

DESIGN MODELING

with

Pro/ENGINEER

5th Edition (Pro/E 2000i²)

8G247-005
Stabilizing Clip

Aluminum
14 Required
Scale 1:2.5
All dimensions in millimeters
All tolerances ± 0.02

7-4-97
JOSE MUNIZ

KARA TINA MWANGI

STABILIZING CLIP

01

James E. Bolluyt


Iowa State University

DETAIL I (2x)

Schroff Development Corporation

Shawnee-Mission, Kansas

SECTION A-A



Preface to the fifth edition (for Pro/E 2000i²)

In many if not most design fields, the possibility of creating a design as a complete and unambiguous three-dimensional computer model is having a profound impact on the entire design process — from analyzing and defining the problem to conceptualizing, analyzing, documenting, and producing the solution. This text is an introduction to the use of computers and software and **solid modeling** concepts to create three-dimensional computer models as an integral part of the design process. In particular, it is an introduction to the design software *Pro/ENGINEER*TM 2000i² (from *Parametric Technologies Corporation*, Waltham, MA), a computer-aided design package that uses a variation of solid modeling called **parametric modeling**.

One of the primary goals of traditional design graphics was to help students "visualize" in 3-D even though the medium was 2-D (pencil on paper). More often than not, however, there was no feedback in the form of a three-dimensional model or prototype with which to compare the 2-D views. Though instructors and peers might point out inconsistencies between views or unrealistic geometry, there were few chances to let students compare the views they created with an actual three-dimensional shape. Little formal design instruction consistently required working in 3-D — the tools were not available.

Today, however, modeling software packages like *Pro/ENGINEER* (*Pro/E* for short) provide the opportunity to design and analyze immediately in 3-D. In fact, in packages like *Pro/E*, there is no practical alternative to working in 3-D. Though one could draw a series of 2-D views in *Pro/E*, it would be tortuous to do so for anything but the simplest shapes and would negate most of the power of the software. Users have no real choice but to work, and therefore think, in 3-D. A failure to think and visualize correctly in 3-D very quickly leads to undesired results which tend to be evident immediately. At best, the model does not look like what had been envisioned; at worst, the software will not do what was requested and/or will produce an error message saying a request was not realistic.

This text was written with the intention of helping students become comfortable working in 3-D. It is intended to help readers understand the basic concepts of 3-D modeling. It is an attempt to illustrate the 3-D computer model database at the core of the design process and 2-D views (*drawings*) as a by-product of the modeling process (usually still necessary, but a by-product nonetheless). It

covers only the very basic modeling tools in *Pro/E* – enough to allow students to create the geometry of some realistic components and assemblies, but nowhere near a comprehensive exposure to all the capabilities of the software. Most chapters begin with a quick introduction to general modeling concepts and then illustrate how those concepts are implemented in *Pro/E*. End-of-chapter exercises are provided for practice in using the software and to further the understanding of particular modeling concepts

Features include:

- an emphasis on 3-D computer modeling characteristics and concepts including
 - modeling methods and corresponding model definitions
 - coordinate systems
 - view types and specifications
 - geometric analysis
 - parametric and feature-based modeling
 - assemblies
 - the 3-D to 2-D conversion for documentation
- a variety of step-by-step *Pro/E* examples to illustrate basic concepts and the tools for applying them
- end-of-chapter exercises on both concepts and software that vary from simple to challenging

ACKNOWLEDGMENTS

An ongoing project such as this is not possible without the help, patience, and understanding of many people. I extend thanks first to my family, especially to my wife Karen for her encouragement and patience. I also want to thank Stephen Schroff at Schroff Development Corporation for his encouragement and help and to the people at Parametric Technology Corporation for their support in this endeavor. Thanks to all those who have contributed to the development of the fantastic design tool of computer modeling. And lastly, thanks to those at Iowa State and elsewhere who are helping to lead us into the computer modeling age.

J.E.B.

Contents

Text Files ix

■	Chapter 1	Introduction to Computers in Design	1
	1.0	Computer-Aided Design 1	
		Modeling and the Design Process 2	
		The Electronic Model Database 3	
	1.1	Model Representation Schemes 3	
		Wireframe Representation 4	
		Surface Representation 4	
		Surface Types 5	
		Solid Representation 5	
	1.2	Solid Model Creation Techniques 7	
		Constructive Solid Geometry 7	
		Boolean Operations 8	
		Sweeping 9	
	1.3	Parametric Modeling 11	
		Chapter 1 Exercises 12	
■	Chapter 2	Working with a 3-D Computer Model	15
	2.0	Basics of the <i>ProENGINEER</i> User Interface 15	
		The Default Screen Layout 15	
		Command Entry 17	
		Aborting a Command 18	
	2.1	On-line help 18	
	2.2	Loading and Saving Model Files 19	
		Finding and Loading Model Files 19	
		Saving Model Files 19	
		Clearing the Screen (and Memory) 21	
		Successive Saves of the Same Model 21	
	2.3	Basic View Manipulation 22	
		Model Views and Coordinate Systems 22	
		Image Types and Associated Controls 22	
		View Types and Associated Controls 24	
		Naming and Saving Particular Views 29	

- 2.4 Multiple Windows 30
- 2.5 Model Inquiry 32
 - Lengths and Distances 34
 - Angles 35
 - The Query Select Tool 36
 - Coordinate System Distances 38
 - Areas 38
- 2.6 Obtaining Hardcopies 39
 - Hardcopy of a Single View Screen Image 39
 - Hardcopy of a Typical Multiview 40

Chapter 2 Exercises 48

■ **Chapter 3 3-D Modeling Basics** 51

- 3.0 Introduction 51
- 3.1 Feature-based Modeling 51
- 3.2 Modeling in *Pro/E* – A First Example 52
 - Beginning the Base Feature 52
 - Setting Up the Sketching Plane 53
 - Creating the 2-D Profile or Sketch 55
 - Constraining the Sketch Geometry 56
 - Modifying Sketch Constraints 58
 - Completing the Base Feature 59
 - Adding a Detail Feature to the Base Feature 59
 - Adding the Hole Features to Complete the Model 61
- 3.3 Example 2: Circular Sweeps (Revolves) 64
- 3.4 Example 3: A Bigger Picture 69
 - Coordinate Systems in Part Creation 74
 - Datum Planes in Part Creation 77
 - Constraints in Part Creation 79
 - Parent-Child Relationships in Part Creation 80
 - Design Intent 82

Chapter 3 Exercises 83

■ **Chapter 4 Documentation – the Basics** 91

- 4.0 Introduction 91
- 4.1 Documentation Standards 92
- 4.2 Dimensioning Concepts 92
- 4.3 Documentation in *Pro/E* – the Basics 93
 - Using a Drawing Setup File 94
 - Obtaining the Necessary Views 95

	"Turning on" Existing Dimensions	98
	Adding Dimensions	99
	"Cleaning" the Dimensions	100
	"Turning on" Centerlines	101
	Editing Documentation Geometry	102
	Adding Notes (Manufacturing Information)	104
	A Summary of the Basic Documentation Process	105
4.4	The Model-Documentation Connection	105
	<i>Chapter 4 Exercises</i>	110
■	Chapter 5 Additional Modeling Tools	111
5.0	Introduction	111
5.1	Rectangular Arrays of Features	111
	Creating a Countersunk Hole	113
	Creating the Rectangular Array	115
	Suppressing Features	117
5.2	Radial Arrays of Features	118
	Creating a Counterbored Hole	118
	Creating the Radial Array	119
5.3	Relations	121
5.4	Offsets	124
5.5	Sweeps	125
5.6	Blends	128
5.7	Shells	130
5.8	Chamfers and Rounds	130
	<i>Chapter 5 Exercises</i>	132
■	Chapter 6 Creating an Assembly	135
6.0	Introduction	139
6.1	Creating an Assembly Model in <i>Pro/E</i>	139
	The Base Feature for an Assembly	140
	Constraining Components in an Assembly	142
	The Nature of the Assembly Model	148
6.2	Exploded Assemblies	148
6.3	Modifying Components in ASSEMBLY Mode	151

6.4	Relations in ASSEMBLY Mode	154
-----	----------------------------	-----

	<i>Chapter 6 Exercises</i>	157
--	----------------------------	-----

■ **Chapter 7 Additional Tools for Documentation** 161

7.0	Introduction	161
-----	--------------	-----

7.1	Section Views	161
	Full Section	163
	Half Section	165
	Offset Section	167
	Broken-out Section	170
	Revolved Section	172

7.2	Other Special View Types	173
	Auxiliary Views	173
	Z-clipped Views	175
	Detail Views	176

7.3	Additional Tools for Documentation Details	176
-----	--	-----

7.4	Dimensional Tolerances	184
-----	------------------------	-----

7.5	2-D Drafting	186
-----	--------------	-----

	<i>Chapter 7 Exercises</i>	188
--	----------------------------	-----

■ **Chapter 8 Model Organization** 189

8.0	Model Organization Schemes	189
-----	----------------------------	-----

8.1	<i>Pro/E</i> Organizational Scheme	189
	Layers in PARTS	190
	Layers in ASSEMBLIES	192
	Component Layers in ASSEMBLIES	195
	Layers in DRAWINGS	196

	<i>Chapter 8 Exercises</i>	200
--	----------------------------	-----

	<i>Bibliography</i>	201
--	---------------------	-----

	<i>Index</i>	203
--	--------------	-----

Introduction to Computers in Design

1.0 COMPUTER-AIDED DESIGN

It is difficult to grasp how rapidly computer technology is changing the way we live and work and play. Electronic brains in the form of microprocessors are parts of the cars we drive, the planes in which we fly, the televisions we watch, the telephone systems we communicate with, the automated tools we use to produce such products, and the computer-aided design (CAD) systems used to design such products and tools. Computer hardware for CAD systems continues to become faster and more powerful and less expensive. Much of the software continues to become more sophisticated in order to take advantage of the more powerful hardware, and also less expensive. And there is no apparent end in sight to such mind-boggling developments.

The first CAD packages were "Computer-aided Drafting" packages – little more than electronic equivalents of the drafting board. Though they offered some advantages over manual drafting, especially as the software became more sophisticated and "user friendly," the 3-D model was still only in the designer's mind, not in the computer file or *database*. 3-D computer modeling requires exponentially more computing power than 2-D electronic drafting, and the first computers, even mainframes, did not have that kind of computational power.

But the power of computers grew rapidly, especially after such developments as the *transistor* and *integrated circuit* (late 1950s), *microprocessor* or "computer on a chip" (1971), and *very large scale integration* or *VLSI* (thousands and eventually millions of electronic devices on a single chip of silicon). Developments in display technology such as the *raster scan display* also made sophisticated graphics applications more attractive as well as practical.

Such developments in computer hardware made possible substantial developments in 3-D computer modeling theory and software in the 1960s and 1970s. But these early efforts were carried on mostly by large, technology-intensive companies like McDonnell-Douglas and General Motors for their own applications and the software was therefore proprietary, at least initially. These early packages also required mainframe computers on which to run and so

were not practical for smaller companies, much less individual users.

It was the rapid development of large-scale integrated circuit technology in the 1980s and, in particular, the introduction of *32-bit processors* that made "personal computers" (PCs) and "personal workstations" possible. (*Bit* is short for *binary digit*; i.e., the 0 or 1 in the binary system of numbers. 32-bit processors handle data in 32-bit chunks.) Such developments brought the power of mainframes to the desktop and made 3-D modeling, along with a host of other sophisticated software applications, practical for an ever increasing number of users. The increasing power, speed, and affordability of computer hardware also made it attractive for software designers and programmers to develop increasingly sophisticated applications to take advantage of the newest hardware. The number and variety of 3-D computer-aided design packages increased dramatically, and most of these were available to anyone willing to pay.

The 3-D modeling packages developed in the late 1980s and early 1990s tended to fall into one of two categories – those that could run on smaller systems (PCs) and those that required the power of a high-end workstation or minicomputer. By the mid 1990s, however, PCs had become powerful enough to make that distinction blur. The newest PCs now come with processors that run at 800 *megahertz* (MHz) or faster, 128 or more *megabytes* (MB) of *random access memory* or RAM, hard disks of 30 *gigabytes* (GB) or more, and reasonably high resolution monitors. That is enough power to run many of the CAD packages that would run only on high-end workstations less than a decade ago.

This text is intended to be an introduction to computer modeling concepts that are characteristic of current 3-D modeling software. In particular, it is an introduction to the 3-D computer-aided design software *Pro/ENGINEER™*. *Pro/ENGINEER* is one of the more sophisticated CAD packages on the market, that, until fairly recently, would not run on a typical PC. All the examples in this text, however, were done on a PC with a 200 MHz Pentium II processor, 80 MB of RAM, and a 6 GB hard disk. By current PC standards (mid 2000), that is not all that much of a machine.

Modeling and the Design Process

We design (and build, manufacture, fabricate, etc.) things in order to solve a problem or fulfill a perceived need. Design is a primary activity in a wide variety of disciplines. Regardless of the design field, however, the process of design includes a characteristic set of "steps." A typical set of "steps" or activities would include:

- **Need Identification**
 - **Problem Definition**
 - **Search**
-

- **Determination of Constraints**
- **Selection of Criteria**
- **Development of Alternative Solutions**
- **Analysis**
- **Selection of Best Solution**
- **Specification of Best Solution**
- **Communication of Best Solution**

Though there is definite logic to the order in which these activities are listed, design is by nature an iterative process. Rarely, if ever, do we proceed step-by-step through the list to an acceptable result. Frequently, steps are repeated for one reason or another. Some steps, such as "Search" (which might include such things as literature reviews, existing product analyses, interviews, lab tests, etc.) are often done repeatedly, perhaps almost continuously throughout much of the design process.

The Electronic Model Database

During this design process, much information pertaining to the design and the process itself is generated. Computer systems offer a means of easy storage and retrieval for most, if not all, of this information. Ideally, the 3-D computer model *database* can serve as the central storehouse for all the information that describes one or more alternative designs as they develop (see Figure 1-1). The model database might include any or all of the following:

- **design components as 3-D solid models**
- **assembly and subassembly models**
- **production drawing files for nonstandard components**
- **manufacturing specifications**
- **bill of materials (or parts list, schedules, etc.)**
- **files required for CAM (computer-aided manufacturing) or CIM (computer-integrated manufacturing)**
- **results of any computer analyses done for the design**

The more tightly all this information is bound together, the better. Ideally, for example, we should be able to make a modification to a component model and see that change reflected in appropriate assemblies, production files, and analyses results either automatically or semiautomatically. Current high-end CAD packages, including *Pro/ENGINEER* (or *Pro/E* for short), have this tight integration of all design database information as a primary goal.

1.1 MODEL REPRESENTATION SCHEMES

Many current CAD packages allow us to create "models" using any

Figure 1-1 The computer and the computerized design database can ideally be the central storehouse for all the information gathered and used during the design process, from initial need identification to final production/construction.

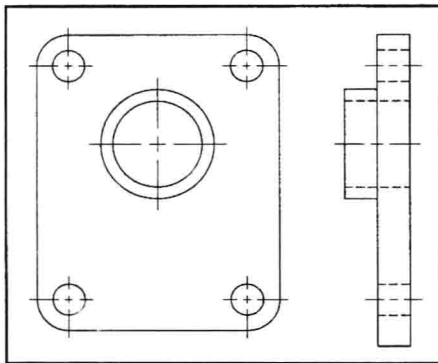
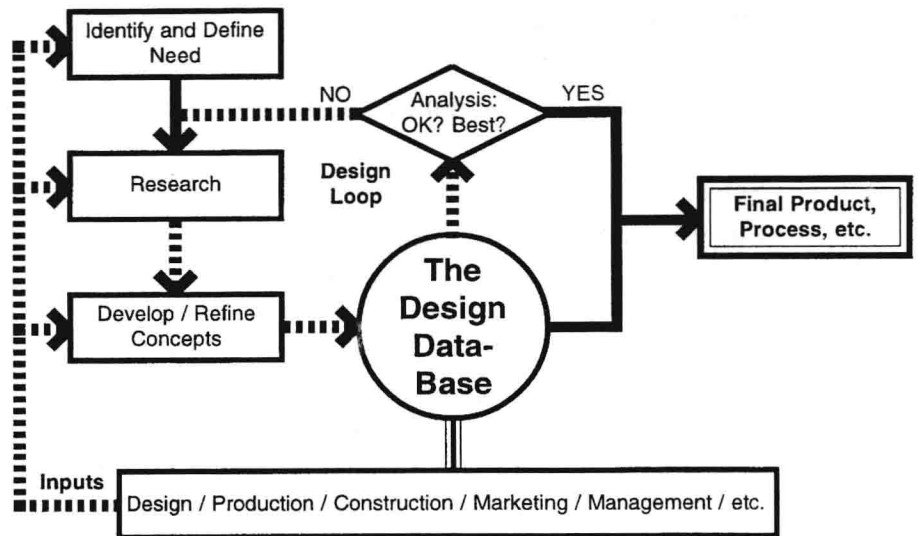


Figure 1-2 Traditional and computer-aided drafting use 2-D "wireframes" to represent 3-D models. But the 3-D model exists only in the designer's or drafter's mind.

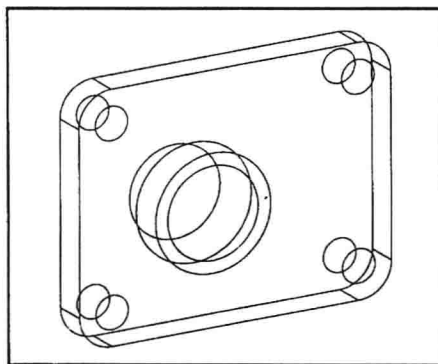


Figure 1-3 A 3-D wireframe is inherently ambiguous. For example, is the cylindrical projection facing toward us or away from us? Do the smallest circles represent "pins"? "holes"? some combination?

one of several representation schemes. For example, even if the software is a 3-D modeler, we might decide to ignore the 3rd dimension and use it to "draw" one or more views of an object on the screen as if we were drawing on paper. We might call this a "2-D wireframe" representation since we would be using a set of 2-D views to represent the 3-D model and we would be using lines or "wires" to represent the edges and surface boundaries seen in each 2-D view as illustrated in Figure 1-2. In this case, however, the 3-D model exists only in our minds, not in the computer database.

Wireframe Representation

Choices of 3-D representation schemes might include *wireframe*, *surface*, or *solid*. The representation scheme used determines the kinds of operations we can do on the model and the kinds of information that can be associated with the model. In a **3-D wireframe** representation, the model definition consists of nodes (or vertices or points) in *xyz* space and lines or wires between appropriate pairs of nodes, but no surface definitions (*connectivity*) or "solidness". This was the scheme employed in the first 3-D modelers because it is the simplest and requires the least computer power to achieve useful performance. But because surface definition is not part of a true wireframe representation, no "hiding" of what would be invisible edges in the real physical model is possible. A 3-D wireframe image is therefore inherently ambiguous as illustrated in Figure 1-3.

Surface Representation

A second possible 3-D representation scheme is a surface representation. In a **3-D surface** representation, the model definition consists of nodes (points) in *xyz* space, lines (edges) between appropriate pairs of nodes, and *connectivity*. Connectivity means that sets of edges are

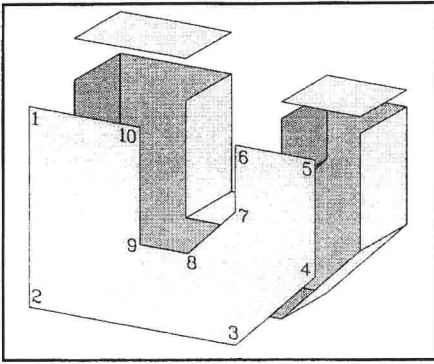


Figure 1-4 In a surface representation, edges are connected such that they enclose one or more surfaces. In the model above, for example, the "front" surface could be represented as the "connectivity list" 1-2-3-4-5-6-7-8-9-10 (vertices or "nodes").

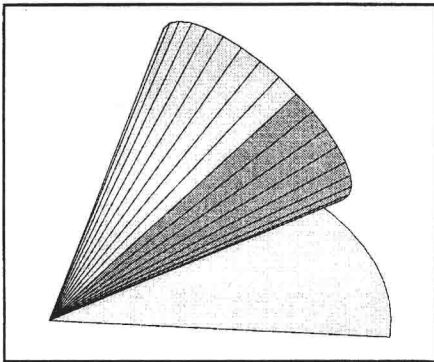


Figure 1-6 Single-curved surfaces have the advantage of being "developable," a useful characteristic in many manufacturing and fabrication operations.

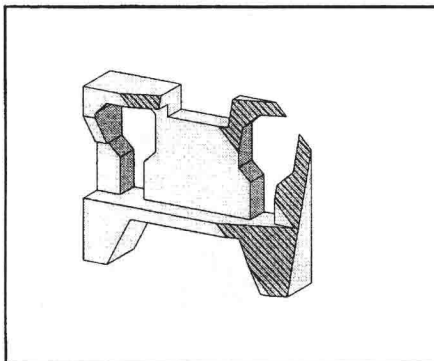


Figure 1-7 If a computer model is defined as a solid, cutting through the model with a "cutting plane" or other solid will produce new surfaces.

somehow "linked" together by the software such that they enclose one or more "surfaces" (see Figure 1-4). Surface models make possible such things as *hides* (hidden line and surface routines for showing correct visibility), surface colors, textures, and patterns, and analyses based on surfaces (area, perimeter, planar relationships, etc.). Such a scheme therefore offers important advantages compared to wireframe representation.

Surface Types

Good modelers provide tools for creating a variety of surface types including *planar*, *single-curved*, *double-curved*, and *warped* surfaces (see Figure 1-5 on page 6). Though planar and single-curved surfaces are the most commonly used surface types, double-curved and warped surfaces also have important applications. Methods for defining complex curves and curved surfaces have been developed specifically for computer modeling.

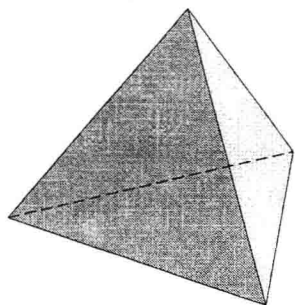
What characterizes most single-curved surfaces is that all the lines or elements that help define the surface are either parallel (as in cylinders) or intersect at a common point (as in cones). Cones and cylinders are by far the most common single-curved surfaces. A useful characteristic of single-curved surfaces is that they are developable; i.e., they can be rolled out or "developed" onto a planar surface (see Figure 1-6).

Double-curved surfaces, as the name implies, are characterized by curvature in two directions. A straight line can lie completely in a single-curved surface, but not in a double-curved surface. Spheres and spheroids are the most common examples of double-curved surfaces, but this category includes a host of more exotic types including some specifically developed for computer applications.

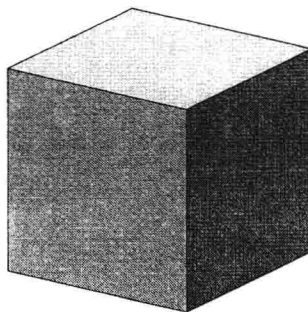
Warped surfaces are characterized by straight-line elements that are skew to one another (i.e., nonparallel and non-intersecting elements). Two fairly common warped surface types are hyperbolic paraboloid and hyperboloid (see Figure 1-5) but this category also includes quite an assortment. Though warped surfaces tend to be difficult to construct or fabricate, they have the structural advantage of being inherently rigid and are useful as *transitional* surfaces.

Solid Representation

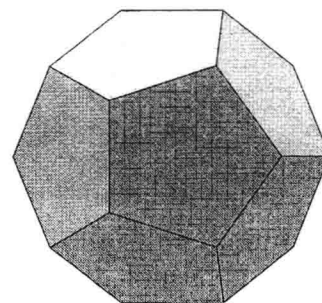
The third and most sophisticated 3-D representation scheme is a *solid* representation. A solid representation includes nodes, edges, and surface definitions like a surface representation scheme, but in addition, a "key" to differentiate "inside" from "outside" of an appropriate set of surfaces. It is a complete and unambiguous mathematical representation of a precisely enclosed and "filled" volume. Since the software defines the volume as "filled", a modeling operation that somehow "cuts" through the volume will produce one or more new



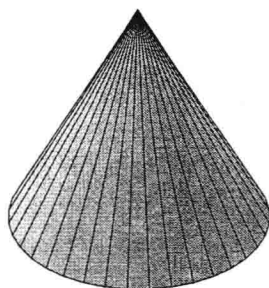
(a) tetrahedron



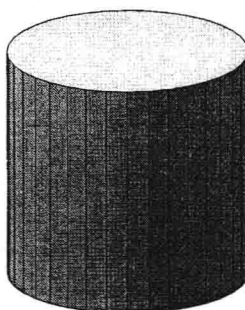
(b) cube



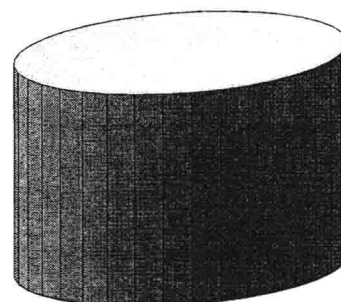
(c) dodecahedron



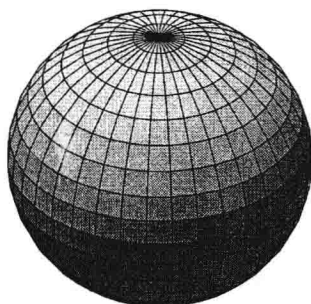
(d) cone



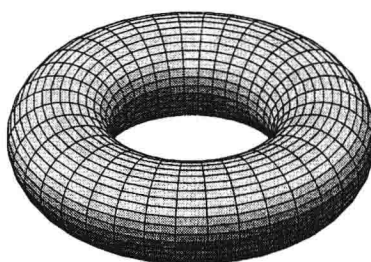
(e) circular cylinder



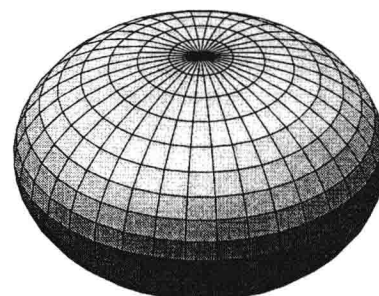
(f) elliptical cylinder



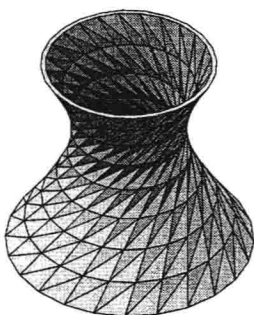
(g) sphere



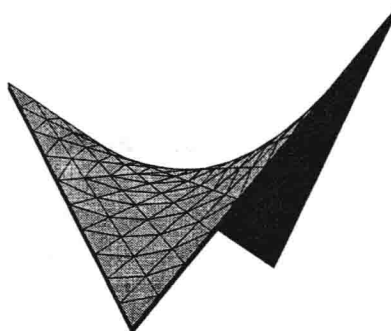
(h) torus



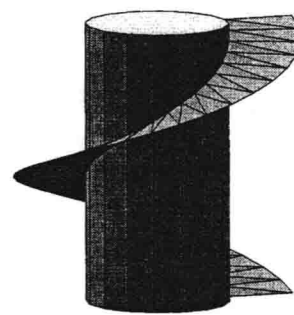
(i) ellipsoid (oblate)



(j) hyperboloid (one nappe)



(k) hyperbolic paraboloid



(l) helicoid (around cylinder)

Figure 1-5 Various surface types may be used to define or help define solid models (or surface models). Surface types include planar (a, b, and c), single-curved (d, e, and f), double-curved (g, h, and i), and warped (j, k, and l).

surfaces as suggested in Figure 1-7.

1.2 SOLID MODEL CREATION TECHNIQUES

Though several techniques have been developed for creating and defining solid models, the two most commonly provided or applied in CAD software are *constructive solid geometry (CSG)* and *sweeping techniques* (see Figures 1-8 and 1-9). In practical modeling, various combinations of these two techniques are typically used to create the final model.

Constructive Solid Geometry

Constructive Solid Geometry or *CSG* can be defined as the *combining* of 3-D *solid primitives* in various ways to create more complex objects. What is considered a “primitive” is rather arbitrary, though modelers that offer basic CSG techniques usually provide definitions for the simpler 3-D solid shapes such as *block*, *pyramid*, *cylinder*, *cone*, *sphere*, etc. As applied in practical CSG modeling, however, a “primitive” can be very complex itself, and, in fact, often is. The combining is done using what are called *Boolean operators* (or *logical* or *set operators*) applied to 3-D solid geometry.

In some modelers, the solid model is stored as a CSG tree (or binary tree). In this form, the model is represented as a set of primi-

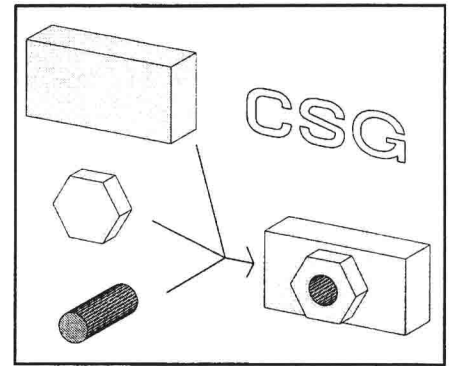


Figure 1-8 Constructive Solid Geometry involves the combining of simple solid shapes to produce more complex solid shapes.

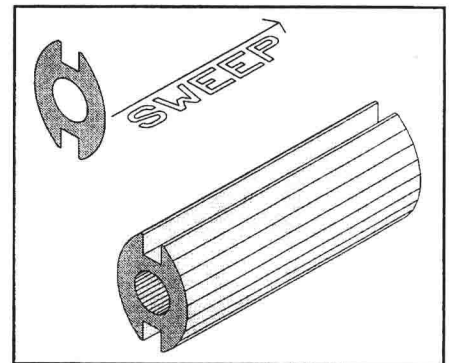


Figure 1-9 Sweeping involves moving geometry through space to produce new geometry.

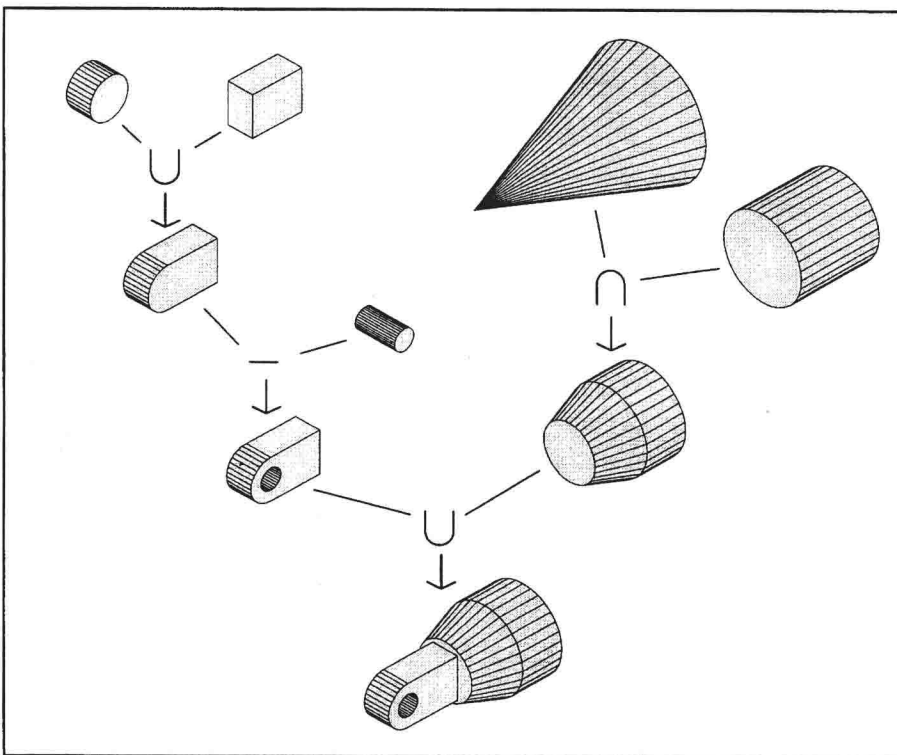


Figure 1-10 If a computer model is created using Constructive Solid Geometry techniques, it can be represented and stored as a **CSG tree**.

tive shapes which are related to one another using an appropriate set of Boolean operators as illustrated in Figure 1-10. The shape, size, orientation, and location of each primitive, along with the Boolean operations that relate each primitive to one or more other primitives is stored in the model database. In a sense, such a CSG tree is like a recipe for the final model.

Boolean Operators

There are three basic Boolean operators, but current modelers usually provide the basic three plus several to many variations on the three. The three basic Boolean operators are called **union**, **difference**, and **intersection**. In CSG, the **union** operator (mathematical symbol \cup) combines two solids or volumes into a single solid or volume (see Figure 1-11). The union operator corresponds to the **OR** logical operator ($A \cup B = \text{all of the volume in } A \text{ OR } B$). In theory, the two initial solids can be overlapping, just touching, or some distance apart. Some modelers only allow the union operation under the first two conditions since making two separated solids into a single piece without moving one of them is not physically meaningful. For the

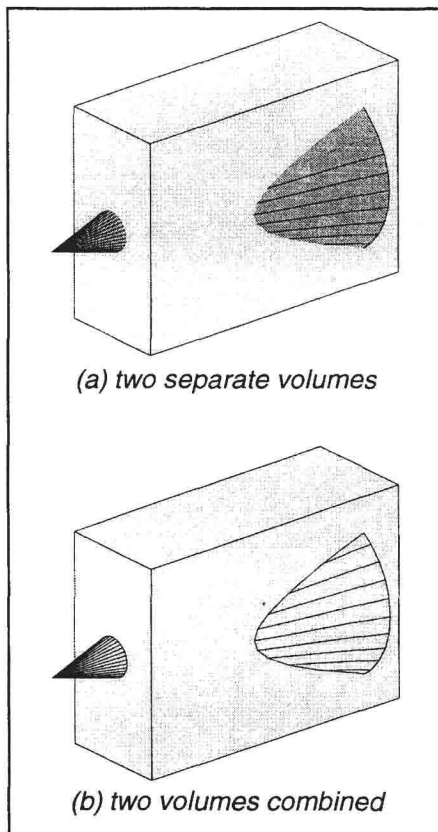


Figure 1-11 The **UNION** operator combines two volumes or solids as in (a) into a single volume or solid as in (b).

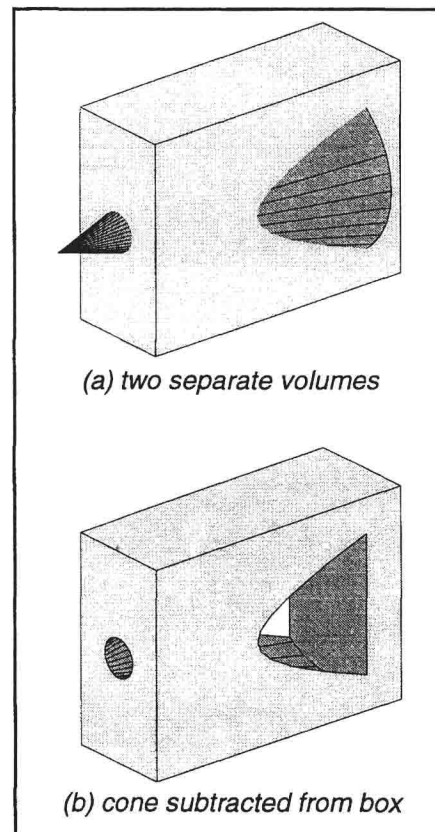


Figure 1-12 The **SUBTRACTION** operator subtracts the volume of one solid from that of another.

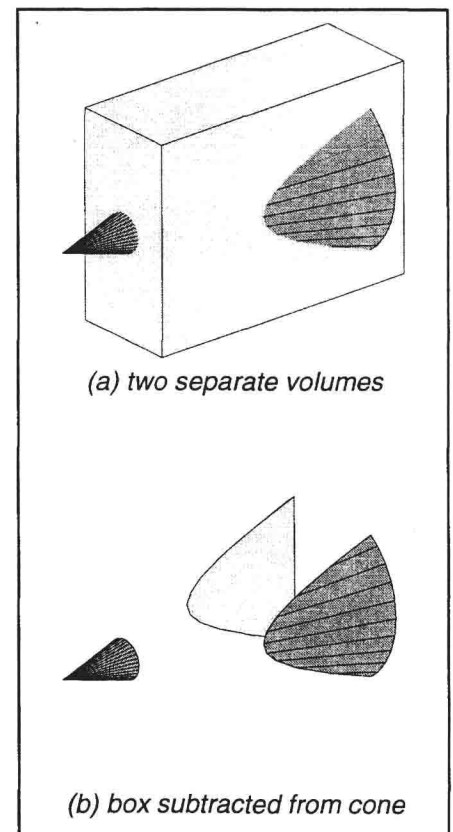


Figure 1-13 For the **SUBTRACTION** operator, order of operands is important to the resulting geometry (compare Figure 1-12).

union operator, order of operands does not effect the geometry of the result; i.e., $A \cup B = B \cup A$.

The second of the three is the **difference** or **subtraction** operator (mathematical symbol $-$). The difference operation removes the volume of one solid from that of a second solid (see Figure 1-12); the solid subtracted is generally deleted from the model. The difference operator corresponds to the **NOT** logical operator ($A - B =$ all the volume in A but NOT that which is also in B). For the difference operation, order of operands does effect the outcome; i.e., in general, $A - B \neq B - A$ as illustrated in Figure 1-13. The difference operation can result in one or more new solids.

The third and last of the basic Boolean operators is the **intersection** operator (mathematical symbol \cap). The intersection operator keeps only that volume common to two solids (see Figure 1-14). The intersection operator corresponds to the logical **AND** ($A \cap B =$ all the volume common to both A AND B). As for the union operator, order of operands does not effect the geometry of the outcome; i.e., $A \cap B = B \cap A$.

Sweeping

The second technique provided by typical modelers for creating solid models is **sweeping**. Sweeping is a valuable modeling tool since there are some shapes that are difficult if not impossible to create with CSG alone (see Figure 1-15). Sweeps might be used to create

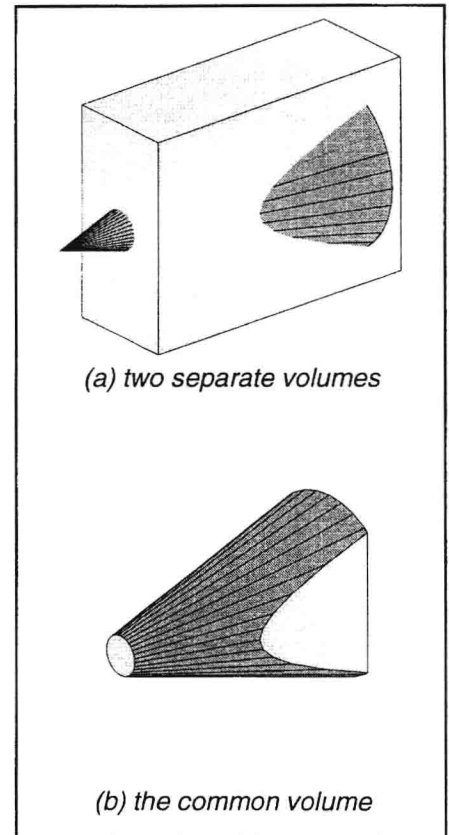


Figure 1-14 The **INTERSECTION** operator finds the volume common to both of the initial volumes.

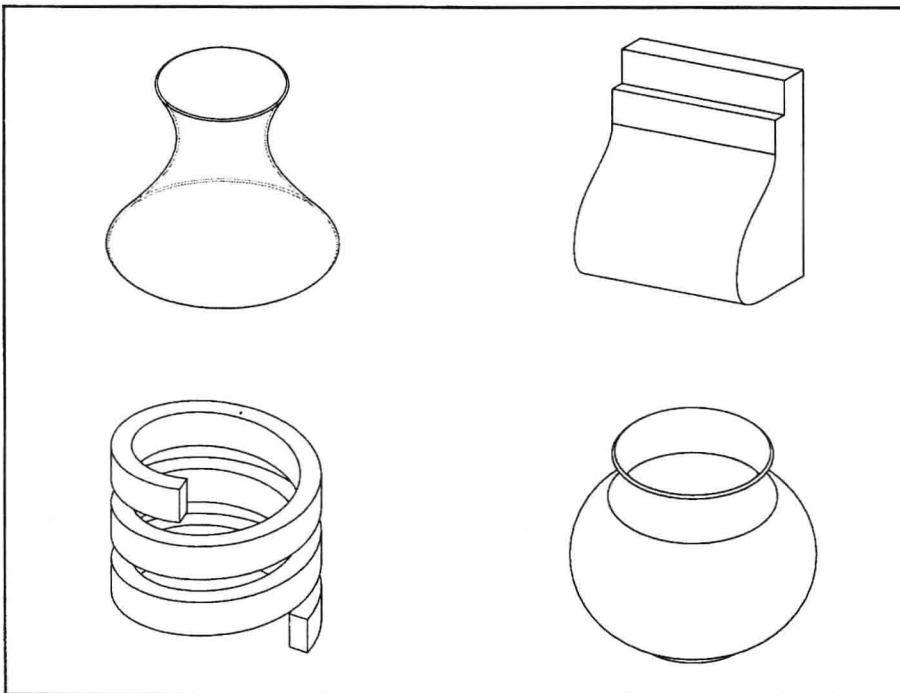


Figure 1-15 Some shapes are difficult if not impossible to create using CSG techniques but quite easy to create with sweeping techniques.

wireframe and surface models as well as solid models, depending on how the CAD package works. In theory, moving (sweeping) any geometry (point, line, surface, solid) through space along an arbitrary path defines new geometry. In practice, the shapes and paths that are allowed, and the resulting model definition, are a function of the software and the limitations of the computer hardware.

The most common types of sweeps in computer modeling use a planar shape as the *generator* and a straight or curved line as the *path* of the sweep (see Figure 1-16). In theory, a straight line path might be perpendicular, oblique, or parallel to (in the plane of) the planar *generator*. In some modelers, the path for linear sweeps is always assumed to be perpendicular to the plane of the generator, though sweep paths that are oblique to the plane of the generator are

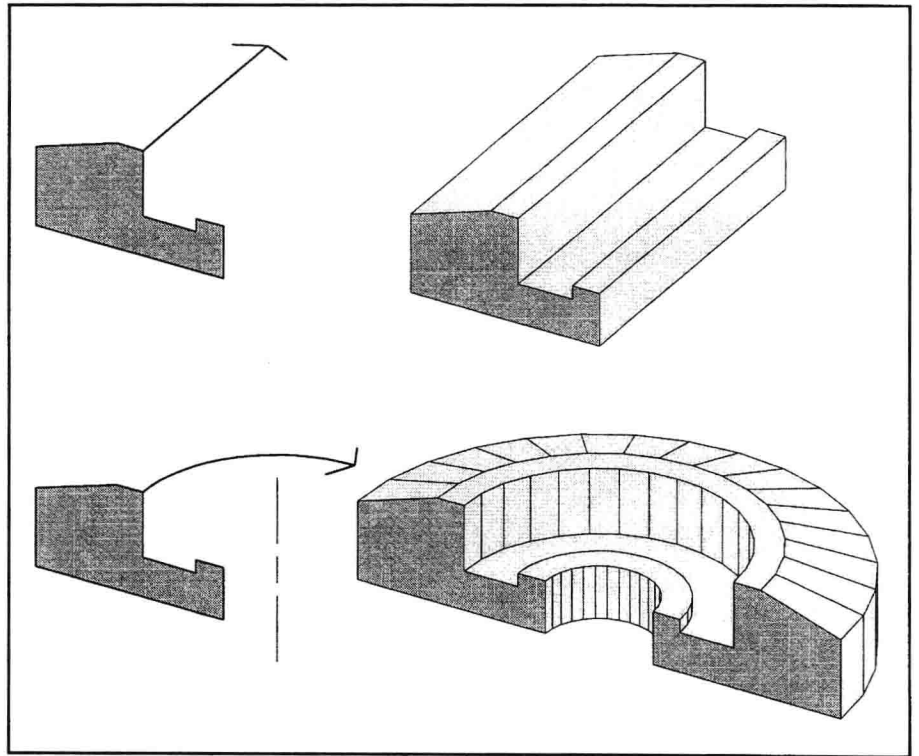


Figure 1-16 The most common sweeps involve a planar generator and either a straight or curved line path.

also sometimes useful. Some modelers, including *Pro/E*, use the term **extrusion** instead of linear sweep.

Circular sweeps are the most common curved-path sweeps, though other types of curved path sweeps such as spiral, elliptical, and more “exotic” forms can also be useful. Most current modelers offer circular sweeps and many, including *Pro/E*, offer some more exotic variations as well. Some modelers use the term **revolve** instead of circular sweep. Figure 1-17 illustrates several variations of a curved-path sweep.