfaces increases. Because of this, CSG primitives are sometimes limited to shapes with quadric surfaces or are first converted to polygonal approximations.

The unevaluated nature of CSG can also limit the design process. For example, since edges are defined implicitly, one cannot easily start with edges (perhaps from a drawing or wireframe representation) to define an object.

Boundary Representation (B-Rep)

B-Rep is the second of the two commonly used database methods of representing solids. Surface elements are assembled to form an "airtight" boundary that encloses the three-dimensional space occupied by the modeled object. Fig. 4.5,a illustrates a B-Rep half cylinder formed by four surface elements: three planar and one cylindrical. The topology can be represented using a graph where the nodes are faces and the connections represent shared edges.

B-Rep versus Traditional Surface Modeling

It is important to understand how B-Rep differs from a conventional surface modeling scheme. While a nonsolid CAD system may represent surfaces, a B-Rep system must also guarantee that the surfaces form a complete partition of space, even after being extensively modified. This is, in practice, a major challenge. If this separation of space fails for any reason, the model becomes invalid and the SM system has made a serious error.

Winged-Edge Data Structure

In practice, the B-Rep graph illustrated in Fig. 4.5,a is usually expanded into what is known as the *winged-edge* data structure. As shown in Fig. 4.5,b, edges are defined by vertices (their endpoints). Faces are defined by loops of edges and are connected at common edges to form a partition of space. Bottom-level nodes determine geometric definition, while connections form topological definition.

Fig. 4.6 shows an expanded winged-edge representation of a simple triangular pyramid where the vertices, edges, faces, and the solid are explicitly represented. Topological elements are shown on

different levels based on their dimensionality. Bottom-level nodes represent vertices. Above this are edges. Their downward connections point to two vertices (their endpoints). Nodes at the next-higher level represent faces. Each has connections to a loop of edges forming its boundary (three each in this case). Finally, at the top level, a single node represents the 3D solid. Its connections indicate the enclosing faces.

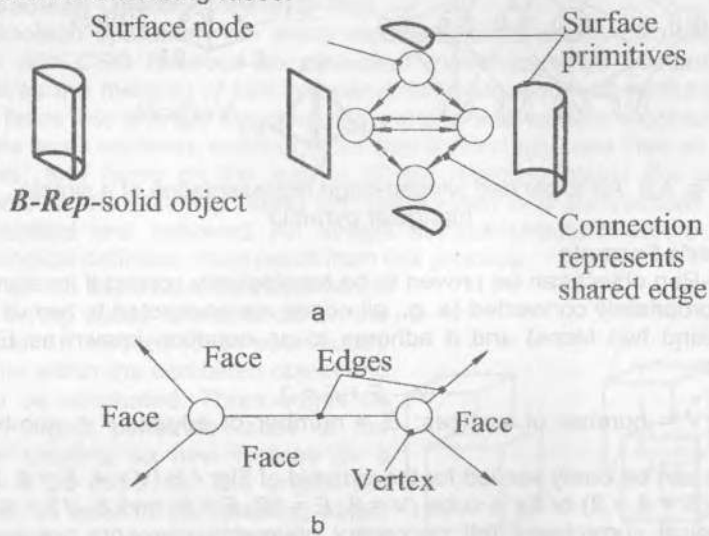


Fig. 4.5. B-Rep database representation of a solid half cylinder: a – graph where nodes represent faces (surface primitives), connections represent common edges; b – winged-edge data structure where vertices are explicitly represented as nodes, connections as edges, and faces as loops (closed paths of connections)

Faces used in B-Rep systems are orientable; that is, they have an inside surface and an outside surface. This information is typically encoded by numbering the edges in a sequence such that the right-hand rule defines a vector that points outward from the object. Note that this is used to number the loop edges of each face of the pyramid.

Note that the shape of the pyramid is completely determined by the location of its vertices. If one is moved, the attached edges and faces will move with it. This can be used to perform local shape changes on B-Rep objects (often called *tweaking*) without changing the topology.

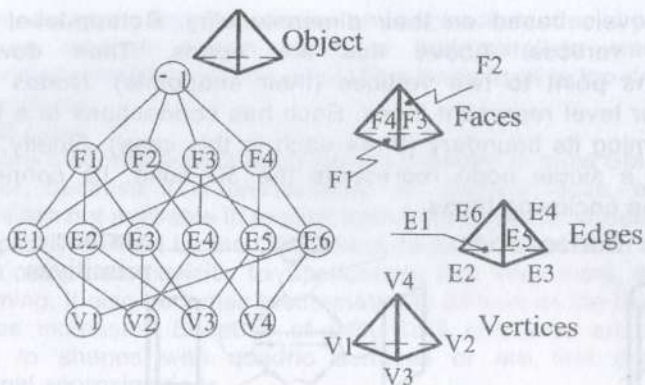


Fig. 4.6. An expanded winged-edge representation of a simple triangular pyramid

Euler's Formula

A B-Rep object can be proven to be topologically correct if its elements are appropriately connected (e. g., all edges are connected to two vertices and bound two faces) and it adheres to an equation known as *Euler's formula*

$$V - E + F = 2,$$

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

This can be easily verified for the pyramid of Fig. 4.6 ($V = 4$, $E = 6$, $F = 4$, and $4 - 6 + 4 = 2$) or for a cube ($V = 8$, $E = 12$, $F = 6$, and $8 - 12 + 6 = 2$). Topological correctness (all necessary geometric elements are present and connected properly) does not guarantee a valid solid, however. An example is shown in Fig. 4.7. If the geometry is changed such that a vertex inadvertently pierces a face (see Fig. 4.7,c), the topological model no

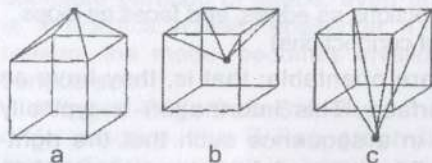


Fig. 4.7. Local surface modifications:
a – original object; b – surface modification that result in valid objects;
c – surface modification that results in an invalid object

longer represents the actual shape of the object. The four top face S intersect the bottom face resulting in new vertices edges and faces that are not represented in the original topological model. Should the new pyramid region below the original bottom be empty (forming a hole through the block) or solid? The geometric and topological definitions of the model are contradictory, resulting in an invalid "nonsense" object.

While CSG methods lend themselves to top-down design, the classical method of generating B-Rep objects is a bottom-up construction from vertices, edges, and faces. Internal database operators are used to make

incremental topological and geometric modifications. These topology operators are formulated to ensure that Euler's formula is not violated, and are commonly called *Euler operators*. Although powerful and effective, Euler operators are complex, operate at a low level, and are relatively difficult for a designer to use. In many B-Rep systems they are used internally to implement high-level operators (e. g., set operations) and are not directly available for general use.

Boolean Operations with B-Rep

Boolean operations are much more difficult to accomplish with B-Rep than with CSG because an evaluated model must be computed. This requires the merging of two interpenetrating structures of vertices, edges, and faces into a single structure. New edges and vertices must be created where faces intersect, existing faces split accordingly, and then all vertices, edges, and faces on the interior of the resulting object (for union) or exterior to the resulting object (for intersection and subtraction) must be dismantled and removed. An airtight surface structure with a flawless topological definition must result from this process.

Fig. 4.8 illustrates the result when two B-Rep cubes are unioned. Of the original 16 vertices (8 on each cube), two lie within the combined object and must be eliminated. Three edges on each object penetrate faces of the other creating six new vertices for a total of 20 ($16-2+6$). The original 24 edges all exist in the resulting object (although 6 are shortened). Six additional edges are generated where three faces on one cube intersect two faces each on the other, resulting in 30 edges ($24+6$). Six of the original square faces become six-sided polygons, but the total number remains the same at 12. A quick calculation of Euler's formula ($20-30+12 = 2$) confirms that the resulting object has the correct number of topological elements.

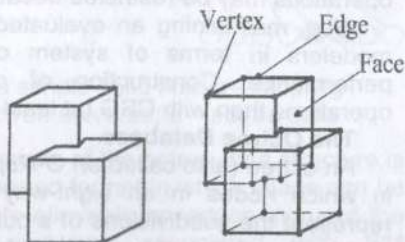


Fig. 4.8. The result when two B-Rep cubes are unioned

Three fundamental problems can be seen to arise with B-Rep boolean operations. These are as follows:

1. The pairing of geometric elements to find intersections exhibits quadratic growth in the number of elements. As mentioned previously, this causes processing time to grow much faster than object complexity.
2. The new topological model is derived from geometric calculations. Errors in determining the topology can have unpredictable results.
3. If the two objects met at a common geometric element (e.g., an edge) a nonmanifold object can result (an illegal situation in some systems).

Although various techniques are used to reduce the occurrence or correct the results of these situations, they remain a continuing problem in B-Rep systems.

B-Rep Pros and Cons

B-Rep database structures exhibit the benefits of being an evaluated representation. Edges and faces are readily available for display or other uses and can be specified directly or from drawings when constructing objects (rather than implied by intersections as with CSG). Also, higher-degree edges can be readily represented (they do not need to be computed from intersections). In some situations the ability to make local changes to topology and geometry (tweaking) is a major advantage. Also, B-Rep is well known for its effective modeling of swept objects.

The fundamental problem with the B-Rep database structure is the inability of the representation itself to guarantee object validity. Unlike CSG, B-Rep validity depends on the correctness of modeler algorithms, the quality of their implementation, and their sensitivity to the subtle computational errors that will inevitably occur. This has been found to be a major challenge with B-Reps. Applying internal constraints to prevent the occurrence of such problem situations (e.g., enforcing a fixed topology) can greatly increase reliability but reduces system utility (e.g., boolean operations may be restricted because they modify topology).

Also, maintaining an evaluated database can be a burden for B-Rep modelers in terms of system complexity, storage requirements, and performance. Construction of objects usually requires many more operations than with CSG (at least internally).

The Octree Database

An octree (also called an *O-Rep*) is an approximate solid representation in which nodes in an eight-way branching tree structure are used to represent the subdivisions of a cubical universe. As shown in Fig. 4.9, the root node of the tree represents the entire universe. There are eight children at the next-lower level representing the eight cubical octants of the universe. This subdivision process can continue to any needed level of resolution. Note that octrees are hierarchical in that the parent and its children represent the same region of space (the children being at a higher resolution).

Octree nodes are given one of three possible values: black, white, or gray. White (**W**) indicates that no part of the object exists within the represented space. Black (**B**) indicates that its space is completely occupied by the object. Otherwise, it is partly occupied (at least part of a surface is within the region) and the node is gray (**G**).

In the example of Fig. 4.9, a simple solid box is represented. Since it partly occupies the universe, the root is a **G** node. The universe is then subdivided. At this level two nodes, 4 and 5, are partly occupied and are given a **G** value. The rest are **W** nodes. The subdivision of the **G** nodes continues. One child of node 4 is completely occupied by the object and is made a **B** node. Two children of node 5 become **B** nodes. The rest are empty. Of significance is the fact that **W** and **B** nodes need not be subdivided because no additional information would be represented.

Because of this it can be shown that the number of nodes in an octree is on the order of (limited by) the surface area of the object modeled.

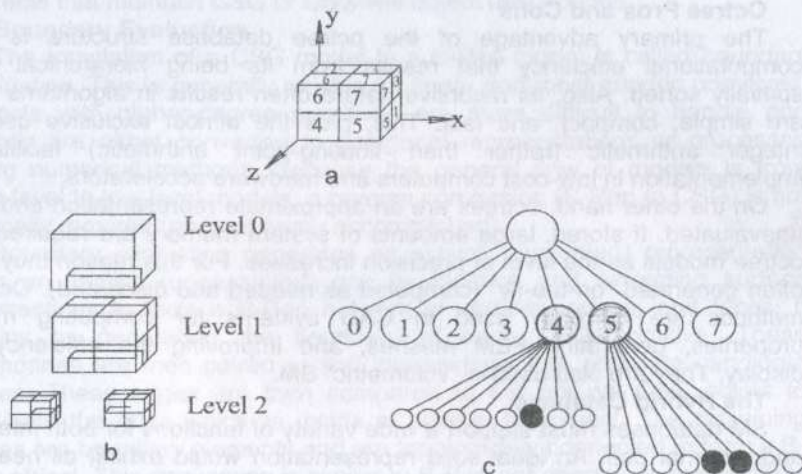


Fig. 4.9. An octree representation of a simple solid box: a – labeling of octree octants; b – subdivision of octree universe; c – octree data

Perhaps the most important characteristic of the octree data structure is that it maintains a model in a spatially sorted format in which space can be searched efficiently. This can be used to eliminate or reduce many of the adverse computational complexity problems associated with SM operations.

Point classification is simple and fast (the octree is simply traversed to the appropriate leaf node). Set operations are also fast because there is no need to generate and examine all the possible pairs of elements between two objects. The two octrees are simply traversed simultaneously and compared node by node to generate a third octree. Display is fast because only the octree nodes on the outer visible surface of an object (from any particular viewpoint) need be accessed from an octree stored in memory (or generated from another solid representation) to produce an image.

Since they are hierarchical, octree algorithms can employ a "coarse-to-fine" processing strategy where low-resolution information is used to determine where the use of higher-resolution information is necessary. During interference detection, for example, progressively higher-resolution regions of the octrees are examined only in regions where there is a continuing intersection. This stops when an intersection is actually found (B nodes in both octrees for the same space) or it is determined that none exists (or a preset resolution limit is reached). Situations where objects are separated by wide gaps or where they intersect in large regions are

resolved quickly, while more difficult situations require additional processing.

Octree Pros and Cons

The primary advantage of the octree database structure is the computational efficiency that results from its being hierarchical and spatially sorted. Also, its recursive nature often results in algorithms that are simple, compact, and fast. This, plus the almost exclusive use of integer arithmetic (rather than floating-point arithmetic) facilitates implementation in low-cost computers and hardware accelerators.

On the other hand, octrees are an approximate representation and are unevaluated. If stored, large amounts of system memory are required for octree models as the level of precision increases. For this reason they are often generated "on-the-fly" (computed as needed and discarded). Octree methods are currently used in CAD systems for computing mass properties, generating FEM meshes, and improving the efficiency of display. They are also used in "volumetric" SM.

The Hybrid Database

SM databases must support a wide variety of functions for both internal and external use. An ideal solid representation would exhibit all needed characteristics to effectively support them. Unfortunately, all existing database methods are deficient in certain critical areas. CSG supports high-level design, but the lower-level details (edges, vertices, etc.) are not readily available. B-Rep explicitly represents the details but lacks some of the design power and efficiency of working with solid elements and high-level tools. Plus, allowing a designer to directly manipulate low-level elements introduces reliability problems. Octrees can be seen as a "least common denominator" approximation that any solid can be converted into. While not suitable as a primary CAD representation, octrees can be effective in facilitating SM operations. They fulfill a role in SM similar to that of polygons in surface modeling.

This situation has led to the development of hybrid modeling schemes in which multiple database representations are employed in order to, hopefully, realize the strengths of each. The actual definition of a hybrid database is debatable because virtually all SM systems make use of secondary models for various purposes (e.g., polygon display lists). For this discussion, a modeler will be considered a hybrid if it employs two or more substantially different internal database representations that are valid solids and are actively maintained to support modeling functions.

The use of multiple representations has several undesirable aspects. The conversion between two formats may be difficult, impossible, inexact, or introduce subtle changes (potentially introducing serious inconsistencies). Because of this, conversion is often in one direction only (e.g., CSG to B-Rep). The use of multiple models also increases the size of programs and storage and increases processing requirements.

With most hybrid modelers the general strategy is to support a high-level CSG representation which can then be internally translated into a

lower-level B-Rep model. Actual implementations range primarily from CSG systems that use B-Rep to facilitate display, to primarily B-Rep systems that maintain CSG or CSG-like object descriptions.

Boundary Evaluation

The translation of a CSG model to a B-Rep model is called *boundary evaluation*. This is generally reliable for linear and most quadric primitives. Objects with higher-degree primitives are more difficult to handle and usually are either converted to polygonal representations or processed using numerical methods. Because the general flow of models is from high-level to low-level models, a reverse conversion (B-Rep to CSG) is not normally required (and not easily accomplished).

Boundary evaluation generates an explicit (evaluated) representation from an implicit (unevaluated) one by locating and extracting surface intersections to form a B-Rep network of vertices, edges, and faces. It begins by generating the boundary of each CSG primitive. These boundaries are then paired to find intersections that generate candidate edges. These edges are then compared to the original CSG object to eliminate the ones which lie inside or outside the object. The remaining edges are boundary elements and are topologically connected to form a B-Rep object. This conversion is similar to B-Rep set operations and exhibits similar computational characteristics. The pairing of primitives is a quadratic growth operation and a large number of candidate edge segments may be generated as edges are subdivided (wherever they intersect a face).

Using Octrees

The need for greater interaction with solid models has sometimes led to the addition of an octree representation into the hybrid mix. With this scheme, an evaluated model is not needed for display (CSG models are displayed directly from octrees). To save computations, a solid model can be partly evaluated as needed by a particular operation (e.g., find the edge of intersection of two CSG primitives). Nonsolid entities such as surfaces, curves, and points can also be directly displayed, providing a powerful visualization capability.

Parametric/Relational Database

In this discussion so far, solid models have been considered to be static shapes that were created by a series of operations requested by an operator. Parametric/relational design attempts to improve on this situation by making models flexible. Here we will discuss the impact such technologies have on database structures. Basically, parametric models are defined by a set of parameters that can range from simple dimensional values (e. g., radius of a specific fillet) to global parameters that have an effect on an entire design (e. g., length of a crankshaft). The goal is to somehow capture the designer's "intent" in the model.

Variational Design

A related technique is referred to as *variational design*. The parameters and constraints are encoded into a set of simultaneous equations. Instead

of replaying a sequential design procedure, the equations are solved to determine a set of parameters to define the shape. This can be used for optimization and has been found to be especially helpful in refining 2D designs based on sketches. As the number of equations and their mathematical sophistication increases, however, finding solutions becomes more difficult and slow. Overconstrained situations can arise and designs can exhibit strange behavior.

Feature-Based Design

Feature-based design is another methodology that impacts the parametric/relational SM database. Essentially, this allows the designer to specify form features (e. g. fillets and flanges) using familiar engineering terms. If a through-hole was specified, for example, the bottom face of a subtracted cylinder would be automatically constrained to lie on the bottom of the object.

While popular with designers, form features are difficult to implement because it is not possible to simply define geometric elements in isolation and then add or subtract them in the database. Their shape depends on finding and analyzing a specific situation existing in the model. For example, a simple chamfer may require two planar faces meeting at a straight edge with sufficient space on each to locate the new plane (without hitting anything else).

As procedures that implement form features become more sophisticated and powerful, the programs that implement them must exhibit greater degrees of intelligence and the ability to perform spatial reasoning. It is often a major challenge for an SM database to support such capabilities. Because geometric elements such as vertices and edges play an important role, an evaluated representation (B-Rep or hybrid) is usually necessary.

5. CURVES, SURFACES, AND COMPUTERS

The problem of representing curves in computers derives from their nature. Curves are by definition continuous; digital computers comprehend a discontinuous framework for the representation of numbers. You cannot directly store a curve in a computer; you must reduce it either to a series of points whose coordinates are stored or to a mathematical equation—an algebraic expression. Storing curves as points is, in general, impractical; too many are needed to enable the computer to reproduce the curve smoothly.

Also, the correspondence between the geometry of curves and the algebra of their representations is tricky. Curves that appeal aesthetically to designers are often difficult to represent mathematically. And the intersection of curve-based surfaces with each other can yield curves that are even more difficult to represent.

Moreover, the manipulation of curved surfaces within the computer is normally accomplished through the manipulation of control points, whose

locations define the curves that prescribe the surface. However, many of the most popular surface-defining curves have their control points located off the surface, which tends to invite confusion.

This problem first appeared when aircraft designers attempted to produce mathematical definitions of surfaces that intersected a given set of points or a given set of curves. Powerful techniques falling under the general heading of parametric surfaces were developed to deal with these issues. But no single technique is applicable to every circumstance.

Curve and Surface Usage

More than any other aspect, the construction of curves and surfaces in SM systems reflects the history of manual approaches to the design of shapes. Motivation for the current approaches derives from two principal sources: automotive and aerospace design.

Automotive designers were – and still are – primarily concerned with aesthetics. In recent years, aerodynamics and their effects on fuel economy and speed have begun to be taken into account as well. But the look of the automobile, and especially the character of the reflective qualities of its surfaces, dominate automotive design parameters. Consumer products, such as kitchen appliances and home-use electric hand tools, follow a similar pattern.

Airplane designers are interested in putting an aerodynamically shaped skin around a tightly constrained physical design in which structural strength, function, weight, size, cost, and other factors are traded off against one another to produce a design solution. Like ship designers of old, the airplane designer is given a set of sections that must be smoothly interconnected by surfaces, or a set of edges for which a "patch" must be designed so that the resultant shape is smooth. This process is termed *lofting*.

Automotive designers start with artistic sketches. The design progresses through many iterations of review before any attempt is made to convert the artistic conception to 3D geometry. Airplane designs, on the other hand, move rapidly to a phase in which the precise mathematical nature of the surfaces becomes a concern.

When a design is initiated analytically, the result is usually a mesh of precisely defined points which must be interpolated into a surface. There is no general surface that can be made to fit an arbitrary collection of points smoothly; that is the main reason for the existence of a variety of mathematical surface types.

Another reason for the confusing proliferation of mathematical surface types is the search for a variety whose construction is "natural" for a particular category of designers. Bezier curves and surfaces, for example, are considered "natural" by designers whose primary initial consideration is the matching of a curve or surface to a predetermined set of points. However, since the control points for these forms lie off the curve or surface, many other designers find them unnatural.

Curves and Splines

A sample of the types of curves that SM systems ought to produce include:

1. Straight lines
2. Conic sections: circles, parabolas, hyperbolas, ellipses
3. Superconics: generalization of conics in which the degree of the terms in the defining polynomial differs from 2.
4. Free-form parametric curves: B-splines, NURBS, etc.
5. Intersection curves: curves resulting from the intersection of two surfaces (Fig. 5.1).

Splines are by definition flexible pieces of wood or metal used by ship and aircraft designers to draw smooth curves. Spline curves are curves that run smoothly through an empirically selected set of points.

Bezier, B-Splines, and NURBS

At Renault in the 1960s, Pierre Bezier developed a definition of a system of curves that combines features of interpolating and approximating polynomials. (Interpolating polynomials pass through a given set of control points; approximating polynomials pass near the control points.) In a Bezier curve, the first and last control point precisely define the endpoints of the curve; intermediate control points influence the path of the curve; and the first two and last two control points define lines which are tangent to the beginning and end of the curve. Bezier curves can therefore blend nicely together, as can Bezier surfaces. The point at which two curves (not only Bezier) are joined is called a *knot*.

Blended Bezier curves give good local control; if a control point is moved, the curve changes only in the immediate vicinity of the control point. But construction constraints can sometimes be restrictive for the designer; under such circumstances, B-spline curves may be useful.

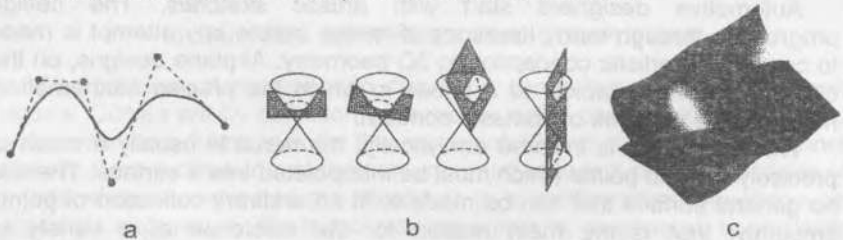





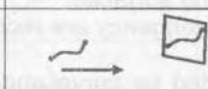
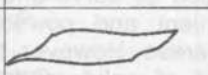







Fig. 5.1. Curve types: a – free-form parametric (**NURBS**); b – conics; c – intersections

B-spline curves, unlike Bezier curves, can employ an arbitrary number of control points. They are generated by multiplying an approximating function in terms of the parameter u , where $0 < u < 1$, by a matrix containing a subset of the control points in the vicinity of the B-spline curve.

Since conic sections are common in manufactured forms, using rational B-splines often makes sense in this context; they run precisely through the defining points of the conic sections. Symmetrical forms yield a uniform distribution of knots. But when combining B-spline curves and surfaces, knots may appear in nonuniform distribution. The family of curves used for creating such shapes has come to be known as NURBS, or nonuniform rational B-splines (Table 5.1).

Table 5.1

Surface Techniques

Curve techniques		Description
Through points		A curve that passes through multiple points
Control points		A curve weighted by control points
Concatenate		Two or more curves joined end to end
Project		Curve projected onto a plane or surface
Edge		Curve generated from existing surface boundary
Surface intersect		Curve of intersection between two planes or surfaces
Silhouette		Curve generated from the silhouette of existing surface
Through tangent		Curve whose end points are tangent to given vectors
Offset		Curve offset from existing curve
Blend		Curve blend from two existing curves
Isoline		Curve generated from isoline of existing surface
Segment		Curve generated from existing curve segment

6. SURFACE TECHNIQUES AND CONCERNS

SM Concerns

1. Watch for surfaces that have the potential to self-intersect. These include sweeps due to trajectory, revolves due to rotation angle, rules due to curve relationships, and offsets due to curvature.
2. All surface areas have seams (as in a revolve) or boundary curves. When positioned in a resulting solid topology, these seams and boundaries should be located away from other edges or vertices. Vertices or edges should never coexist within system tolerances unless they in fact share common boundaries.
3. Surfaces generated directly from boundary curves such as rules and patches are preferred over surfaces that are excessively trimmed.
4. When possible, always share existing vertices and edges between surfaces.
5. Techniques such as lofting, meshing, and patching approximate surface curvature between defining boundaries.
6. If possible, when techniques generate new boundaries on surfaces (such as with fillets), reconstruct mating surfaces from the new boundaries rather than trimming existing surfaces.
7. Techniques that maintain surface tangency are recommended.

When Trimming Surfaces

Surfaces may be trimmed or divided by curve and/or other surface boundaries. This is a very convenient and powerful technique to seemingly discard unwanted surface areas. However, trimmed surfaces pose concerns for the development of solid models and for data exchange through neutral formats.

Many trimmed surfaces in a single model will increase database size and complexity since more information is required for storage (original surface, trim boundaries, and resulting trimmed surface). Also, the resulting surface edge is dependent on the accuracy and dependability of the system's surface-trimming +functions. Airtight surface enclosures are required for solids, so always check trim boundaries that should mate with other surfaces.













When Extending Surfaces

Surfaces may be extended in either or both the U and V directions. Extending surfaces is very helpful during the development of surface geometry (for example, to establish an additional section curve further away from a surface's current boundary or to allow the continued tangency of subsequent surface functions). As a rule, surface extensions are used as a means to an end rather than an end in themselves. The development of new surface boundaries is generally the result of surface extensions.

Table 6.1 lists some of the more common techniques for constructing surface geometry in SM as well as traditional CAD systems

Table 6.1

Surface Techniques

	Sweep a curve along a specified trajectory – sweep
	Rotate a curve about a specified axis – revolve
	Rule a surface between two curves – rule
	Offset a surface by a specified distance – offset
	Surface through a regular grid – grid
	Extrude a curve along a linear trajectory – extrude
	Loft a surface between dissimilar curves – loft
	Mesh a surface through control curves – mesh
	Patch a surface between boundaries – patch
	Surface through regular or irregular control points
	A constant or variable fillet between two dissimilar surfaces – rolling ball
	A tangent corner patch between three surfaces – corner

Surface Tangency

Some surface operations (such as fillets and corners) maintain surface tangency during their construction. Such surface areas should not be reconstructed from their boundaries in lieu of trimming as recommended here for other surface areas unless subsequent tangency constraints can be applied. Even though their boundaries would be the same, the surface patch operation may not automatically take tangency into account, and the resulting surface may differ from the one it is intended to replace.

Sculpted Surfaces

Sculpted surfaces are at best painful to construct when subsequent features are to be attached or added. It is best to create any nonsculpted surfaces or features first, and then use the existing edges to create the final sculpted surfaces in place.

When Modifying Curves Locally

Surfaces may be modified locally. Any deviation that maintains a surface's current boundaries and thus the solid's current topology is acceptable unless the surface deviation internally or externally pierces another solid face. Local surface deviations take several forms. Here are a few:

1. Radius dome. A smooth radiused surface deviation computed to fit a selected surface.
2. Section dome. Replaces an entire planar surface with a blended, extruded, or swept sculptured surface.

3. Local push. Smoothly deforms a circular or rectangular surface region.

Variable Radii Blends

Sometimes a blend of four or more surfaces may be required, each with different converging fillet radii along their edges. In many cases this blend must be performed manually and would require a complex surface patch. At first, the solution (the resulting patch) may seem overwhelming to achieve. The steps provided here can be applied to both simple and complex surface blends (Fig. 6.1). The surfacing capability of some systems may eliminate the need for this procedure.

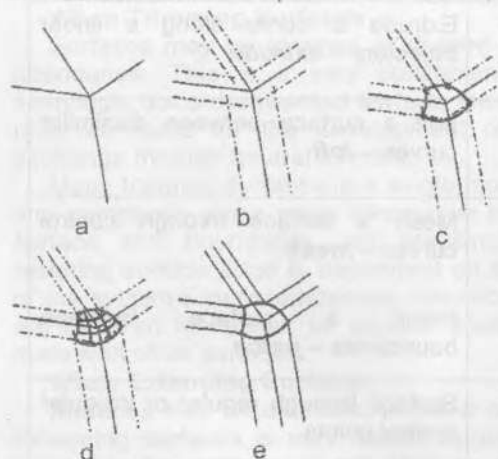


Fig. 6.1. Variable radius blend:

- a – converging surface edges;
- b – tangent lines of dissimilar fillet radii;
- c – section curves normal to fillets at intersection of tangent lines;
- d – four sided patch created from closed section curves;
- e – converging fillets trimmed or replaced

1. Start by defining the tangent lines of the converging radii. Where two intersect, a corner will be defined. Use this method to define each corner of the required patch.
2. Generate a section curve between corner points normal to and through each converging fillet surface. These curves will produce a closed boundary.
3. Use the closed boundary curves to generate the surface patch. If your SM system can apply surface tangency constraints, use them now.
4. Trim the converging fillets to the boundaries of the new surface patch. It is recommended that the fillets be replaced by ruled surfaces instead of trimming.
5. If there are more than four boundary curves, the resulting patch may be stretched or twisted. Even though this surface may not be desired, cut a section curve through it to approximate the definition of a new curve boundary and thus divide your required patch in two or more patches of three or four sides each. Use this approximated curve as a reference to generate a new curve that is tangent at its endpoints.
6. Validate the resulting surface patch to ensure that it blends smoothly along its entire surface as well as at the fillet boundaries. Here are some ways to perform this validation.
 - a. Change the display settings to show a large number of isolines on the surface patch and then rotate the model to see if the patch looks smooth.
 - b. If the preceding visual test appears to indicate a problem, generate section curves through the surface patch where it intersects the mating fillets. These curves should indicate a smooth transition.

Surface Offsets

Some desirable operations introduce mathematical complexities that have no general solution – the offset operation, so useful in the design of numerical control tool paths, is one such operation. While most SM systems support offset surfaces in one way or another, they all impose restrictions to keep the resultant surfaces within the realm of expectation while maintaining mathematical integrity.

Solid modeling Considerations

Constructing curves and surfaces for use in solid models carries an additional set of considerations above and beyond normal recommended practices for these techniques. These concerns parallel those of the NC programmer whose job depends on all surfaces being present, that they meet, and that there are no hidden cracks or gaps. A little extra care and consideration is all that is required when constructing curves and surfaces for SM.

Geometric Progression

The inherent integrity of SM functions when used properly will result in valid closed 3D volumes. When the geometry of a design becomes too complex for existing SM functions to model properly, designers are forced to utilize curve and surface techniques to supplement the model-building

process. This deviation from the SM environment carries a price. A governing rule for model validity (surface closure) must be addressed manually during construction (Fig. 6.2).

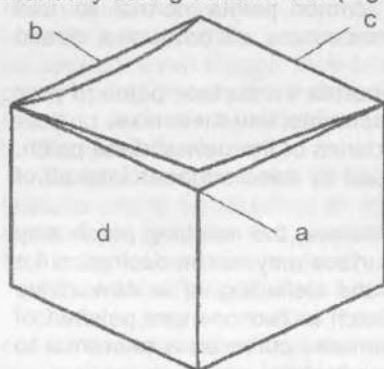


Fig. 6.2. Modeling for closure:

a – mating points should be coincident; b – mating edges should share existing boundaries; c – mating edges should have equal segments; d – surface to boundaries instead of trimming when possible

Endpoints of connecting curves should always remain coincident. Remember that during wireframe and surface operations, internal controls for model integrity and closure found in SM are not automatically applied. Develop good point and curve strategies.

Curves on Surfaces

Just as points are the defining elements for curves, curves are the defining elements for surfaces. Whenever possible, always generate curves from existing surfaces. Be aware of how local curve modifications can affect associated surface geometry. Always use simple curves (lines and arcs) instead of splines whenever possible.

Surfaces on Solids

Surfaces must form a closed volume to define a solid and should always share curve boundaries when they meet. If possible, mating curve boundaries should always share equal segments. Difficult surface areas should always be redone rather than piecing small surface segments together. The endpoints of connecting boundary curves for mating surfaces should always remain coincident

Levels of Integration

There are many levels of integration among wireframe, surfaces, and solids. SM systems that closely integrate these three representation schemes in a common database are more flexible than those that do not.

There is a logical progression of geometric elements that, when constructed properly, will help ensure model closure. This process is referred to here as modeling for closure. Modeling for closure begins with points, and progresses to a closed solid volume. This progression includes:

1. Points on curves
2. Curves on surfaces
3. Surfaces on solids

Points on Curves

Points are the most basic geometric elements. They form the basis of all geometric curves. Ensuring model closure begins with the proper creation and manipulation of points and curves. Control points for planar curves should always remain coplanar

Usually, such systems allow the creation, joining, and exploding of surfaces into and out of solids as needed without any loss of data or integrity. Until SM functions can accurately represent and model complex surface areas, the need to closely integrate wireframe and surface functions into the SM database will remain great

Preferred Geometry

In surfacing, just as with any craft, there are preferred methods. Again, always try to construct surfaces to defined boundaries rather than performing extensive trimming. This is especially true if models are to be exported to dissimilar systems via neutral formats. Try to use ruled surfaces instead of surface patches whenever possible. Ruled surfaces require only two boundary curves, whereas a surface patch requires three or more defining boundaries. Any technique that simplifies construction and requires fewer database resources is recommended. SM databases can and will become quite large. Early simplification saves time and worries later. The techniques can't be overstressed.

The only measure of quality for curves and surfaces in an SM system is the extent to which the facilities meet the designer's needs. Systems and approaches to these issues are far too diverse at this time to admit to a linear ranking. All present products are somewhat deficient in robustness, that is, the user can create shapes that will "break" any system. Designers should know the limitations of the systems they are considering, and expect them to degrade "gracefully" when they "break," without loss of data.

Combining Associative and Nonassociative Elements

In general, the greater the extent of associativity in a model, the less you can modify the model through direct curve or surface manipulation commands. In many cases, a fully associative model can be modified only by changing the parent elements or the dimensions driving the associative elements. However, by carefully combining associative and nonassociative elements in a single model, you can create a model that is modifiable both through dimension edits and through direct manipulation commands. Four common examples follow

Nonassociative Curves and Surface

Fig. 6.3 illustrates a simple example of a nonassociative surface placed through nonassociative curves. Note that Fig. 6.3,a shows the original curves and surface. Fig. 6.3,b shows the result after one of the curves is modified. Note that the surface does not follow the modified curve. You would then have to delete the original surface and create a new one. Fig. 6.3,c shows the result of moving poles on the surface. Note that, although the surface has been modified in the middle, the edges were not modified. However, if the surface had been modified such that the edge row of poles moved, the surface would deviate from the curves

Since both the curves and the surface are nonassociative, either the curves or the surface may be modified. However, the drawback is that there is no intelligence between the surface and the curves, and if one of

the endpoints of a curve is moved, the adjacent curve will not be connected to the modified curve.

Variational Curves and Associative Surface

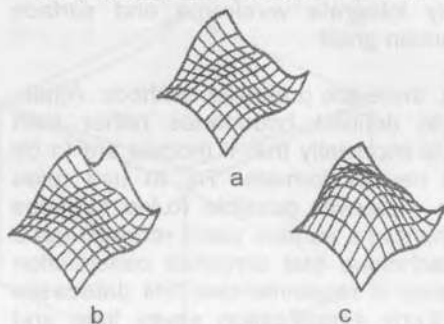


Fig. 6.3. Nonassociative curves and surface: a – original surface; b – after modifying curve; c – after modifying surface

This example shows how you can construct an associative free-form surface from fully constrained variational profiles. Although the concept of a fully constrained free-form surface may sound like a contradiction in terms, this surface is fully dimensionally constrained. Once the four profiles making up the boundaries of the surface are fully constrained, you can then modify any of the dimensions to modify the surface. In Fig. 6.4 the dimensional constraints for both cross sections are all proportional to the overall width and height of

the first cross section. By setting proportional expressions, you can then modify the surface predictably by simply changing the height and/or width. As shown in Fig. 6.4, the height is changed from 2.5 to 1.25 and the width is changed from 3.5 to 5.0. This causes the surface to update to match the new cross-sectional shapes.

There are three important items to note about the example in Fig. 6.4.

1. Since both the cross-section curves and the trace curves are variational profiles with coincidence defined between the endpoints of the curves, the curves will always remain connected at their endpoints. When the cross sections are modified, the trace curves update to remain connected. This is important

since the skinning surface would have problems if the endpoints were not connected.

2. Since the surface is associative to the curves, when the curves are modified, the surface updates accordingly.

3. Depending on the SM system used, you may not be able to modify either the curves or the surface with direct manipulation commands

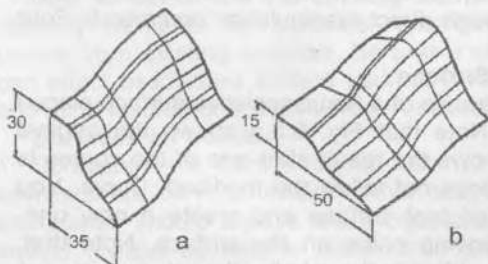


Fig. 6.4. Variational curves and associative surface: a – original surface; b – surface after modifying width and height

since they are both associative. Standard constraint modification techniques may be the only way to make changes.

Nonassociative Curves and Associative Surface

This example shows one of the most useful combinations for pure free-form modeling. In this case we created nonassociative curves and then passed an associative skinning surface through the curves. The big advantage here is that you can use any of the available wireframe modification techniques to directly manipulate the curves. The associative surface will update to reflect any changes made to the curves. As shown in Fig. 6.5, the original surface is shown, then the resulting surface after directly modifying the poles on the curves.

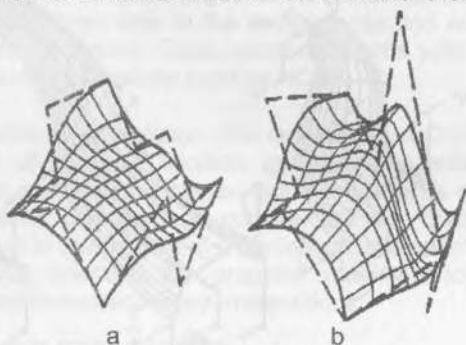


Fig. 6.5. Nonassociative curves and associative surface: a – original surface with poles of curves displayed; b – surface after moving poles on curves

Advisory Notes. The important point about the example in Fig. 6.5 is that you do not have to recreate the surface to see the result of free-form modifications to the curves. This lets you do "what-if" studies on a free-form, or aesthetic, shape without recreating surfaces. The drawback is that, depending on the SM system used, you may not be able to directly modify the surface, and you must be careful not to modify the curves so that the endpoints lose their connections.

Associative and Nonassociative Curves with Associative Surface

In this example, both associative and nonassociative curves are used to construct an associative surface. This method lets you modify some aspects of a surface through direct curve manipulation commands, and other aspects through dimension edits.

In Fig. 6.6, two variational profiles are used for the cross sections. The single trace curve is a nonassociative B-spline. The reference planes, on which the two cross-section profiles are constructed, are created through the endpoints of the B-spline. Therefore, the B-spline can be modified in any way and the cross-section profiles will still remain at the ends of the B-spline trace curve. Fig. 6.6,a shows the original curves and surface, a skinning surface created with the B-spline trace curve and two open profiles for cross sections. Fig. 6.6,b shows the result of modifying the B-spline trace curve. Fig. 6.6,c shows the subsequent result of modifying the dimensional values on the two profiles as shown.

In both cases in Fig. 6.6,b and 6.6,c, the surface updates according to the changes made, since the surface is associative to the cross sections and trace curve. The important thing to note here is that you have some flexibility in modifying the surface. You can either manipulate the trace curve or modify the profiles.

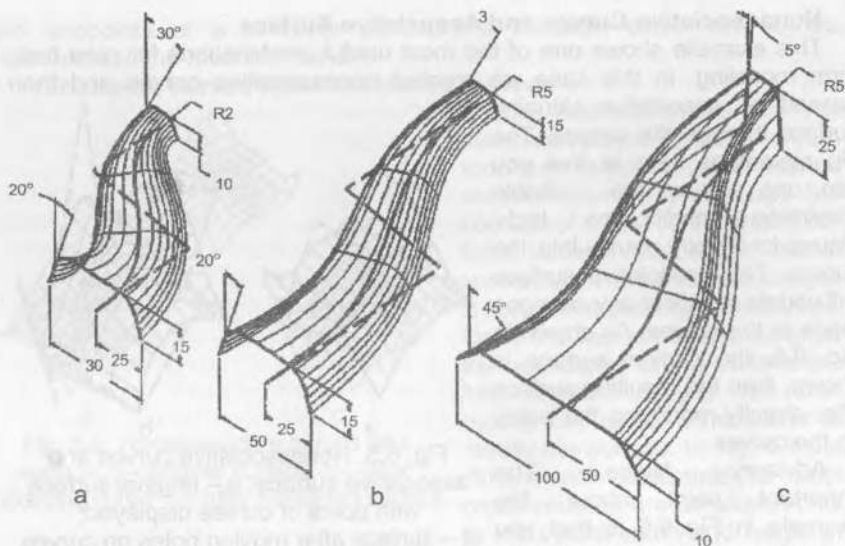


Fig. 6.6. Associative and nonassociative curves with associative surface:
 a – original curves and surface; b – result after moving poles on trace
 curve; c – result after editing dimensions

Most Effective Modeling Approach

The examples above present a few ways in which you can combine nonassociative elements, associative elements, and variational profiles into a single model, particularly as they relate to free-form surfaces. The examples given are only a few of the possible combinations. The important thing to remember is that a model does not have to be fully associative to be useful. Oftentimes the most effective modeling approach will include a combination of variational profiles and other associative and nonassociative elements.

System Tolerances

Be aware of your system's tolerance. Any deviations from your system tolerance limits may result in unacceptable gaps and cracks. Modifying your system tolerance to obtain closure is not recommended. The proper solution is to employ techniques that will allow the generation of curves and surfaces within specified tolerance limits.

Effective Color Usage

Use differences in color and layers effectively. When duplicating or projecting curves, arrange it so that the resulting new curve will be a different color or layer than the original. A newly constructed curve may differ only very slightly from an existing one. Color coding causes less confusion when a selection is required. This use of color and layering will also make it easier to identify each entity as well as to manipulate entities as a group. Blanking and deleting can be done by color and layer on most SM systems.

Use Existing Edges

When adding a feature to a solid model using surfaces or when sewing surfaces together, try to use existing and common edges for curve boundaries. This will make the model easier to change later in the design cycle and will keep gaps from appearing between surfaces. Gaps cause problems when converting to a solid and in downstream operations such as NC.

Approximation

Many SM systems employ a CSG representation. The developers of CSG systems introduced the concept of applying boolean operations to solid shapes. These work fine when all surfaces are planar. But with surfaces of any greater complexity, intersections can result in complex curves. Such curves may not be precisely defined in the context of a particular SM system. Rather than applying an analytic solution, the program must employ numerical approximation techniques to express these intersections.

7. MODELING STRATEGIES

Solid models are built using the capabilities provided by a given Solid Modeling (SM) system. Such capabilities often include solid primitives, operator-defined profiles swept along a path, features, and surfaces. SM system operators may combine these shapes in many and various ways to meet the requirements of a given design.

Every SM system must provide a base set of geometric construction capabilities apart from related functions such as drafting, analysis, parametric/relational definition, or data exchange.

Solid Modeling attributes include surface finish, threads, location tolerance, position tolerance, annotation, display color, etc. The solid model definition is incomplete without these. Please note that not all SM systems include this type of data in export files. Tolerances are so important that most SM systems treat them as a separate entity group.

Solid primitives

A primitive is a volume defined by simple, standard geometrical shapes, such as a box, cone, and cylinder. Primitives are used by the CSG modeling method. There is a limited set of ready-made primitive shapes, each requiring very little input. Primitives can be defined using simple menus and can be combined to form complex geometry. Input can be relative to local or global coordinate systems or other existing geometry.

Very complex shapes can be created by combining primitives through the use of boolean operations. The resulting complex solid can then be saved and treated as an operator-defined primitive.

A box is a right rectangular hexahedron. That is, it has six rectangular faces. Typical input required: Length, width, height, and location. The center point and orthogonal edge vectors as well as two diagonally opposite corners are specified with some systems.

A cone has a base circle a curved exterior surface tapering to a point, and an axis of revolution normal to the base circle. All lines on the curved

exterior surface between the base circumference and the apex are linear. Typical input required: Base radius, height or length, and the base center point as a location.

A frustum is typically defined as the portion of a cone that lies between two parallel intersecting planes. Typical input required: Bottom center point for location an axial direction, bottom radius, top radius, and length.

Cylinders are defined by a constant diameter bounded on each end by right-angular, parallel circles of equal radius. The axis is normal to the ends. Typical input required: Direction, base point or location, radius, length.

A prism is similar to a right rectangular box, except that it has more faces. Typical input required: Base point or location directional vector, the radius of the defining circle, and length. Prisms may be inscribed or circumscribed about the defining circle. Some systems also allow the number of equilateral edges or facets on its base to be specified.

A tetrahedron has an equilateral triangular base and three triangular sides. Typical input required: Base point or location, directional vector, the radius of a defining circle, and length or height. The base may be inscribed or circumscribed about the defining circle. A variation is known as a pyramid, defined by four sides and a rectangular base.

A sphere is the volume generated by a semicircle revolved about an axis passing through its end points. Typical input required: Spherical radius and the center or polar point.

A torus is generated by revolving a circle about an axis in the plane of the circle. The axis must not pass through the center of the circle and must lie outside the circle in most SM systems. Typical input required: Radius of the circle, the radius from the center of the circle to the axis of revolution, and the direction of the axis must be defined. A hollow tube is created when two concentric circles of differing radii are revolved.

Geometric Operations

Geometric operations in SM are volumes created using cross-sectional planar profiles and dragging functions, such as blends, extrudes, sweeps, revolves, fillets, corner rounds, and local surface deviations. It should be noted here that in an SM system these geometric operations result in a single closed solid entity. Similar operations are available in surface modeling that result in one or more individual surface entities.

These SM volumes may have constant or variable cross sections and linear, multilinear, curvilinear, or axisymmetric paths. The common naming conventions used for such entities are: extrude, sweep, blend or transition, and revolve. Fig. 7.1 illustrates common geometric operations.

Blends or Transitions

A blend or transition has a variable cross section swept along an arbitrary path as illustrated in Fig. 7.1,a. Its definition consists of two or more closed planar profiles of equal number of segments and a trajectory curve. The blending operation causes the cross section between defining profiles to be approximate. The trajectory may be a spline curve passing through the

ordered sequence of profiles. The cross section blends from segment to segment from one profile to the next. The profiles are generally normal to the trajectory, but may be allowed to be normal to another surface as the profile moves along the trajectory. Two profiles and a linear trajectory can be used to create a linear blend.

Blending Example

An engine intake manifold which is of a circular profile at the intake valve quickly blends along a tight three-dimensional curve into a rectangular profile with large fillets (due to space limitations), and then more gradually blends along an increasing radius into an elliptical profile before again becoming circular. The cross-sectional area must remain unchanged along the length, requiring accurate approximations of intermediate cross sections.

Extrusions

Solid extrusions have a constant cross section dragged along a single linear path. The profile does not rotate about the extrude line. Consequently, lines on the cross-section profile generate prismatic surfaces. The extrude operation generally uses a closed planar profile that is extruded along a line normal to the profile. In SM, planar profiles have a front and a backside. Normal to the profile means that the front of the profile is facing the direction of extrusion. The extrude line may be defined as a distance offset from the planar profile, parallel to a straight edge of the solid, or may extend between two parallel planar faces.

An outside planar profile and a nonintersecting inside planar profile on the same plane can be used to create a hollow tubelike extrusion. A single SM volume results.

Revolutions

A solid revolution is a volume created by revolving an arbitrary planar cross-section profile about an axis. The profile should be closed and not cross the axis. If the profile is open, the axis must pass through its endpoints.

Sweeps

A solid sweep is a volume created by sweeping a planar cross-section profile along an arbitrary curve. A sweep operation uses a closed planar profile that is generally normal to and swept along a multiple segmented curvilinear sweep trajectory. The adjacent segments of the sweep trajectory are usually required to be tangent.

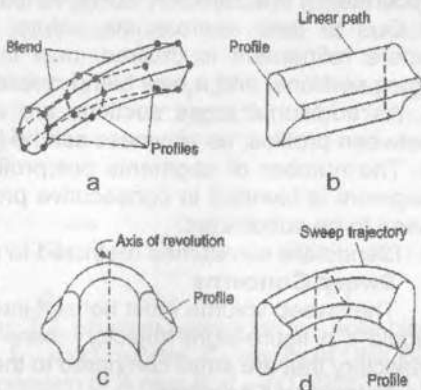


Fig. 7.1. Common geometric operations: a – blend or transition; b – extrusions; c – revolutions; d – sweeps

Sweeping a Helix

Multiple sweep trajectories are sometimes allowed. One sweep trajectory is used to define the sweep path, and another controls the profile's local axis orientation (i.e., the rotation of the profile along the trajectory). For example, the sweep trajectory may be a helix, and the local axis trajectory for a noncircular cross section may lie along the long axis of the helix. The local axis of the profile will always point toward the long axis of the helix. A helix of circular profile requires only one trajectory.

Blending Concerns

1. Additional profiles to control cross sections may be difficult to position accurately if the trajectory curve varies rapidly.
2. Due to their approximate nature, blends can be created, sectioned where refinement is desired, new improved profiles constructed at the cross sections, and a new blend created.
3. As additional cross sections are added, check for surface continuity between profiles, as waviness can be introduced.
4. The number of segments per profile is required to be equal, as each segment is blended in consecutive profiles. Arcs and lines of profiles may need to be subdivided.
5. Blends are sometimes restricted to profiles on parallel planes.

Sweep Concerns

The sweep volume must not self-intersect along the sweep trajectory as it would if a figure-eight trajectory were used. Arc or spline segments of the trajectory that are small compared to the profile can cause self-intersection.

Extrusion Concerns

Two or more nonintersecting outside planar profiles on the same plane can be extruded simultaneously to create multiple unconnected parallel solid primitives. Unconnected objects can define a valid solid, but applications such as automatic finite element meshing and automatic NC toolpath generation cannot use a solid of multiple volumes.

Profiles, Chamfers, and Fillets

When sweeping cross-section geometry into a solid, keep that geometry as simple as possible. Do not include fillets, radii, and chamfers (unless you have an incomplete or nontangent one). In some systems you are limited to the number of entities that you are allowed to sweep. Also, make sure that your geometry is planar before it is swept. Some nonplanar geometry can be swept, but may result in an invalid or unstable solid.

Plan Your Edge Sequence

Before attempting to apply fillets or radii to a solid, spend some time studying the model and plan the sequences of edges to which you want to apply them. Think about the way the model would be manufactured. Use the "undo" command (if your system has this feature) when the result is not as expected.

Copy Profiles First

Before you use a profile in a sweep operation, copy the profile in place to another layer, and use the copied profile for the sweep. Some SM systems do this automatically.

Modifying Profiles

One thing to look for when evaluating SM systems is which techniques are available for modifying profiles. Some systems allow you to change profiles by changing dimensions. Some require that you extract the defining curve, modify it, then replace the original curve. Or, you can just replace the curve with a new one. Other systems won't let you modify the profile. You must delete the part and start over, or modify it with boolean operations.

Boolean Operations

A solid boolean operation is the mathematical joining by UNION, SUBTRACTION, or INTERSECTION of two or more solids into a single new solid object (Fig. 7.2). Any valid solid can be used in boolean operations, including primitives, geometric operations, or the results of previous boolean operations.

The resulting solid will be either (1) the combined volumes of the prior solids, (2) the net volume of subtracting one solid from another, or (3) the volume common to both prior solids. Boolean operations give SM highly productive capability by modifying large amounts of geometry (each solid) in a single command operation.

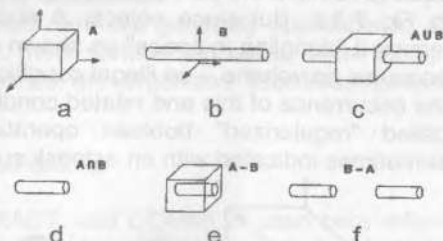


Fig. 7.2. Boolean operations: a – solid block; b – solid cylinder; c – results of UNION operation of A plus B ; d – INTERSECTION; e – SUBTRACTION $A - B$; f – SUBTRACTION $B - A$

Combining Solids

The boolean combine operation is typically named ADD, COMBINE, JOIN, MERGE, or UNION. The new solid is equal to the combined volumes of each selected solid. Typically, two or more solids can be selected for a single join command. However, the SM system will join each solid separately and display the final result.

Subtracting Solids

The boolean subtract operation is typically named CUT, DIFFERENCE, REMOVE, or SUBTRACT. Only two solids are typically selected for a subtract operation. The base solid is usually selected first, followed by the subtracted solid. The resulting solid is defined as the volume of the first less the volume of the second.

Intersecting Solids

The boolean intersect operation is typically named COMMON, CONJOIN, or INTERSECT. Two or more solids can be selected for the intersect operation, and the order of selection is allowed to vary. If more than two solids are selected, the underlying database will perform each intersect sequentially and display on the end result. The resulting solid is defined as the volume common to the original solids.

Boolean Groups and Errors

Some SM systems allow the selection of a group of solids that require the same boolean operation. Remember that each operation is performed sequentially and only the accumulative result is displayed. If any of the operations should fail, the erroneous operation should be undone and the model saved in this uncorrupted state. One additional boolean may require being undone to remove the corrupted operation because, frequently, the error is not known to the SM system or the designer until one boolean operation after the actual error has occurred.

Regularized Boolean Operations

As illustrated in Fig. 7.3, certain difficulties can arise when using boolean operations. An intersection operation between objects **A** and **B** is used to remove the part of object **A** to the left. The desired result is shown in Fig. 7.3,b. But since objects **A** and **B** share a face, the intersection leaves it "dangling in space" as shown in Fig. 7.3,c. This part of the object occupies no volume – an illegal condition in many SM systems. To prevent the occurrence of this and related conditions, specially formulated versions called "regularized" boolean operations are often used. They are sometimes indicated with an asterisk superscript (*).

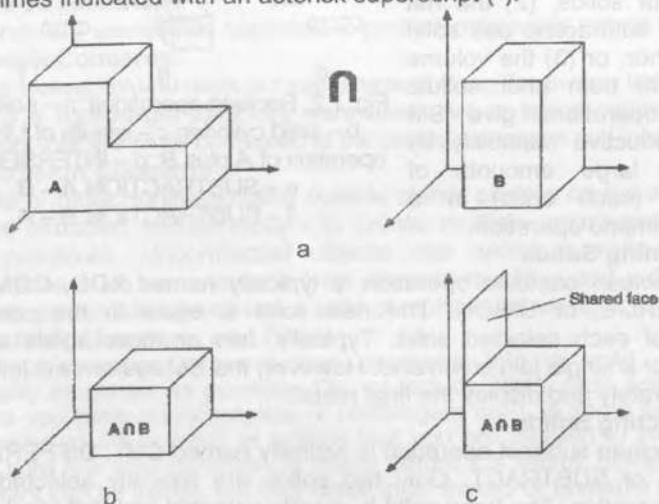


Fig. 7.3. Regularized boolean operations: a – removal operation by the intersection of **A** and **B**; b – desired result; c – possible result

Using Booleans on Coincident Faces

A boolean operation on objects having coincident faces may fail in some cases, such as when faces adjacent to the coincident faces are tangent. Solution: Use any of the following alternate construction techniques instead of a boolean operation.

1. Always creates a single profile to be swept along the same drive curve (trajectory).

2. Rather than sweeping a profile, then sweeping a face of the result, and then joining the two objects, sweep the profile once. Use local operations to move faces when possible.

3. Guidelines: You should also keep the following guidelines in mind to avoid coincident faces, which may cause a boolean to fail:

a. Fillet a resulting solid after a boolean, so a large number of coincident or tangent faces do not result. Exception: In a few cases (such as a vertex with four edges), if attempts to fillet the resulting solid fail, try filleting the objects first.

b. When splitting or trimming solids, extend the trimming surfaces beyond the faces of the solid being modified so that the "knife" protrudes outside the solid.

c. Position boolean operands that make through-holes so that they extend beyond the faces being punched, rather than making them flush.

Modeling Guidelines

The modeling guidelines discussed here are generally applicable to all CSG and B-Rep SM systems and most surface modelers, even though implementations vary. These guidelines are organized into three groups addressing:

1. Minimum modeling time
2. Model size and complexity classifications
3. Clean modeling

The terminology UNION, SUBTRACT, and COMMON used here reflect CSG boolean terminology. However, the information is also applicable to B-Rep modeling. Merely substitute ADD, CUT, and TRIM as required.

Minimum Modeling Time

The ideal CSG tree will have several levels of constructions as shown by the model classes in Fig. 7.4. The class 1 CSG tree should never be used for complex models. The ideal B-Rep model is also built using constructions. CSG and B-Rep modeling and editing time will both be greatly reduced with the use of multilevel constructions. Because the model is built in levels, deleting the boolean associations of constructions to the model, or of individual shapes to each construction, will delete only a few specific boolean or surface intersection associations.

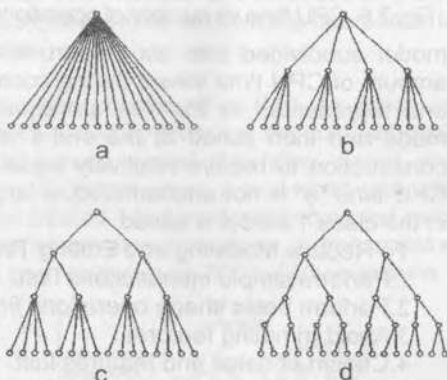


Fig. 7.4. Classes 1 through 4 CSG trees: a – class 1 models have a single level of constructions; b – class 2 models have two levels of constructions; c – class 3 models have three levels of constructions; d – class 4 models have four levels of constructions

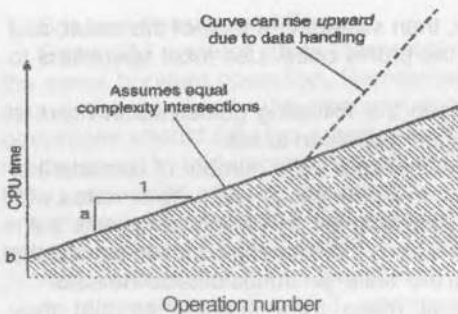


Fig. 7.5. CPU time vs number of operations

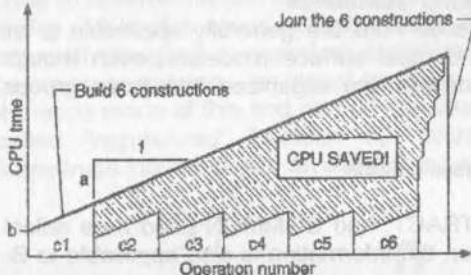


Fig. 7.6. CPU time vs number of operations

Fig. 7.5 illustrates the amount of modeling time required versus the number of operations in a class 1 model. In a class 1 model, the time necessary to perform a boolean or surface intersection operation has a small overhead time "b". If the operation fails immediately, the CPU time "b" is still expended. Each subsequent operation of the same complexity takes a little longer ("a") because more model surfaces must be checked for possible intersection with the current shape. If all SM operations are to a single root object, then the total CPU time is equal to the area under the curve shown. Note that the curve can rise dramatically in later operations due to data handling.

Fig. 7.6 shows the same model subdivided into six constructions of a class 2 model. Note the amount of CPU time saved during construction. In a class 2 model like the one represented in Fig. 7.5, six equal subdivided constructions can be made and then joined at the end. The subdivision process causes each construction to require relatively equal CPU time. Since the accumulative CPU time "a" is not encountered, a large portion of the CPU time required in the class 1 model is saved.

To Reduce Modeling and Editing Time:

1. Perform simple intersections first.
2. Perform basic shape operations first.
3. Avoid trimming features.
4. Construct detail and features last.
5. Add features to subdivision constructions.
6. Minimize the extent of intersections.

Subdivision Saves Time

As illustrated in Fig. 7.5 and 7.6, a complex solid model can often be subdivided into logical regions called constructions. Each construction can be individually modeled and then combined to achieve the final solid. Dividing the model into logical regions can save 50 percent of modeling time and 80 percent of editing time. Please note, however, that smooth, continuous surfaces should not be broken in the subdivision process.

CPU Time versus Constructions and Operations

A solid consisting of 100 primitives and geometric operations can be built in many ways. For example, the model could have 100 constructions of one primitive each, 50 constructions of two primitives each, etc. The relative CPU time to build such a model is compared in Fig. 7.7. Ideal modeling is achieved when the total number of constructions equals the total number of operations per construction.

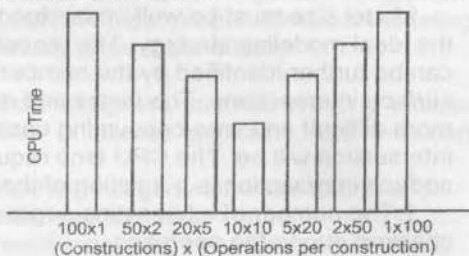


Fig. 7.7. CPU Time for a 100-operation solid

For a solid having 100 operations, using 10 constructions of 10 operations each yields the minimum required CPU time, 20 constructions of 5 operations each or 5 constructions of 20 operations each requires only slightly more time. At the extremes model creation requires over twice the minimum time. A poorly constructed larger model can consume over three times the minimum modeling time.

The CPU Time Curve

As shown in Fig. 7.8, the boolean (CSG) or intersection (B-Rep). CPU time curve is greater for a series of complex surface intersections than for a series of simple surface intersections. The number of intersections grows rapidly with each succeeding operation, causing all succeeding operations to perform even more intersection checks.

Simple Intersections First

Perform simple intersections first when possible (Fig. 7.9). As much as 80 percent of the total CPU time can be saved by following this rule, although 30 to 50 percent savings is probably more typical. Modeling speed and efficiency considerations become very important when a real-world part with actual manufacturing surfaces must be created. Most moderately complex parts require at least 50 shapes or surface intersection operations. A transmission housing or engine block, for example, can easily require 300 to 500 operations.

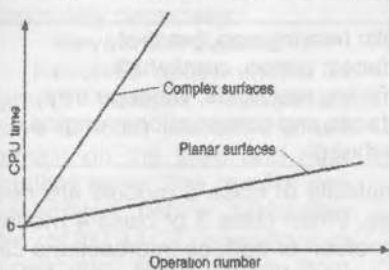


Fig. 7.8. CPU time vs operations and surface complexity

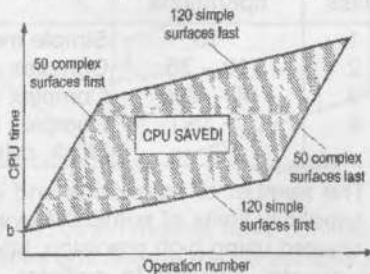


Fig. 7.9. Performing simple surface operations first

Model Size and Complexity

Model size must be well understood in order to properly determine the ideal modeling strategy. The model classifications shown in Fig. 7.4 can be further identified by the number and complexity of required surface intersections. The larger and more complex the final model, the more difficult and time-consuming each subsequent boolean or surface intersection will be. The CPU time required to perform a boolean or surface intersection is a function of the following:

1. The number of intersecting segments of all surfaces that must be checked during the operation
2. The number of new intersection segments produced in the current intersection
3. The order in which the intersection operations are performed
4. The accuracy of the intersections computed
5. Other factors such as the data transfer rate with the database, how I/O is performed, etc.

A complete intersection check requires more time to perform than a local intersection check. Some SM systems limit the number of surfaces checked for possible intersection with the current object by checking only at surfaces selected. An intersection may be missed during a local operation if complex surfaces are nearby. In some cases, the option of performing the complete intersection check when desired or required is available.

Model size can be approximated by measuring the number of boolean operations for primitive and geometric shapes plus features (CSG), or the number of surface intersection operations (B-Rep). In Table 7.1 model classifications are further identified by the total operations involved.

Table 7.1
Model Classifications by Number of Operations

Model class	Number of operations	Descriptive examples
1	15	Simple models; bearing cap, bracket
2	10 - 75	Complex surfaces; piston, crankshaft
3	50 - 200	Complex surfaces; manifolds, windage tray
4	> 150	Complex surfaces and constructions; engine block, cylinderhead

The number of operations and complexity of class 3 models are near the capability limits of some SM systems. When class 3 or class 4 models are created using high precision, later boolean or surface intersections can take several minutes to perform. Consequently, simplified models are sometimes created which may not be useful for all applications. An understanding of SM fundamentals can partially alleviate this problem.

As surface complexity increases, the time required to perform intersections can increase dramatically. Without proper direction, modeling time for a 300 operation solid can easily grow exponentially.

Clean Modeling

Clean modeling techniques allow designers to avoid creating excess geometry. If a model is not clean of excess geometry, the following conditions may occur:

1. Automatic finite element meshing will generate more nodes and elements, and the mesh will become more difficult to control.
2. NC toolpath generation will require stepping over extraneous edges and surfaces.
3. Graphic displays may be inconsistent.
4. Drawings are incorrect, ambiguous, and undimensionable.
5. Tolerances and surface finishes cannot be specified properly.

For Clean Modeling

A clean solid model is more useful than an unclean one to all downstream applications. Remember the following:

1. Always create continuous surfaces.
2. Always remove unwanted material.
3. Construct as produced.
4. Avoid offsetting surfaces.
5. Minimize narrow faces.
6. Avoid small corner angles.

Continuous Surfaces

Clean models have continuous surfaces wherever possible (Fig. 7.10). This requires that the total number of surfaces are kept to a minimum: surfaces have the fewest vertices, edges, and no gaps. Do not break a continuous surface unless absolutely necessary.

Revolution Seams

Revolved (axisymmetric or turned) shapes sometimes have a seam present that will remain on the solid and near another edge. The revolve can be rotated before performing the SUBTRACT operation to move the seam away from other edges or to eliminate it entirely.

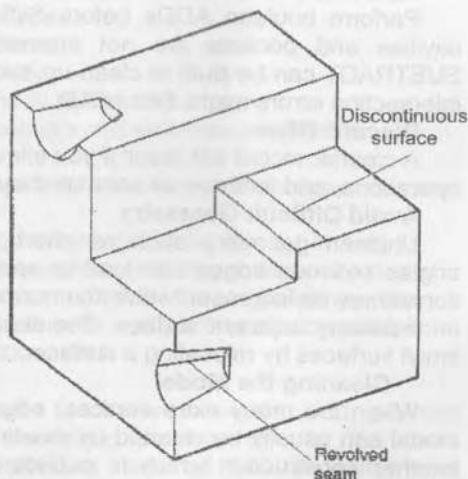


Fig. 7.10. Clean models have continuous surfaces and seams whose vertices are not within system tolerance proximity of each other

Edge Precedence

Perform intersections in an order to produce the fewest number of vertices, edges, and surfaces. The edges of shapes added usually take precedence over existing solid model edges. Depending on the SM system, existing edges may take precedence over the added edges, both sets of edges may remain, or all edges on continuous surfaces are automatically removed (this is the cleanest model).

Unnecessary edges will cause an automatic finite element mesh to possess many more nodes and elements, and NC toolpaths will encounter more surfaces, etc.

Rework Unclean Geometry

Always rework unclean geometry rather than covering it up or simply trimming it off. The solid will retain more operations due to creating unclean geometry and covering it up than it will if the geometry is remodeled properly. The additional operations can contribute to future errors and cause later operations to consume more time than is necessary.

Construct As Produced

Solids should be constructed similar to the way they will be machined. Build a SUBTRACTION construction in the shape of pockets to produce cleaner pockets, rather than subtracting a series of solids. Build a UNION construction to produce a cleaner appendage, rather than UNIONING a series of solids. Fillets should be created last as fillet features.

ADD Before SUBTRACT

Perform boolean ADDs before SUBTRACTS when possible so that cavities and pockets are not erroneously filled in. Another identical SUBTRACT can be built to clean up such an error, but coincident surface intersection errors might then result.

Discard Often

A cleaner model will result if you eliminate primitives, geometric operations, and features as soon as they are no longer needed.

Avoid Difficult Geometry

Unclean geometry such as short edges, narrow faces, and small angles between edges can lead to application difficulties. Fillet radii can sometimes be increased within the normal design tolerance to eliminate an unnecessary adjacent surface. The design might be adjusted to eliminate small surfaces by relocating a surface.

Cleaning the Model

When too many extra vertices, edges, and surfaces are created, the model can usually be cleaned by creating and subtracting from the model another construction which is outside the solid and coincident with its surfaces. In some cases, the extra edges and surfaces may be removed from the model, but the vertices may remain. Removal of a finite amount of material may be required to remove the vertices. Alternatively, a construction can sometimes be built internal to the model and coincident to its surface, and a UNION performed.

Exploding Solids

Some SM systems allow the solid to be exploded into individual surfaces without loss of definition. Surface areas can then be cleaned up, rejoined, and made into a solid once again. Geometry required for function must not be eliminated, of course.

Select Actual Edges

When selecting a solid edge for an operation, make the selection on an actual edge, not a meshed line. **Rationale:** Future operations may change a mesh pattern. If so, selecting a meshed line will cause regeneration to fail. **Usage tip:** You can distinguish if something is an actual edge by using selection filters.

Navigating Errors

Modeling Errors

Modeling errors are almost always due to one of the following:

1. Exceeding array sizes in the computer code. A particularly complex operation may require storing more entities such as edges than the software was coded to handle.
2. Exceeding the memory size of the computer's CPU. A complex operation may require more memory than is available.
3. Imprecise geometry.

To Reduce Modeling Errors

1. Avoid trimming features.
2. Avoid trimming surfaces.
3. Avoid latent inappropriate surfaces.
4. Minimize concurrent intersections.
5. Create final intersections directly.
6. Perform boolean ADDs before SUBTRACTS.
7. Minimize concurrent operations to avoid software array limits.
8. Eliminate extra operations.
9. Newer edges have precedence over coincident older edges.

Avoid Trimming Features

If at all possible, avoid trimming features (such as through-holes) by basic-shape boolean or surface intersections. All intersections at exterior faces will have been needlessly performed twice. If a single intersection fails to be determined, the entire boolean or surface intersection will fail. Always perform basic shape operations first.

Avoid Trimming Surfaces

Trimming surfaces can lead to more boolean and surface intersection errors later in the modeling process due to the extra geometry and the extra operations not required with an untrimmed surface. It is always best to build a surface to its trimmed location. However, the definition of a complex surface at the desired boundaries are not always known; hence, the trimmed surface is used. Trimmed surfaces can also be a cause of lost geometry when translating to neutral formats.

Avoid Latent Gratuitous Surfaces

Latent or gratuitous surfaces are those that are no longer visible after the boolean or intersection operation because they lie inside or outside the solid. They can cause problems for subsequent operations. After a SUBTRACTION has been performed, the box ADDITION operation may fail due to its coincidence with the now-latent surfaces of the box SUBTRACT, especially if complex surfaces are present (Fig 7.11). This error can often be corrected with a shorter box SUBTRACT or a longer box COMMON where the coincidence does not occur. Try to avoid complete coincidence with latent surfaces.

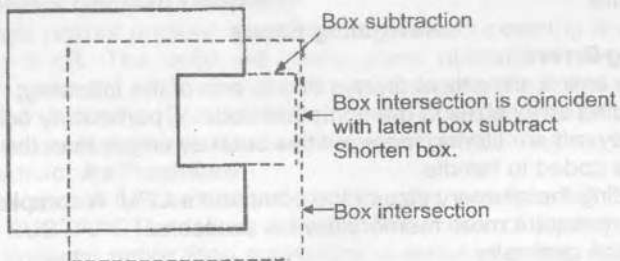


Fig. 7.11. Latent gratuitous surfaces

Minimize Concurrent Intersections

The hub should be intersected with just one spoke, and the resulting solid intersected to the rim. The remaining spokes should then be intersected to the model, one at a time. Alternatively, but to be avoided, all of the spokes can be intersected to the hub first. However, intersecting the rim will then entail numerous simultaneous intersections of the rim to all the spokes. The intersections look as though they are all identical; but due to orientation, positioning, and accuracy errors, they differ. If just one intersection segment of a spoke to the rim fails, the entire intersection operation fails.

Large versus Small Geometry

Accuracy-related errors can occur when very large and very small geometries exist within close proximity.

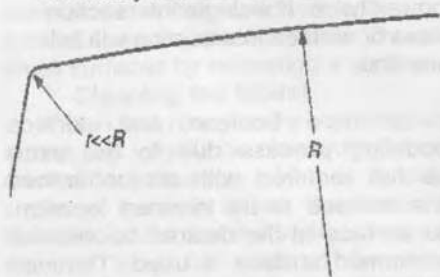


Fig. 7.12. Large vs. small geometry

The coexistence of very large and very small geometry can produce accuracy errors. The accuracy required by the small radius r is clouded by the imprecision generated from the large radius R when the two are required to be tangent. Perhaps radius r can be left sharp or increased without undue impact to the design in Fig. 7.12

the existence of the small radius is clouded by the precision lost with the occurrence of the very large radius. The precise locations of the intersections at the radii clearly require that the geometry possess more than eight digits of accuracy. Consequently, the intersection may fail.

Minimize Size and Complexity of Intersections

To reduce memory and CPU requirements, always minimize the size and complexity of intersections. Software array limits can overflow. As illustrated in Fig. 7.13, the order in which boolean operations are performed can help minimize the size and complexity of intersections. In this simplified example, part **B** should be intersected with part **A** prior to the intersection of parts **C** to reduce the risk of possible intersection errors.

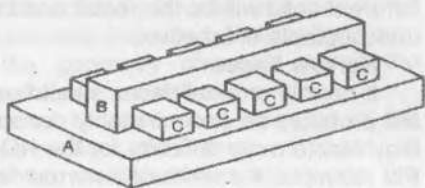


Fig. 7.13. Minimize the size and complexity of intersections

Accuracy Settings

In Fig. 7.14, the search for the intersection of the large and small geometries will not be successful if the search path is too coarse. Coarseness greatly speeds up the search process, but at the cost of accuracy. If the SM system allows the temporary tightening of intersection tolerances, extreme caution should be exercised in doing so.

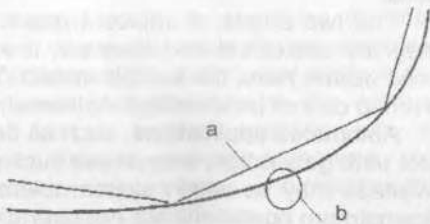


Fig. 7.14. Accuracy settings:
a – polygonalized search line;
b – small feature

The coarseness of the system's accuracy settings can cause small features to appear not to intersect when they actually do. In this example, the coarse polygonalized intersection search line along the curve misses an intersecting feature. Do not modify the system's accuracy settings unless absolutely necessary. Small feature doesn't intersect search line.

Caution: Reset the accuracy to the default value for later intersections. Also, changing accuracy from one intersection to another may cause errors in later intersections.

Intersection Errors

Intersection errors generally result from reduced accuracy, which causes geometry to be very slightly noncoincident or non-tangent when it should be. The geometry types for coincidence and tangency errors are vertex-vertex, vertex-edge, vertex-surface, edge-edge, edge-surface, and surface-surface.

Surface-Surface Tangency

Errors can often be overcome by extending and embedding one surface inside another solid or surface. Transition and torus surface tangency errors are the most prevalent, due to the variable radius and double curvature conditions present in these geometries. In some cases a different solid will be the result and the design may change, making this an unacceptable alternative.

Narrow Faces

If narrow or small faces should exist, a subsequent finite element mesh will probably be uncontrollably dense at this location. The narrow surfaces may create more difficulty for the NC programmer, and/or the NC software. For example, the width of a narrow face may be much smaller than the end of the tool bit, causing the NC software to interpret this condition as an error.

Twin Edges

The topology of a polyhedron (solid body) can be defined as: (1) both ends of every edge is a vertex, (2) edges bound an infinite surface into a face, and (3) exactly two faces (polygons) meeting along each edge of the solid.

The two edges of adjacent faces are known as *twin edges* because they are coincident and identical. If a gap or overlap in the faces of the solid occurs here, the solid is invalid. The twin of each edge at the gap or overlap cannot be identified mathematically at this location.

Automated applications, such as finite element mesh generation or NC tool path generation, cannot use such geometry. Note that some gaps and overlaps may be within system tolerance levels. Also check to see that downstream operations are not operating at a tighter tolerance level.

Vertex-Vertex Coincidence

Two vertices must coincide precisely (to about 12 significant figures) or must be at clearly different locations (differing by at least one digit in the fifth significant figure) or a boolean or surface intersection error becomes more probable.

Intersection Error Debugging

If it is noted which and when error messages occur, and what the source of the error is, then debugging becomes possible. A flowchart can then be constructed where the message suggests the possible causes, and each path suggests possible situations that will cause this message and solutions to overcome the error. This practice applies to all system and operational error messages as well.

8. EDITING STRATEGIES

Editing solids is very important because it frequently occurs late in the product development cycle when changes can drastically impact product cost, quality, schedule, and profitability. Editing which alters only geometry may have limited impact, while that which alters topology (the total number of vertices, edges, and faces) usually results in an impact of much greater consequence.

Occurring Types of Change

During the course of product development, three types of changes will usually occur before the solid model is released for manufacturing. These include the areas of (1) geometry, (2) topology, and (3) cosmetics.

Geometry Changes

If editing alters only geometry, then associativity is valid, and associated applications of the solid will not be measurably impacted. Notifications sent to the affected applications about the geometry changes permit the applications to generate new or additional views, update notes, etc.

Editing that typically affects geometry includes only changes in the surface intersection accuracy value, units, and minor changes in dimensions, angles, dimensioning scheme, and tolerances which alter the number of decimal places of dimensions and angles. The topology is unchanged, as noted by no changes in the number of vertices or edges of each face, and no change in the number of faces (see Fig. 8.1). Geometry changes allow vertices to move and both edges and faces to be stretched and or truncated.

Topology Changes

Topology changes can invalidate associativity by modifying the total number of vertices, edges, and faces of a solid, often forcing complete rework and delays in associated applications (Fig. 8.1). Labor costs in the associated applications occur with each change. Later in the product development cycle, costs may also include product retesting, new or modified manufacturing processes and fixtures, packaging changes, etc. adds or deletes vertices, modifies, adds, or deletes faces, and modifies the total number and relationship of vertices, edges, and faces.

Cosmetic Changes

Some design information is only cosmetic to the solid model and is not evaluated into the SM geometry or topology. This information usually conveys design intent to other applications.

Cosmetic information never changes geometry or topology and includes changes in dimension locations, material specifications, notes, and attributes. SM attributes are entities such as tolerance, surface finish, surface plating and coating, nonevaluated threads, flatness, perpendicularity, angularity, concentricity, parallelism, straightness, runout, and circularity.

Editing solids

Solids can be edited directly for all operations that modify only geometry as well as those operations that modify topology without adversely violating surfaces. There is an exception, however, in SM systems that prevent

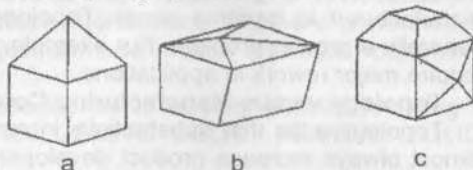


Fig. 8.1. Editing geometry and topology:
a – original solid; b – editing geometry;
c – editing topology

modification of an evaluated root solid. Direct editing of solids can include:

1. Geometry
2. Topology
3. Dimensions
4. Surface intersection accuracy

Editing Geometry

Dimensional and angular changes frequently alter the size and location of surfaces without altering topology. This editing can be performed directly on the individual shapes without editing the CSG tree, as discussed later in this chapter. The primary solid or local details can be stretched in one or more directions, details repositioned, and parameters changed. Editing functions are typically accessed via a MODIFY menu. Editing choices under MODIFY may be for each primitive or geometric operation type or for each editing operation type such as STRETCH, BEND, MOVE, ROTATE, ENLARGE, WARP, TWEAK, or CHANGE.

Editing Topology

Topology editing changes the number of vertices or edges of some faces, or the number of faces (see Fig. 7.1c). Editing that adds, removes, or modifies topology without causing unwanted surface violations can be performed directly on a solid.

Adding and Removing Topology

Topology edits that remove material are generally of minor impact. These edits can include adding holes, chamfers, and minor details such as pockets and slots. Manufacturing can sometimes add a tool and toolpath to machine these changes, and additional analysis might not be required. However, removing holes, chamfers, etc., may cause parts already manufactured to become scrap. Topology edits which add material are generally a greater problem. For example, bosses, fillets, and draft usually require major rework in applications.

Topology versus Manufacturing Costs

Topology edits that substantially increase the number of surfaces will almost always increase product development costs and production costs. Manufacturing may require more tool changes, setups, and toolpaths. These changes may be justified if they increase product quality. Conversely, topology changes that substantially reduce the number of surfaces and simplify the product can decrease product development and production costs.

Editing Dimensions

Dimensions are typically edited by changing the dimension values, relations, scheme, or changing tolerances. The product development team will have used varying techniques to determine the ideal parameters for optimal product quality, manufacturing time, and cost characteristics. By this process, critical dimensions are identified, critical tolerances are tightened, and noncritical tolerances are loosened. These techniques require time to perform and, consequently, may require editing the solid model.

The dimensioning scheme and related tolerances will cause a particular tolerance, stack-up. They convey to manufacturing which dimensions and tolerances are most important for achieving product quality, and which are not as important. In some SM systems changing a tolerance will change the number of decimal places in the dimension to which the tolerance is applied.

Editing Surface Intersection Accuracy

Changing the surface intersection accuracy will affect all subsequent intersections. A full model regeneration will of course recompute all intersections at the new accuracy. The accuracy is sometimes tightened to refine geometric and topological data for manufacturing.

Editing the CSG Tree

The CSG tree must be edited when:

1. Unwanted surface violations occur during direct SM editing.
2. The modeling scheme must be changed due to major redesign.
3. The logical shape of constructions has changed or a different number of constructions must be used.
4. A shape type must be changed. For example, a revolve geometric operation must be replaced by a cone primitive, a blend geometric operation, or a construction.

If your SM system is associative, the CSG tree is edited by breaking the associations of the shapes to be edited from the tree, editing the necessary shapes, and then reconstructing the CSG tree associations. The associations broken may be of the root shape to individual primitives, solid geometric operations, or constructions composed of primitives and geometric operations. Only remove as many associations as necessary to limit the amount of rework. Surface intersections that were performed to create the B-Rep boundary object are also broken and again performed upon reconstruction.

Breaking the CSG tree associations is usually performed using a function called MODIFY SOLID, DELETE SOLID, REMOVE SOLID, SMASH SOLID, etc.

Edited shapes may be individual primitives and geometric operations. Hence, a construction removed from the solid for editing must in turn have the associations and intersections of the shapes to be edited removed from its CSG tree.

Editing Order of Shapes

It may be necessary to edit the order of constructions in a solid model, or shapes in a construction because:

1. A shape fills or removes a portion of another shape.
2. Computational and labor time can be saved during subsequent reevaluations by shifting simpler shapes to earlier positions.
3. The design is better understood, allowing intuitive ordering.
4. Modeling operations can be grouped better.

Adding or Removing Shapes

Shapes can be added or removed from the solid model as required for design changes. Shapes added will be placed at the highest level in the CSG tree. For example, a shape added to the root solid will be placed at the highest level of the entire CSG tree; a shape added to a construction (node) will be placed at the highest level of the CSG tree of that construction; and a shape added to a leaf node (basic shape) may be added at a lower level.

To Reduce CPU Editing Time

A solid model that was built using class 2,3, or 4 CSG constructions can be edited in an order-of-magnitude less time than if a class 1 CSG construction was used. Refer to Fig. 8.2 for examples of each CSG tree class. The number of associations and boundary intersections that must be recomputed during this process is substantially reduced by using the proper class of CSG tree. Note that over 90 percent of the CPU time required for editing is saved. This is illustrated by the shaded area

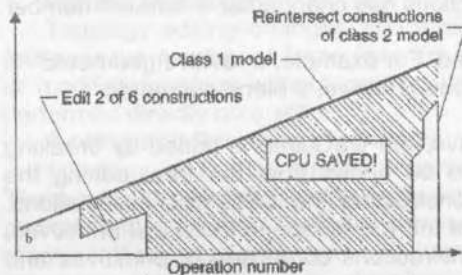


Fig. 8.2. Editing the solid model of a class 2 vs. a class 1 CSG Tree

must be redimensioned prior to removal to avoid corruption. A warning message is usually issued if position associativity exists. If your SM system is associative, a general awareness during all editing process is highly recommended.

Eliminate Unused Operations

Solids created with boolean operations should be edited or modified to eliminate unused operations and "scabbing-on" other operations. For example, don't increase a hole diameter by subtracting a larger cylinder from a solid without deleting the original subtraction operation.

Editing Solid Primitives

Solid primitives are edited by changing their parameter values. For example, a cone may be edited by changing its base radius, height, base point, axis, and coordinate system of definition.

Each primitive usually has its own set of parameters accessed via a function called EDIT PRIMITIVE or MODIFY PRIMITIVE.

Editing Solid Geometric Operations

Solid geometric operations such as Extrude, Sweep, Blend or

Transition, and Revolve can be edited by replacing, adding, or removing a cross-section profile curve. Substantial changes in surface definition can occur.

Editing Blends

A blend can be edited by reorienting a cross section, changing a cross-section shape, replacing one cross section with another, adding or removing a cross section to the set of cross sections, or by modifying the trajectory curve(s).

Editing Extrusions

A solid extrusion can be edited by reorienting the cross section, changing the cross-section shape, replacing the cross section with another, or changing the linear path length and/or direction.

Editing Revolutions

A solid revolution can be edited by reorienting the cross section, changing the cross-section shape, replacing the cross section with another, or changing the revolution angle and direction of revolution.

Editing Sweeps

A solid sweep can be edited by reorienting the cross section, changing the cross-section shape, replacing the cross section with another, or by replacing or modifying the trajectory curve;

Affecting Other Shapes

To edit an Extrude, Sweep, Blend, or Revolve, redisplay the original cross-section profile curves. Determine if the cross section or any segment(s) of it are used to define any other shapes. For example, a cross section may be used to extrude in one direction and sweep in the opposite direction. If so, determine if editing or deleting the cross section will affect the other shape as well. The other shape may have its own copy of the cross section, which is not associated to the cross section being edited. Determine if others are to be modified also.

NC Considerations

Replacing an arc with a spline in a cross-section profile of an Extrude, Sweep, Blend, or Revolve will alter the surface definition produced. Toolpaths become more difficult to produce. Conversely, replacing a spline with an arc can have a desirable affect.

Blend Cross Sections

It is often necessary to change a particular cross section of the blend to attain a new cross-section shape, area, perimeter, etc. The number of segments in the cross section must not change unless all cross sections are modified to use the new value. Some SM systems allow a new cross section to be built and the current cross section then replaced with the new one.

Blend Segments

Cross sections are generally added or removed from the original set, or a cross section is reoriented in order to alter the surface between blend segments. The original surface may be too wavy between cross sections, or have inadequate clearance with surrounding components. Remember

that blends are approximate and are controlled only at their defining cross sections.

Parametric/Relational Solid Modeling System

Classical P/R SM systems are composed of five major components.

These are:

1. The sketcher
2. The SM system
3. The dimensional constraint engine (DCE) (also called *variational engine and parametric engine*)
4. The feature manager
5. The assembly manager

Editor's Note: Many of these concerns are eliminated in newer systems.

History Reflects Intent

Parametric modeling captures the operations you use to build a product. The captured operations, called *parametric history*, reflect your design intent (any geometric relationships you establish) and strictly adhere to that intent as sizes of the design change.

Importance: During the life cycle of a design, the product matures and design intent can change. As a result, you may have to redesign portions of the product. You can use many techniques to incorporate flexibility into the parametric history of a model, such as with equational constraints, so that it is easy to handle certain design-intent changes.

Order Is Important

History is time-dependent. **Importance:** Parametric modelers record operations in a chronological order and maintain any relationships you define with displayed parameters. Because operations are order-dependent, they are aware of a model's state only at the time you insert them. Operations do not know about any future operations you may enter. When regenerating a model, the system may not conduct an operation if either of the following occur:

1. A previous operation negates it. (**Example:** If you change the parameter of a solid so that it no longer intersects with any other entities, any boolean operations in which the solid is an operand will fail.)
2. It requires another operation before it, but you placed the needed operation after it.

Verify History Periodically

As you build a model and relationships get more complex, take the time to periodically review dependencies. **Purpose:** Using these options can show which entities are dependent on a selected command or operation so that you can verify history independence.

Example: If you create an entity that contains more than one operation of the same name, you may be able to display the history number of each operation. You can then be sure you are selecting the desired operation for something, such as the removal of a hole.

Modifying Events

You can typically make two kinds of parametric changes:

1. You can remove an operation from the parametric history of a model.
2. You can change the value of a parameter to alter such things as the size, shape, and location of an entity.

Depending on the SM system, you may not, however, be able to modify a parametric event. *Example:* You may not be able to create a general radius fillet having four radius values and later add two more radius values.

Possible work-around: If you know you may need to add radius values, create that fillet with extra but equal values. Later in the design process, you can change parameters as required or use constraints to build relationships between the values.

The Sketcher

The sketcher is one of three common ways in which geometry (profiles and features) are supplied to a P/R SM system. The other two ways are geometry which is created using standard CAD and SM construction techniques and geometry which is imported from external design systems.

Sketchers today are another form of entering planar (or, in some of the more sophisticated systems, 3D) preliminary designs. Since most P/R SM systems allow sketching with a grid, it is possible for a "sketch" to have the same dimensional accuracy that a figure constructed in any other part of the SM system would have.

Sketching typically provides for the entry of line, arc, and (in some cases) spline data. Most P/R SM systems allow for lines to be constrained under a designer's control. Thus, it is possible to constrain all lines to the closest 45.0° angle, or the closest 10.0°, or 1.0° angle, etc.

Also, most sketchers today incorporate implication engines which examine the geometry as it is sketched, and if it meets certain implications, assigns attributes to the geometry. For example, if two lines are approximately parallel, then as a first guess, the sketcher will supply the constraint "parallel" to these lines. If two arcs have approximately the same radius, then most sketchers will imply that the radii are meant to be equal.

A sketch is typically used to generate the profile for either an axial or a rotational sweep, for user-defined features which can be generated from an axial or rotational sweep, or for *hard-coded* features. *Hard-coded* or *canned* refers to features which are selected from a menu (choice list), icon, or other mechanism for identifying a predetermined geometric configuration, the methods for controlling its placement, and the types of variables it can contain. Sketches are ordinarily controlled by modals. These allow the designer to influence the implication engine.

Sketch types include, but are not limited to:

1. Chain (i.e., "heel and toe")
2. Centerline
3. Axis-symmetric

Examples of sketch modes are:

1. Snap angle
2. Sketch mode
 - a. Linear
 - b. Nonlinear
3. Fillet mode
 - a. Automatic between linear sketch curves
 - b. No fillets between linear sketch curves
4. Dynamic notch
5. Dynamic round
 - a. Semicircular
 - b. Free angle
6. Sketch closure
 - a. Direct to start point
 - b. Horizontal, then vertical to start point
 - c. Vertical, then horizontal to start point
 - d. Direct to start point with fillet between first and last curves

The Solid Modeling System

The solid modeler is the area in which planar profiles and 3D curves are turned into solid objects by axial or rotational sweeps, general surface enclosures, or creation of analytic solids (cubes, spheroids, cones, toroids, right-circular cylinders, etc.). While the objective of all P/R SM systems is to generate solid models, that fact is often obscured by the sketcher, relational edit, feature generation, and assembly capabilities. All of these components would be useless without a mechanism for creating the actual solid.

The most common ways of generating a solid are by either an axial or rotational sweep. However, robust P/R SM systems include a significant number of additional ways to produce solids. These include:

Each P/R SM system has special cases for producing a solid model. However, the axial and rotational sweep as well as boolean operations which are applied to the results of these sweeps produce a large percentage of the solids in any given system. (Refer to Chap. 6 for specific modeling strategies.)

The Dimensional Constraint Engine (DCE)

The DCE (also called *variational engine* and *parametric engine*) provides for the modification of dimensions (variables) associated with the model and the regeneration of the model after variables have been changed. The DCE is the P/R segment which provides for the following:

1. The establishment and modification of dimensions
2. The establishment and modification of topological relationships between geometric entities (a topological relationship is used here to mean the way geometry is arranged relative to other geometry);
3. The creation and modification of mathematical relationships in which one entity is constrained by arithmetic values associated with parameters of one or more other entities

The DCE is the heart of a P/R SM system and often contains most of the functionality of a "variational" engine. The DCE is a software component which manipulates geometric designs to satisfy given dimension and topological constraints. When geometry is changed, either directly or by changing an associated dimension, it is the function of the DCE to analyze the impact of that change on other geometry in the design and to modify the geometry corresponding to the constraints and relations which have been established for that design.

The DCE also determines whether a particular design is (1) underdimensioned, (2) dimensioned correctly, or (3) overdimensioned. In the event that a design is underdimensioned, a typical DCE flags the geometry which is not dimensioned adequately so that the designer can add the dimensions necessary to make a "correctly" dimensioned part. Regardless of the dimensional state of a design, a proper DC should operate for each of these situations.

DCEs are typically concerned with evaluating geometry in a 2D subspace of 3D. They typically allow points, lines, circles, conics, and splines as the representative geometry, along with geometric constraint such as concentric, parallel, perpendicular, tangent, and coincident. A significant amount of work is currently under way in several organizations to generate a DCE which will handle full three-dimensional geometry.

Five factors determine the usefulness of a DCE (a discussion of each follows):

1. Dimension types recognized
2. Degree of geometric constraints
3. Freedom to postpone dimensioning
4. External compatibility
5. Configuration of variables

Dimension Types Recognized

This refers to the number and type of dimensions supported by the DCE. Every major design and/o manufacturing organization adheres to a drafting standard. While specific standards vary throughout the world, there is usually one significant standard within a particular country. It is important that geographically dependent dimensioning standards be acceptable to the DCE. If the DCE forces a particular mode or type of dimensioning which is inconsistent with the standard used by the industry in which it is operational, that particular DCE is of academic and not industrial value.

If varying types of dimensions are accepted by the DCE, they should include, but not be limited to:

1. Angle
2. Baseline
3. Diameter
4. Horizontal
5. Ordinate

6. Parallel
7. Parallel with reference line
8. Radius
9. Thickness
10. Vertical

Degree of Geometric Constraints

This refers to the number and type of geometric constraints supported by the DCE. A robust should provide support for, but not be limited to:

1. Coincident arcs
2. Concentric arcs
3. Colinear endpoints
4. Colinear lines
5. Fixed
6. Horizontal
7. Vertical
8. Normal
9. Parallel
10. Points On a curve
11. Tangent

The Feature Manager

The feature manager provides for the addition of pockets, slots, holes, bosses, ribs, etc., and user-define features to any face or faces of a model. The functionality which contributes to the dramatic increase performance of a feature-based P/R SM system over a traditional CSG/B-Rep hybrid SM system is the intelligence which is imparted by features. A necessity in any feature-based P/R SM system is a user defined feature. This allows the geometric and topological characteristics of the feature to be total defined by the designer and then applied to existing solids.

Many P/R SM systems also provide a full selection of default (hard-coded) features including, but not limited to:

1. Constant offset pockets
2. Circular and rectangular arrays of holes
3. Free slots
 - a. Straight or two-point slots
 - b. Three-point T and L slots
 - c. Four-point U slots
 - d. Circular slots
 - e. Slots parallel to a boundary
4. Necks or protrusions
5. Flanges

Traditional SM systems that are not feature-based or P/R force designers to use CSG boolean operators in order to create complex objects from simple components. P/R SM systems use not only CSG boolean operators, but all of the geometry associated with embedded

features as well. Thus a hole, a slot, a groove, and a pocket are recognized for what they are and do not require the creation of additional geometry which is subtracted (using boolean operations) from the base geometry. While some of the earlier P/R systems have the boolean operations embedded in the feature codes, most modern (post-1991) P/R systems use entirely different mathematics, leveraging off of the knowledge inherent in features, to generate the required geometry. This "feature-driven" philosophy both speeds the design process and simplifies the underlying database.

Advantages of Feature-Based Modeling

Designing with features provides a number of advantages. You can:

1. Insert intelligent parametric geometry into a model without having to create intermediate construction geometry.
2. Associate a variety of information to portions of a model.
3. Create features with attributes unique to your specific design requirements. Additional advantages:

4. Conventional CSG boolean operations which are time-consuming are not required.

5. Embedded feature characteristics are conveyed more intelligently to manufacturing applications such as NC. Holes are holes, pockets are pockets, etc., instead of simply raw surface geometry.

Relocating Features

Although highly productive, the P/R intelligence of feature definitions and their subsequent geometric locations places additional burden on the designer in the form of forethought and planning. Options for relocating features are limited to those that would result in equal topology.

Example: Relocating a boss feature on a flat surface close to an edge may not be possible. If the boss has a base radius which intersects the edge radius, the existing topology is violated.

The Assembly Manager

The assembly manager provides for the creation of subassemblies from piece-parts and the creation of assemblies from piece-parts and/or subassemblies. The assembly manager also controls the regeneration of assemblies and subassemblies when changes have been made to individual piece-parts. In addition, it controls the modification of the way in which assemblies are put together. Last, assembly management provides for the explosion of assemblies into their component piece-parts and subassemblies.

During the design phase, adequate planning often identifies those components which can or should be used as subassemblies based on design function or the manufacturing processes required.

There are two distinct ways of placing parts when creating a subassembly or assembly: (1) relative placement and (2) boolean placement. Each is discussed here, along with gathering operations for placing and relating parts in an assembly or subassembly.)

Relative Placement

In this method parts are placed relative to each another. This placement can be thought of as "loosely gluing" the parts to form a unit. The idea is that at some point in the future it would be desirable to explode or unrelate the assembly back into individual parts.

Boolean Placement

This method utilizes boolean operations. The result is the formation of a new single part, typically referred to as a *constructive solid geometry* (CSG) solid combined from two or more parts. The CSG solid usually cannot be exploded or unrelated easily.

Gathering Operations

Subassemblies and assemblies are created by the use of "gathering" operations. These operation define and retain relationships among parts. Gathering operations include, but are not limited to:

1. Align axes
 - a. Match centers
 - b. End to end
 - c. Offset from end
 - d. Normal to face-centered on face
 - e. Normal to face-centered on normal point
 - f. Normal to face-centered on face-offset
 - g. Normal to face-centered on normal point-offset
2. Align corners
 - a. Both flush
 - b. Offset first
 - c. Offset second
 - d. Offset both
3. Align definition coordinate system (DCS)
4. Align edges
 - a. Flush
 - b. Offset
5. Align faces
 - a. Align faces
 - b. Center aligned faces
 - c. Offset face from a face
6. Align features
 - a. Centered
 - b. Align an edge
7. Logical union (addition, $A + B$)
8. Logical difference (subtraction, $A - B$)
9. Logical intersection (both A and B)

Dimension-Driven Editing

In a P/R SM system, dimensions are typically assigned a symbolic variable name at one of three times: (1) when they are created, (2) when they are automatically scanned by the system, or (3) when they are chosen for use with a profile. In most systems the variable name can be

generated by the system or supplied by the designer, and it is usually displayed below or near the dimension value.

There are three classes of edits that can be performed on dimensions or variables. These are (1) value, (2) reference, and (3) appearance editing. Each is discussed in the following sections.

Value Editing

The first and most obvious way to change the value of a dimension is to select it and enter a new value. A second method is to select the name of a part or feature, or to select one of its profile curves. When this is done, many systems display a list of the pertinent variables and dimensions for that item, allowing the designer to enter one or more new values.

A third method for editing a dimension value is to key in the symbolic variable name followed by an expression that establishes the relationship between this dimension and other dimensions or variables. If a dimension is assigned an expression, the expression is usually displayed in place of the dimension name.

Reference Editing

Reference editing refers to the ability to toggle the reference status of a dimension. This gives the designer an option of displaying a dimension on-screen without having it affect the operation of the dimensional constraint engine (DCE).

Normally, a dimension is active and considered in all relevant relational operations. But if the dimension status is switched to reference, it is displayed but otherwise ignored. This can be useful, for example, if a design is overdimensioned. Changing one or more dimensions from active to reference allows the designer to eliminate the overdimensioning while still being able to view the information in those dimensions.

Appearance Editing

The first and most obvious method of appearance editing is changing a dimension's location. This is sometimes necessary after a dimensional change to a model. P/R SM systems do position dimensions automatically, but the ability to adjust the position manually allows the designer to improve both the legibility and the aesthetics of the annotated design. There are other methods of appearance editing which include:

Shared Dimensions

This method allows a dimension to be shared by more than one location in the design. This reduces clutter on the display while constraining the geometry just as if the extra dimensions were there.

Bidirectional Associativity

Associative dimensioning implies that a dimension is automatically updated when the geometry it is dimensioning changes. Dimension-driven implies the ability to change a dimension and have all affected geometry change. Bidirectional associativity implies that both update methods are available.

Associativity

Clarifying Associative Dimensioning

With associative dimensioning, a dimension is associated with geometry such that if the geometry is changed, the dimension is updated automatically. An example would be a dimension that changed to reflect that a line had been stretched. This capability does not change the length of the line if the dimension is changed. A P/R SM system adds this last step, where changes to dimensions or parameters drive geometry changes.

Tracking Associativity

While associativity can be a very powerful tool when used properly, it could be disastrous if not implemented properly. A master model that is used in multiple assemblies cannot be changed without consideration of each instance of its use. Each change to a model should be tracked. One cannot have an effective engineering system if changes can be propagated without each person who uses the data being aware that changes have been made. An integrated tracking system should allow those affected by a change to approve or deny it.

Бібліографічний список

Альберт Д. Теория сплайнов и ее приложения/ Д. Альберт. – М.: Наука, 1971. – 315 с.

Гилой В. Интерактивная машинная графика/ В. Гилой. – М.: Мир, 1981. – 380 с.

Иванов В. П. Трехмерная компьютерная графика/ В. П. Иванов, А. С. Батраков. – М.: Радио и связь, 1995. – 224 с.

Роджерс Д. Алгоритмические основы машинной графики/ Д. Роджерс. – М.: Мир, 1989. – 504 с.

Роджерс Д. Математические основы машинной графики/ Д. Роджерс, Д. Адамс. – М.: Машиностроение, 1980. – 240 с.

Уокер С. Интерактивная машинная графика/ С. Уокер, Д. П. Гурд, Е. А. Дроник. – М.: Машиностроение, 1980. – 171 с.

Фоли Дж. Основы интерактивной машинной графики: в 2 т./ Дж. Фоли, А. Вэн Дэм. – М.: Мир, 1985. – Т.1. – 367 с.

Bernus P., Mertins K., Schmidt G. Handbook on Architectures of Information Systems. – Springer-Verlag. – 1998. – 285 p.

Bernus P., Nemes L., Modelling and Methodologies for Enterprise Integration. – London: – Chapman and Hall. – 1996. – 312 p.

Bloor M. Product Data Exchange. London: – UCL Press Limited. – 1995. – 195 p.

Fisher A. S. CASE: Using Software Development Tools. – New York: J. Wiley & Sons Inc., 1988. – 289 p.

Hill D. R. C. Object-oriented analysis and simulation. – New York: Addison-Wesley Publishing Company, 1996. – 291 p.

Hoffmann, Christoph. Geometric and Solid Modeling: An Introduction. – Morgan Kaufmann, 1989. – 310 p.

LaCourse, Donald E. Handbook of Solid Modeling. – McGraw-Hill, Inc. – 1995. – 626 p.

Mantyla, Martti. An Introduction to Solid Modeling. Computer Science Press. – 1988. – 180 p.

Mortenson, Michael. Geometric Modeling. – John Wiley & Sons, 1985. – 269 p.

Pipes A. Drawing For 3-Dimensional Design. – Wm. C. Brown Publishers, 1992. – 176 p.

Samet H. The Design and Analysis of Spatial Data Structures. – Addison-Wesley, 1989. – 384 p.

Spur G., Mertins K., Joachem R. Integrated Enterprise Modelling. – Berlin: Beuth-Verlag, 1996. – 280 p.

Vernadat, F. B., Enterprise Modelling and Integration. – London: Chapman and Hall, 1996. – 420 p.

Борисевич Володимир Володимирович

SOLID MODELING

Редактор Т.А. Ястремська

Технічний редактор Л.О. Кузьменко

Зв. план, 2008

Підписано до друку 25.03.2008

Формат 60×84 1/16. Папір. офс. №2. Офс. друк

Ум. друк. арк. 4. Обл.-вид. арк. 4,5. Наклад 100 прим. Замовлення 163.

Ціна вільна

Національний аерокосмічний університет ім. М.С. Жуковського

«Харківський авіаційний інститут»

61070, Харків-70, вул. Чкалова, 17

<http://www.khai.edu>

Видавничий центр «ХАІ»

61070, Харків-70, вул. Чкалова, 17

izdat@khai.edu