Resolution can be as high as 400 × 800 dpi, with gray scales ranging from 16–128 values. These are medium- to high-throughput devices, producing complex images in about a minute. On-board computing facilities, such as RISC processors and fast hard disk storage mechanisms, contribute to rapid drawing and processing speeds. Expansion slots accommodate interface cards for LANs or parallel ports.

Inkjet Plotter. Inkjet plotters and printers fire tiny ink droplets at paper or a similar medium from minute nozzles in the printing head. Heat generated by a separate heating element almost instantaneously vaporizes the ink. The resulting bubble generates a pressure wave that ejects an ink droplet from the nozzle. Once the pressure pulse passes, ink vapor condenses and the negative pressure produced as the bubble contracts draws fresh ink into the nozzle. These plotters do not require special paper and can also be used for preliminary drafts. Inkjet plotters are available both as desktop units for 8.5 × 11-in. graphics and in wide format for engineering CAD drawings. Typical full-color resolution is 360 dpi, with black-and-white resolution rising to 700 × 720 dpi. These devices handle both roll-feed and cut sheet media in widths ranging from 8.5–36 in. Also, ink capacity in recently developed plotters has increased, allowing these devices to handle large rolls of paper without depleting any one ink color. Inkjet plotters are very user-friendly, often including sensors for the ink supply and ink flow that warn users of an empty cartridge or of ink stoppage, allowing replacement without losing a print. Other sensors eliminate printing voids and unwanted marks caused by bubbles in the ink lines. Special print modes typically handle high-resolution printing by repeatedly going over image areas to smooth image lines. In addition, inkjet plotters typically contain 6–64 megabytes of image memory and options such as hard drives, an Ethernet interface for networking, and built-in Postscript interpreters for faster processing. Inkjet plotters and printers are increasingly dominating other output technologies, such as pen plotters, in the design laboratory.

Laser Plotter. Laser plotters produce fairly high-quality hard copies in a shorter period of time than pen plotters. A laser housed within the plotter projects rasterized image data in the form of light onto a photostatic drum. As the drum rotates further about its axis, it is dusted with an electrically charged powder known as toner. The toner adheres to the drum wherever the drum has been charged by the laser light. The paper is brought into contact with the drum and the toner is released onto the paper, where it is fixed by a heat source close to the exit point. Laser plotters can quickly produce images in black and white or in color, and resolution is high.

13.7 SOFTWARE

Software is the collection of executable computer programs including operating systems, languages, and application programs. All of the hardware described above can do nothing without software to support it. In its broadest definition, software is a group of stored commands, sometimes known as a program, that provides an interface between the binary code of the CPU and the thought processes of the user. The commands provide the CPU with the information necessary to drive graphical displays and other output devices, to establish links between input devices and the CPU. The commands also define paths that enable other command sequences to operate. Software operates at all levels of computer function. Operating systems are a type of software that provides a platform upon which other programs may run. Likewise, individual programs often provide a platform for the operation of subroutines, which are smaller programs dedicated to the performance of specific tasks within the context of the larger program.

13.7.1 Operating Systems

Operating systems have developed over the past 50 years for two main purposes. First, operating systems attempt to schedule computational activities to ensure good performance of the computing system. Second, they provide a convenient environment for the development and execution of programs. An operating system may function as a single program or as a collection of programs that interact with each other in a variety of ways.

An operating system has four major components: process management, memory management, input/output operations, and file management. The operating system schedules and performs input/output, allocates resources and memory space and provides monitoring and security functions. It governs the execution and operation of various system programs and applications such as compilers, databases, and CAD software.

Operating systems that serve several users simultaneously (e.g., UNIX) are more complicated than those serving only a single user (e.g., MS-DOS, Macintosh Operating System). The two main themes in operating systems for multiple users are multiprogramming and multitasking.

Multiprogramming provides for the interleaved execution of two or more computer programs (jobs) by a single processor. In multiprogramming, while the current job is waiting for the input/output (I/O) to complete, the CPU is simply switched to execute another job. When that job is waiting for I/O to complete, the CPU is switched to another job, and so on. Eventually, the first job completes its I/O functions and is serviced by the CPU again. As long as there is some job to complete, the CPU remains active. Holding multiple jobs in memory at one time requires special hardware to protect each job, some form of memory management, and CPU scheduling. Multiprogramming increases CPU use and decreases the total time needed to execute the jobs, resulting in greater throughput.

The techniques that use multiprogramming to handle multiple interactive jobs are referred to as *multitasking* or *time-sharing*. Multitasking or time-sharing is a logical extension of multiprogramming for situations where an interactive mode is essential. The processor's time is shared among multiple users. Time-sharing was developed in the 1960s, when most computers were large, costly mainframes. The requirement for an interactive computing facility could not be met by the use of a dedicated computer. An interactive system is used when a short response time is required. Time-sharing operating systems are very sophisticated, requiring extra disk management facilities and an on-line file system having protective mechanisms as well.

The following sections discuss the two most widely used operating systems for CAD applications, UNIX and Windows NT. It should be noted that both of these operating systems can run on the same hardware architecture.

UNIX

The first version of UNIX was developed in 1969 by Ken Thompson and Dennis Ritchie of the Research Group of Bell Laboratories to run on a PDP-7 minicomputer. The first two versions of UNIX were created using assembly language, while the third version was written using the C programming language. As UNIX evolved, it became widely used at universities, research and government institutions, and eventually in the commercial world. UNIX quickly became the most portable of operating systems, operable on almost all general-purpose computers. It runs on personal computers, workstations, minicomputers, mainframes, and supercomputers. UNIX has become the preferred program-development platform for many applications, such as graphics, networking, and databases. A proliferation of new versions of UNIX has led to a strong demand for UNIX standards. Most existing versions can be traced back to one of two sources: AT&T System V or 4.3 BSD (Berkeley UNIX) from the University of California, Berkeley (one of the most influential versions).

UNIX was designed to be a time-sharing, multi-user operating system. UNIX supports multiple processes (multiprogramming). A process can easily create new processes with the fork system call. Processes can communicate with pipes or sockets. CPU scheduling is a simple priority algorithm. Memory management is a variable-region algorithm with swapping supported by paging. The file system is a multilevel tree that allows users to create their own subdirectories. In UNIX, I/O devices such as printers, tape drives, keyboards, and terminal screens are all treated as ordinary files (file metaphor) by both programmers and users. This simplifies many routine tasks and is a key component in extensibility of the systems. Certifiable security that protect users' data and network support are also two important features.

UNIX consists of two separable parts: the kernel and the system programs. The kernel is the collection of software that provides the basic capabilities of the operating system. In UNIX, the kernel provides the file system, CPU scheduling, memory management, and other operating system functions (I/O devices, signals) through system calls. System calls can be grouped into three categories: file manipulation, process control, and information manipulation. Systems programs use the kernel-supported system calls to provide useful functions, such as compilation and file manipulation. Programs, both system and user-written, are normally executed by a command interpreter. The command interpreter is a user process called a *shell*. Users can write their own shell. There are, however, several shells in general use. The Bourne shell, written by Steve Bourne, is the most widely available. The C shell, mostly by Bill Joy, is the most popular on BSD systems. The Korn Shell, by David Korn, has also become quite popular in recent years.

Windows NT

The development effort for the new high-end operating system in the Microsoft Windows family, Windows NT (New Technology), has been led by David Culter since 1988. Market requirements and sound design characteristics shaped the Windows NT development. The architects of "NT," as it is popularly known, capitalized on the strengths of UNIX while avoiding its pitfalls. Windows NT and UNIX share striking similarities. There are also marked differences between the two systems. UNIX was designed for host-based terminal computing (multi-user) in 1969, while Windows NT was designed for client/server distributed computing in 1990. The users on single-user general-purpose workstations (clients) can connect to multi-user general-purpose servers with the processing load shared between them. There are two Windows NT-based operating systems: Windows NT Server and Windows NT Workstation. The Windows NT Workstation is simply a scaled-down version of Windows NT Server in terms of hardware and software. Windows NT is a microkernel-based operating system. The operating system runs in privileged processor mode (kernel mode) and has access to system data and hardware. Applications run on a non-privileged processor mode (user mode) and have limited access to system data and hardware through a set of digitally controlled application programming interfaces (APIs). Windows NT also supports both single-processor and symmetric multiprocessing (SMP) operations. Multiprocessing refers to computers with more than one processor. A multiprocessing computer is able to execute multiple threads simultaneously, one for each processor in the computer. In SMP, any processor can run any type of thread. The processors communicate with each other through shared memory. SMP provides better load-balancing and fault-tolerance. The Win32 subsystem is the most critical of the Windows NT environment subsystems. It provides the graphical user interface and controls all user input and application output.

Windows NT is a fully 32-bit operating system with all 32-bit device drivers, paving the way for future development. It makes administration easy by providing more flexible built-in utilities and removes diagnostic tools. Windows NT Workstation provides full crash protection to maximize uptime and reduce support costs. Windows NT is a complete operating system with fully integrated networking, including built-in support for multiple network protocols. Security is pervasive in Windows NT to protect system files from error and tampering. The NT file system (NTFS) provides security for multiple users on a machine.

Windows NT, like UNIX, is a portable operating system. It runs on many different hardware platforms and supports a multitude of peripheral devices. It integrates preemptive multitasking for both 16- and 32-bit applications into the operating system, so it transparently shares the CPUs among the running applications. More usable memory is available due to advanced memory features of Windows NT. There are more than 1400 32-bit applications available for Windows NT today, including all major CAD and FEA software applications.

Hardware requirements for the Windows NT operating system fall into three main categories: processor, memory, and disk space. In general, Windows NT Server requires more in each of the three categories than does its sister operating system, the Windows NT Workstation. The minimum processor requirements are a 32-bit x86-based microprocessor (Intel 80386/25 or higher), Intel Pentium, Apple Power-PC, or other supported RISC-based processor, such as the MIPS R4000 or Digital Alpha AXP. The minimum memory requirement is 16 MB. The minimum disk space requirements for just the operating system are in the 100-MB range. NT Workstation requires 75 MB for x86 and 97 MB for RISC. For the NT Server, 90 MB for x86 and 110 MB for RISC are required. There is no need to add additional disk space for any application that is run on the NT operating system.

13.7.2 Graphical User Interface (GUI) and the X Window System

DOS, UNIX, and other command-line operating systems have long been criticized for the complexity of their user interface. For this reason, GUI is one of the most important and exciting developments of this decade. The emergence of GUI revolutionized the methods of man-machine interaction used in the modern computer. GUIs are available for almost every type of computer and operating system on the market. A GUI is distinguished by its appearance and by the way an operator's actions and input options are handled. There are over a dozen GUIs. They may look slightly different, but they all share certain basic similarities. These include the following: a pointing device (mouse or digitizer), a bit-mapped display, windows, on-screen menus, icons, dialog boxes, buttons, sliders, check boxes, and an object-action paradigm. Simplicity, ease of use, and enhanced productivity are all benefits of a GUI. GUIs have fast become important features of CAD software.

Graphical user interface systems were first envisioned by Vannevar Bush in a 1945 journal article. Xerox was researching graphical user interface tools at the Palo Alto Research Center throughout the 1970s. By 1983, every major workstation vendor had a proprietary window system. It was not until 1984, however, when Apple introduced the Macintosh computer, that a truly robust window environment reached the average consumer. In 1984, a project called Athena at MIT gave rise to the X Window system. Athena investigated the use of networked graphics workstations as a teaching aid for students in various disciplines. The research showed that people could learn to use applications with a GUI much more quickly than by learning commands.

The X Window system is a non-vendor-specific window system. It was specifically developed to provide a common window system across networks connecting machines from different vendors. Typically, the communication is via Transmission Control Protocal/Internet Protocal (TCP/IP) over an Ethernet network. The X Window system (X-Windows or X) is not a GUI. It is a portable, network-transparent window system that acts as a foundation upon which to build GUIs (such as AT&T's OpenLook, OSF/Motif, and DEC Windows). The X Window system provides a standard means of communicating between dissimilar machines on a network and can be viewed in a window. The unique benefit provided by a window system is the ability to have multiple views showing different processes on different networks. Since the X Window system is in the public domain and not specific to any platform or operating system, it has become the de facto window system in heterogeneous environments from PCs to mainframes.

Unfortunately, a window environment does not come without a price. Extra layers of software separate the user and the operating system, such as window system, GUI, and an Application Programming Interface (ToolKit) in a UNIX operating environment. GUIs also place extra demands on hardware. All visualization workstations require more powerful processing capabilities ($>$ 6 MIPS), large CPU memory and disk subsystems, built-in network Input/Output (I/O) with typically Ethernet high-speed internal bus structures ($\geq$ 32 MB/sec)—high-resolution monitors ($\geq$ 1024 $\times$ 768), more colors (> 256), and so on.

For PCs, both operating systems and GUIs are in a tremendous state of flux. Microsoft Windows, Windows NT, and Windows 95 are expected to dominate the market, followed by the Macintosh. For workstations, the OSF/Motif interface on an X-Windows system seems to have the best potential to become an industry-wide graphical user interface standard.

13.7.3 Computer Languages

The computer must be able to understand the commands it is given in order to perform desired tasks at hand. The binary code used by the computer circuitry is very easy for the computer to understand, but can be tedious and almost indecipherable to the human programmer. Languages for computer programming have developed to facilitate the programmer's job. Languages are often categorized as low- or high-level languages.

Low-Level Languages

The term *low-level* refers to languages that are easy for the computer to understand. These languages are often specific to a particular type of computer, so that programs created on one type of computer must be modified to run on another type. Machine language (ML) and assembly language (AL) are both considered low-level languages.

Machine language is the binary code that the computer understands. ML uses an operator command coupled with one or more operands. The operator command is the binary code for a specific function, such as addition. The numbers to be added, in this example, are operands. Operators are also binary codes, arbitrary with respect to the machine used. For a hypothetical computer, all operator codes are established to be eight digits, with the operator command appearing after the two operands. If the operator code for addition then were 01100110, the binary (base 2) representation of the two numbers added would be followed by the code for addition. A command line to perform the addition of 21 and 14 would then be written as follows:

000101010000111001100110

The two operands are written in their 8 bit binary forms (21_{10} as 00010101_2 and 14_{10} as 00001110_2) and are followed by the operator command (01100110 for addition). The binary nature of this language makes programming difficult and error-correction even more so.

AL operates in a similar manner to ML but substitutes words for machine codes. The program is written using these one-to-one relationships between words and binary codes and separately assembled through software into binary sequences. Both ML and AL are time-intensive for the programmer and, because of the differences in logic circuitry between types of computers, the languages are specific to the computer being used. High-level languages address the problems presented by these low-level languages in various ways.

High-Level Languages (HLLs)

High-level languages give the programmer the ability to bypass much of the tediousness of programming involved in low-level languages. Often many ML commands will be combined within one HLL statement. The programming statements in HLL are converted to ML using a compiler. The compiler uses a low-level language to translate the HLL commands into ML and check for errors. The net gain in terms of programming time and accuracy far outweighs the extra time required to compile the code. Because of their programming advantages, HLLs are far more popular and widely used than low-level languages. The following commonly used programming languages are described below:

- FORTRAN
- Pascal
- BASIC
- C
- C++

FORTRAN (FORmula TRANslation). Developed at IBM between 1954 and 1957 to perform complex calculations, this language employs a hierarchical structure similar to that used by mathematicians to perform operations. The programmer uses formulas and operations in the order that would be used to perform the calculation manually. This makes the language very easy to use. FORTRAN can perform simple as well as complex calculations. FORTRAN is used primarily for scientific or engineering applications. CFP95 Suite, a software benchmarking product by Standard

Performance Evaluation Corp. (SPEC) is written in FORTRAN. It contains 10 CPU-intensive floating point benchmarks.

The programming field in FORTRAN is composed of 80 columns, arranged in groups relating to a programming function. The label or statement number occupies columns 1–5. If a statement extends beyond the statement field, a continuation symbol is entered in column 6 of the next line, allowing the statement to continue on that line. The programming statements in FORTRAN are entered in columns 7–72. The maximum number of lines in a FORTRAN statement is 20. Columns 73–80 are used for identification purposes. Information in these columns is ignored by the compiler, as are any statements with a C entered in column 1.

Despite its abilities, there are several inherent disadvantages to FORTRAN. Text is difficult to read, write, and manipulate. Commands for program flow are complicated and a subroutine cannot go back to itself to perform the same function.

Pascal. Pascal is a programming language with many different applications. It was developed by Niklaus Wirth in Switzerland during the early 1970s and named after the French mathematician Blaise Pascal. Pascal can be used in programs relating to mathematical calculations, file processing and manipulation, and other general-purpose applications.

A program written in Pascal has three main sections: the program name, the variable declaration, and the body of the program. The program name is typically the word *PROGRAM* followed by its title. The variable declaration includes defining the names and types of variables to be used. Pascal can use various types of data and the user can also define new data types, depending on the requirements for the program. Defined data types used in Pascal include strings, arrays, sets, records, files, and pointers. Strings consist of collections of characters to be treated as a single unit. Arrays are sequential tables of data. Sets define a data set collected with regard to sequence. Records are mixed data types organized into a hierarchical structure. Files refer to collections of records outside of the program itself, and pointers provide flexible referencing to data. The body of the program uses commands to execute the desired functions. The commands in Pascal are based on English and are arranged in terms of separate procedures and functions, both of which must have a defined beginning and end. A function can be used to execute an equation and a procedure is used to perform sets of equations in a defined order. Variables can be either "global" or "local," depending on whether they are to be used throughout the program or within a particular procedure. Pascal is somewhat similar to FORTRAN in its logical operation, except that Pascal uses symbolic operators while FORTRAN operates using commands. The structure of Pascal allows it to be applicable to areas other than mathematical computation.

BASIC (Beginners All-Purpose Symbolic Interactive Code). BASIC was developed at Dartmouth College by John Kemeny and Thomas Kurtz in the mid-1960s. BASIC uses mathematical programming techniques similar to FORTRAN and the simplified format and data manipulation capabilities similar to Pascal. As in FORTRAN, BASIC programs are written using line numbers to facilitate program organization and flow. Because of its simplicity, BASIC is an ideal language for the beginning programmer. BASIC runs in either direct or programming modes. In the direct mode, the program allows the user to perform a simple command directly, yielding an instantaneous result. The programming mode is distinguished by the use of line numbers that establish the sequence of the programming steps. For example, if the user wishes to see the words *PLEASE ENTER DIAMETER* displayed on the screen immediately, he would execute the command *PRINT "PLEASE ENTER DIAMETER."* If, however, that phrase were to appear in a program, the above command would be preceded by the appropriate line number.

The compiler used in the BASIC language is unlike the compiler used for either FORTRAN or Pascal. Whereas other HLL compilers check for errors and execute the program as a whole unit, a BASIC program is checked and compiled line by line during program execution. BASIC is often referred to as an "interpreted" language as opposed to a compiled one, since it interprets the program into ML line by line. This condition allows for simplified error debugging. In BASIC, if an error is detected, it can be corrected immediately, while in FORTRAN and Pascal, the programmer must go back to the source program in order to correct the problem and then recompile the program as a separate step. The interpretive nature of BASIC does cause programs to run significantly more slowly than in either Pascal or FORTRAN.

C. C was developed from the B language by Dennis Ritchie in 1972. C was standardized by the late 1970s when B. W. Kernighan and Ritchie's book *The C Programming Language* was published. C was developed specifically as a tool for the writing of operating systems and compilers. It originally became most widely known as the development language for the UNIX operating systems. C expanded at a tremendous rate over many hardware platforms. This led to many variations and a lot of confusion and, while these variations were similar, there were notable differences. This was a problem for developers that wanted to write programs that ran on several platforms. In 1989, the American National Standards Committee on Computers and Information Processing approved a standard version of C. This version is known as *ANSI C* and it includes a definition of a set of library routines for file operations, memory allocation, and string manipulation.

A program written in C appears similar to Pascal. C, however, is not as rigidly structured as Pascal. There are sections for the declaration of the main body of the program and the declaration of variables. C, like Pascal, can use various types of data and the programmer can also define new data types. C has a rich set of data types, including arrays, sets, records, files, and pointers. C allows for far more flexibility than Pascal in the creation of new data types and the implementation of existing data types. Pointers in C are more powerful than they are in Pascal. Pointers are variables that point not to data but to the memory location of data. Pointers also keep track of what type of data is stored there. A pointer can be defined as a pointer to an integer or a pointer to a character. CINT95 Suite, a software benchmarking product, is written in C. It contains eight CPU-intensive integer benchmarks.

C++. C++ is a superset of the C language developed by Bjarne Stroustrup in 1986. C++'s most important addition to the C language is the ability to do object-oriented programming. Object-oriented programming places more emphasis on the data of a program. Programs are structured around objects. An object is a combination of the program's data and code. Like a traditional variable, an object stores data, but unlike traditional languages, objects can also do things. For example, an object called *triangle* might store both the dimensions of the triangle and the instructions on how to draw the triangle. Object-oriented programming has led to a major increase in productivity in the development of applications over traditional programming techniques.

A program written in C++ no longer resembles C or Pascal. More emphasis is placed on a modular design around objects. The main section of a C++ code should be very small and may only call one or two functions, and the declaration of variables in the main function should be avoided. Global variables and functions are avoided at all costs and the use of variables in local objects is stressed. The avoidance of global variables and functions that do large amounts of work is intended to increase security and make programs easier to develop, debug, and modify.

Some computer languages have been developed or modified for use with software applications for the Windows NT operating system. These include languages such as Ada, COBOL, Forth, LISP, Prolog, Visual BASIC, and Visual C++.

13.8 CAD SOFTWARE

Contemporary CAD software is often sold in "packages" that feature all of the programs needed for CAD applications. These fall into two categories: graphics software and analysis software. Graphics software makes use of the CPU and its peripheral input/output devices to generate a design and represent it on-screen. Analysis software makes use of the stored data relating to the design and applies them to dimensional modeling and various analytical methods using the computational speed of the CPU.

13.8.1 Graphics Software

Traditional drafting has consisted of the creation of two-dimensional technical drawings that operated in the synthesis stage of the general design process. However, contemporary computer graphics software, including that used in CAD systems, enables designs to be represented pictorially on the screen such that the human mind may create perspective, thus giving the illusion of three dimensions on a 2D screen. Regardless of the design representation, the drafting itself only involves taking the conceptual solution for the previously recognized and defined problem and representing it pictorially. It has been asserted above that this "electronic drawing-board" feature is one of the advantages of computer-aided design. But how does that drawing board operate?

The drawing board available through CAD systems is largely a result of the supporting graphics software. That software facilitates graphical representation of a design on-screen by converting graphical input into Cartesian coordinates along x-, y-, and sometimes z-axes. Design elements such as geometric shapes are often programmed directly into the software for simplified geometric representation. The coordinates of the lines and shapes created by the user can then be organized into a matrix and manipulated through matrix multiplication, and the resulting points, lines, and shapes are relayed back to the graphics software and, finally, the display screen for simplified editing of designs. Because the whole process can take as little as a few nanoseconds, the user sees the results almost instantaneously. Some basic graphical techniques that can be used in CAD systems include scaling, rotation, and translation. All are accomplished through an application of matrix manipulation to the image coordinates. While matrix mathematics provides the basis for the movement and manipulation of a drawing, much of CAD software is dedicated to simplifying the process of drafting itself because creating the drawing line by line, shape by shape is a lengthy and tedious process in itself. CAD systems offer users various techniques that can shorten the initial drafting time.

Geometric Definition

All CAD systems offer defined geometric elements that can be called into the drawing by the execution of a software command. The user must usually specify the variables specific to the desired element. For example, the CAD software might have, stored in the program, the mathematical definition of a circle. In the x–y coordinate plane, that definition is the following equation:

$$(x - m)^2 + (y - n)^2 = r^2$$

Here, the radius of the circle with its center at (m, n) is r. If the user specifies m, n, and r, a circle of the specified size will be represented on-screen at the given coordinates. A similar process can be applied to many other graphical elements. Once defined and stored as an equation, the variables of size and location can be applied to create the shape on-screen quickly and easily. This is not to imply that a user must input the necessary data in numerical form. Often, a graphical input device such as a mouse, trackball, digitizer, or light pen can be used to specify a point from which a line (sometimes referred to as a *rubber-band line* due to the variable length of the line as the cursor is moved toward or away from the given point) can be extended until the desired length is reached. A second input specifies that the desired endpoint has been reached, and variables can be calculated from the line itself. For a rectangle or square, the line might represent a diagonal from which the lengths of the sides could be extrapolated. In the example of the circle above, the user would specify that a circle was to be drawn using a screen command or other input method. The first point could be established on-screen as the center. Then the line extending away from the center would define the radius. Often the software will show the shape changing size as the line lengthens or shortens. When the radial line corresponds to the circle of desired size, the second point is defined. The coordinates of the two defined points give the variables needed for the program to draw the circle. The center is given by the coordinates of the first point and the radius is easily calculated by determining the length of the line between points 1 and 2. Most engineering designs are much more complex than simple, whole shapes, and CAD systems are capable of combining shapes in various ways to create the desired design.

The combination of defined geometric elements enables the designer to create many unique geometries quickly and easily on a CAD system. The concepts involved in two-dimensional combinations are illustrated before moving on to three-dimensional combinations.

Once the desired geometric elements have been called into the program, they can be defined as *cells,* individual design elements within the program. These cells can then be added as well as subtracted in any number of ways to create the desired image. For example, a rectangle might be defined as cell "A" and a circle might be defined as cell "B." When these designations have been made, the designer can add the two geometries or subtract one from the other, using Boolean logic commands such as union, intersection, and difference. The concept for two dimensions is illustrated by Fig. 13.11. The new shape can also be defined as a cell and combined in a similar manner to other primitives or conglomerate shapes. Cell definition, therefore, is recognized as a very powerful tool in CAD.

13.8.2 Solid Modeling

Three-dimensional geometric or solid modeling capabilities follow the same basic concept illustrated above, but with some other important considerations. First, there are various approaches to creating the design in three dimensions (Fig. 13.12). Second, different operators in solid-modeling software may be at work in constructing the 3D geometry.

In CAD solid-modeling software, there are various approaches that define the way in which the user creates the model. Since the introduction of solid-modeling capabilities into the CAD main-

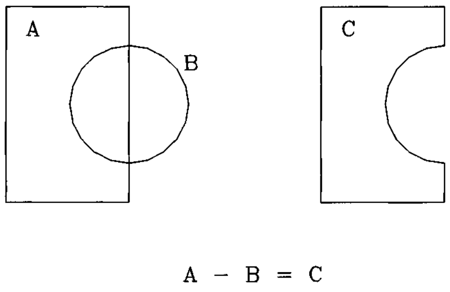

Fig. 13.11 Two-dimensional example of Boolean difference.

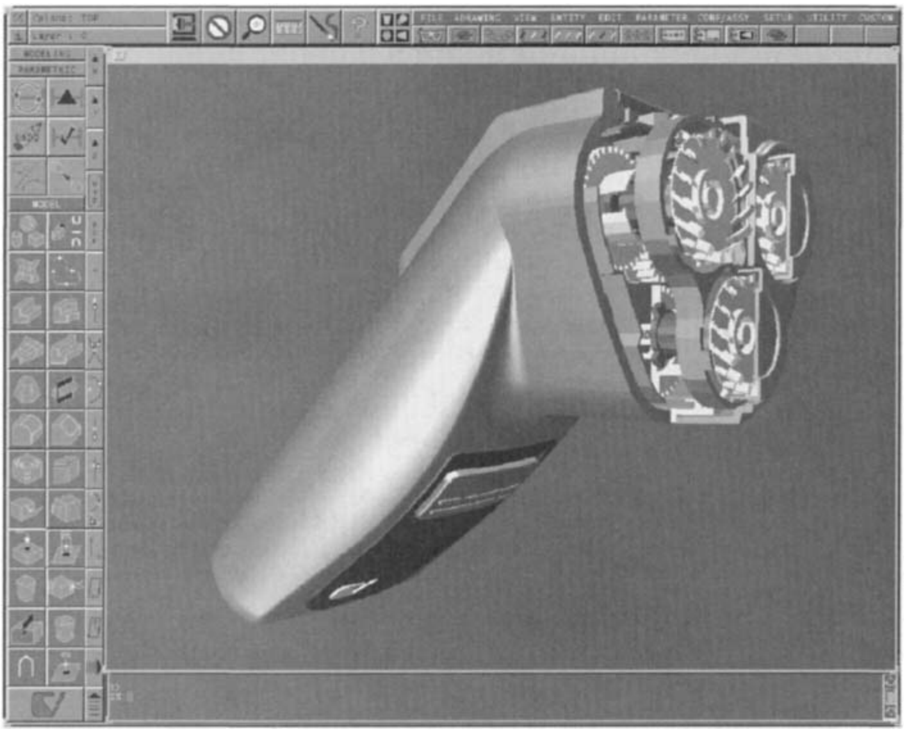

Fig. 13.12 Solid model of an electric shaver design (courtesy of ComputerVision, Inc.).

stream, various functional approaches to solid modeling have been developed. Many CAD software packages today support dimension-driven solid-modeling capabilities, which include variational design, parametric design, and feature-based modeling.

Dimension-driven design denotes a system whereby the model is defined as sets of equations that are solved sequentially. These equations allow the designer to specify constraints, such as that one plane must always be parallel to another. If the orientation of the first plane is changed, the angle of the second plane will likewise be changed to maintain the parallel relationship. This approach gets its name from the fact that the equations used often define the distances between data points.

The *variational modeling* method describes the design in terms of a sketch that can later be readily converted to a 3D mathematical model with set dimensions. If the designer changes the design, the model must then be completely recalculated. This approach is quite flexible because it takes the dimension-driven approach of handling equations sequentially and makes it nonsequential. Dimensions can then be modified in any order, making it well suited for use early in the design process when the design geometry might change dramatically. Variational modeling also saves computational time (thus increasing the run-speed of the program) by eliminating the need to solve any irrelevant equations. Variational sketching (Fig. 13.13) involves creating two-dimensional profiles of the design that can represent end views and cross sections. Using this approach, the designer typically focuses on creating the desired shape with little regard for dimensional parameters. Once the design shape has been created, a separate dimensioning capability can scale the design to the desired dimensions.

Parametric modeling solves engineering equations between sets of parameters such as size parameters and geometric parameters. Size parameters are dimensions such as the diameter and depth of a hole. Geometric parameters are constraints such as tangential, perpendicular, or concentric relationships. Parametric modeling approaches keep a record of operations performed on the design such that relationships between design elements can be inferred and incorporated into later changes in the design, thus making the change with a certain degree of acquired knowledge about the relationships between parts and design elements. For example, using the parametric approach, if a recessed area in the surface of a design should always have a blind hole in the exact center of the area, and the recessed portion of the surface is moved, the parametric modeling software will also move the blind hole to the new center. In Fig. 13.14, if a bolt circle (BC) is concentric with a bored hole and the bored hole is moved, the bolt circle will also move and remain concentric with the bored

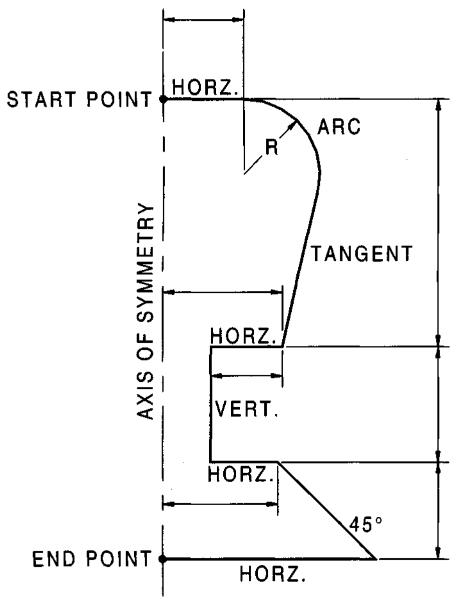

Fig. 13.13 Variational sketch.

hole. The dimensions of the parameters may also be modified using parametric modeling. The design is modified through a change in these parameters, either internally, within the program, or from an external data source, such as a separate database.

Feature-based modeling allows the designer to construct solid models from geometric features, which are industrial standard objects such as holes, slots, shells, and grooves (Fig. 13.15). For example, a hole can be defined using a "through-hole" feature. Whenever this feature is used, independent on the thickness of the material through which the hole passes, the hole will always be open at both sides. In variational modeling, by contrast, if a hole were created in a plane of specified thickness and the thickness were increased, the hole would be a blind hole until the designer adjusted the dimensions of the hole to provide an opening at both ends. The major advantage of feature-based modeling is the maintenance of design intent regardless of dimensional changes in design. Another significant advantage in using a feature-based approach is the capability to change many design elements relating to a change in a certain part. For example, if the threading of a bolt is changed, the threading of the associated nut would be changed automatically, and if that bolt design were used more than once in the design, all bolts and nuts could similarly be altered in one step. A knowledge base and inference engine make feature-based modeling more intelligent in some feature-based CAD systems.

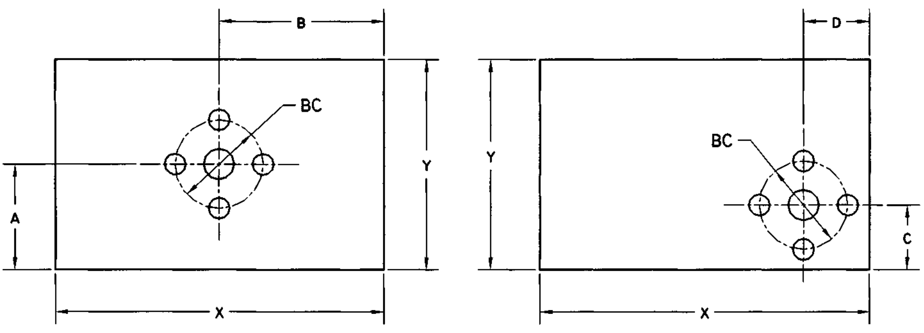

Fig. 13.14 Parametric modeling.

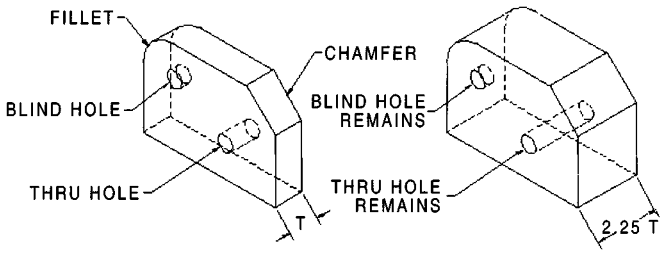

Fig. 13.15 Feature-based modeling.

Regardless of the modeling approach employed by a software package, there are usually two basic methods for creating 3D solid models: constructive solid geometry (CSG) and boundary representation (B-rep). Most CAD applications offer both methods.

With the CSG method, using defined solid geometries, such as those for a cube, sphere, cylinder, prism, and so on, the user can combine them by subsequently employing a Boolean logic operator, such as union, subtraction, difference, and intersection, to generate a more complex part. In three dimensions, the Boolean difference between a cylinder and a torus might appear as in Fig. 13.16.

The boundary representation method is a modeling feature in 3D representation. Using this technique, the designer first creates a 2D profile of the part. Then, using a linear, rotational, or compound sweep, the designer extends the profile along a line, about an axis, or along an arbitrary curved path, respectively, to define a 3D image with volume. Figure 13.17 illustrates the linear, rotational, and compound sweep methods.

Software manufacturers approach solid modeling differently. Nevertheless, every comprehensive solid modeler should have five basic functional capabilities: interactive drawing, a solid modeler, a dimensional constraint engine, a feature manager, and assembly managers.

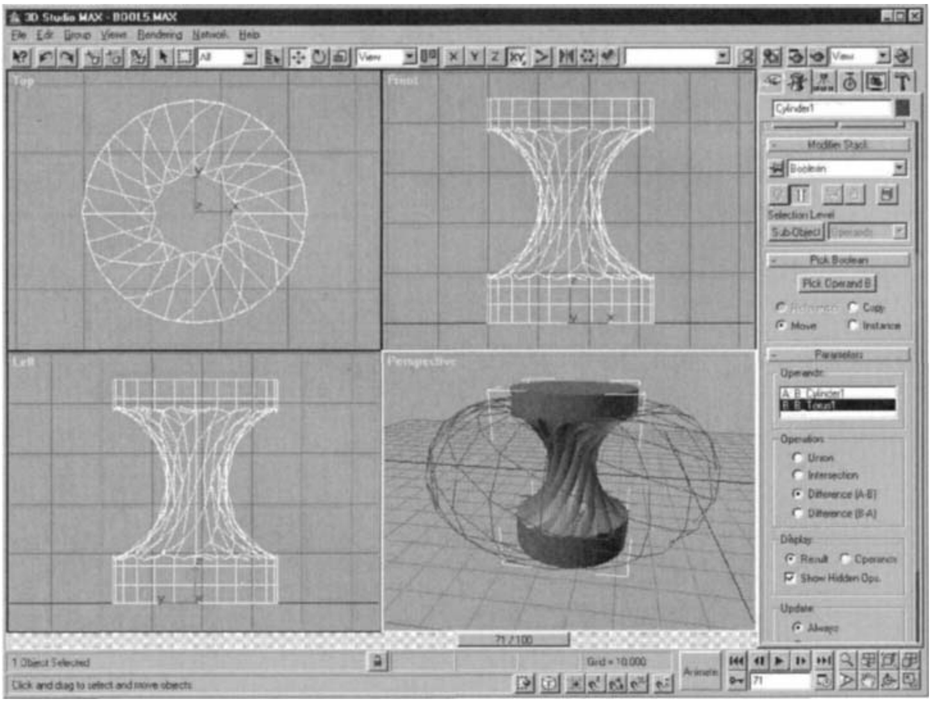

Fig. 13.16 Boolean difference between a cylinder and a torus using Autodesk 3-D StudioMax software (courtesy of Autodesk, Inc.).

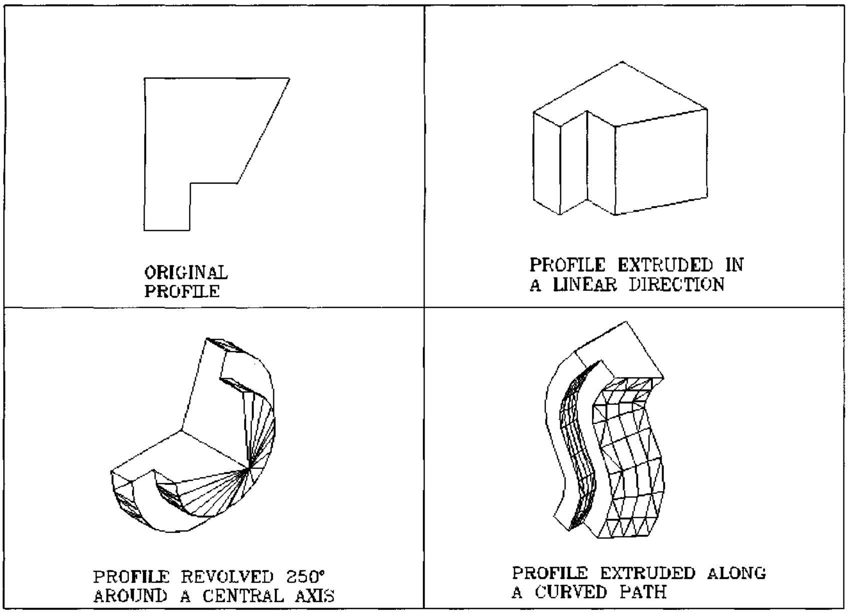
ORIGINAL PROFILE
PROFILE EXTRUDED IN A LINEAR DIRECTION
PROFILE REVOLVED 250° AROUND A CENTRAL AXIS
PROFILE EXTRUDED ALONG A CURVED PATH

Fig. 13.17 Various common sweep methods in CAD software.

The drawing capabilities should indicate shapes and profiles quickly and easily, usually with one or two mouse clicks. The purpose of drawing interactively on the screen should be to capture the basic concept information in the computer as efficiently as possible. The solid modeler should be able to combine geometric elements using Boolean logic commands and transform 2D cross sections into 3D models with volume using linear, rotational, and compound sweep methods. A dimensional constraint engine controls relational variables associated with the model such that when the model is changed, the variables change correspondingly. It is the dimensional constraint engine that allows variables to be defined in terms of their relatedness to other variables instead of as fixed geometric elements in the design file. The feature manager allows features such as holes, slots, and flanges to be introduced into the design. These features can save time in later iterations of design and represent a major advance in CAD system software in recent years. Assembly management involves the treatment of design units as conglomerate entities, often called *cells,* as a single functional unit. Assembly cells make management of the design a fairly easy task, since the user can essentially group any elements of interest into a cell, and perform selective tasks on the cell as a whole.

Another significant advance in solid modeling over the past 20 years has been the creation of parts libraries using CAD data files. In early systems, geometries had to be created from within the program. Today, many systems will accept geometries from other systems and software. The Initial Graphics Exchange System (IGES) is an ANSI standard that defines a neutral form for the exchange of information among dissimilar CAD and CAM systems. Significant time can be saved when using models from differing sources. Often, corporations will supply magnetic disks or CD-ROMs with catalogued listings of various parts and products. In this way, the engineer can focus on the major design considerations without constantly redesigning small, common parts such as bearings, bolts, cogs, sprockets, and so on.

Editing Features

CAD systems also offer the engineer powerful editing features that reduce the design time by avoiding all the manual redrawing that was traditionally required. Common editing features are performed on cells of single or conglomerate geometric shape elements. Most CAD systems offer all of the following editing functions, as well as others that might be specific to a program being used:

- *Movement.* Allows a cell to be moved to another location on the display screen
- *Duplication.* Allows a cell to appear at a second location without deleting the original location
- *Rotation.* Rotates a cell a given angle about an axis
- *Mirroring.* Displays a mirror image of the cell about a plane

- *Deletion.* Removes the cell from the display and the design data file
- *Removal.* Erases the cell from the display, but maintains it in the design data file
- *Trim.* Removes any part of the cell extending beyond a defined point, line, or plane
- *Scaling.* Enlarges or reduces the cell by a specified factor along x-, y-, and z-axes
- *Offsetting.* Creates a new object that is similar to a selected object at a specified distance
- *Chamfering.* Connects two nonparallel objects by extending or trimming them to intersect or join with a beveled line
- *Filleting.* Connects two objects with a smoothly fitted arc of a specified radius
- *Hatching.* User can edit both hatch boundaries and hatch patterns

Most of the editing features offered in CAD are transformations performed using algebraic matrix manipulations.

Transformations

Transformation in general refers to the movement or other manipulation of graphical data. Two-dimensional transformations are considered first in order to illustrate the basic concepts. Later, these concepts are applied to geometries with three dimensions.

Two-Dimensional Transformations. To locate a point in a two-axis Cartesian coordinate system, x and y values are specified. This two-dimensional point can be modeled as a 1×2 matrix: (x,y). For example, the matrix $p = (3,2)$ would be interpreted to be a point that is 3 units from the origin in the x-direction and 2 units from the origin in the y-direction.

This method of representation can be conveniently extended to define a line segment as a 2×2 matrix by giving the x and y coordinates of the two end points of the line. The notation would be

$$l = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix}$$

Using the rules of matrix algebra, a point or line (or other geometric element represented in matrix notation) can be operated on by a transformation matrix to yield a new element.

There are several common transformations: translation, scaling, and rotation.

Translation. Translation involves moving the element from one location to another. In the case of a line segment, the operation would be

$$\begin{cases} x_1' = x_1 + \Delta x & y_1' = y_1 + \Delta y \\ x_2' = x_2 + \Delta x & y_2' = y_2 + \Delta y \end{cases}$$

where x', y' are the coordinates of the translated line segment,
x, y are the coordinates of the original line segment,
Δx and Δy are the movements in the x and y directions, respectively.

In the matrix notation, this can be represented as

$$l' = l + T$$

where $T = \begin{bmatrix} \Delta x & \Delta y \\ \Delta x & \Delta y \end{bmatrix}$ is the translation matrix.

Any other geometric element can be translated in space by adding Δx to the current x value and Δy to the current y value of each point that defines the element.

Scaling. The scaling transformation enlarges or reduces the size of elements. Scaling of an element is used to enlarge it or reduce its size. The scaling need not necessarily be done equally in the x and y directions. For example, a circle could be transformed into an ellipse by using unequal x and y scaling factors.

A line segment can be scaled by the scaling matrix as follows:

$$l' = l \times S$$

where $S = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}$ is the scaling matrix.

Note that the x scaling factor α and y scaling factor β are not necessarily the same. This would produce an alteration in the size of the element by the factor α in the x-direction and by the factor β in the y-direction. It also has the effect of repositioning the element with respect to the Cartesian system origin. If the scaling factors are less than one, the size of the element is reduced and it is moved closer to the origin. If the scaling factors are larger than one, the element is enlarged and removed farther from the origin. Scaling can also occur without moving the relative position of the element with respect to the origin. In this case, the element could be translated to the origin, scaled, and translated back to the origin location.

Rotation. In this transformation, the geometric element is rotated about the origin by an angle θ. For a positive angle, the rotation is in the counterclockwise direction. This accomplishes rotation of the element by the same angle, but it also moves the element. In matrix notation, the procedure would be as follows:

$$l' = l \times R$$

where $R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$ is the rotation matrix.

Besides rotating about the origin point (0, 0), it might be important in some instances to rotate the given geometry about an arbitrary point in space. This is achieved by first moving the center of the geometry to the desired point and then rotating the object. Once the rotation is performed, the transformed geometry is translated back to its original position.

Concatenation. The previous single transformations can be combined as a sequence of transformations. This is called *concatenation,* and the combined transformations are called *concatenated transformations.*

During the editing process, when a graphic model is being developed, the use of concatenated transformations is quite common. It would be unusual that only a single transformation would be needed to accomplish a desired manipulation of the image. One example in which combinations of transformations would be required would be to uniformly scale a line l and then rotate the scaled geometry by an angle θ about the origin. The resulting new line is then

$$l' = l \times R \times S$$

where R is the rotation matrix and S is the scaling matrix. A concatenation matrix can then be defined as

$$C = RS$$

Concatenation is a unique feature used in many CAD functions in which a number of transformations are applied to a geometry. The advantage is in the amount of multiplication performed to get the desired picture. In the concatenation procedure, the transformation matrix C is first evaluated and then stored for future use. This eliminates the need of premultiplying the individual matrix to yield the desired transformed geometry.

The above concatenation matrix cannot be used in the example of rotating geometry about an arbitrary point. In this case, the sequence would be translation to the origin, rotation about the origin, then translation back to the original location. Note that the translation has to be done separately.

Three-Dimensional Transformations. Transformations by matrix methods can be extended to three-dimensional space. The same three general categories defined in the preceding section are considered.

Translation. The translation matrix for a point defined in three dimensions would be

$$T = (\Delta x, \quad \Delta y, \quad \Delta z)$$

An element would be translated by adding the increments Δx, Δy, and Δz to the respective coordinates of each of the points defining the three-dimensional geometry element.

Scaling. The scaling transformation is given by

$$S = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix}$$

Equal values of α, β, γ, produce a uniform scaling in all three directions.

Rotation. Rotation in three dimensions can be defined for each of the axes. Rotation about the z axis by an angle θ_z is accomplished by the matrix

$$R_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation about the y axis by the angle θ_y is accomplished similarly

$$R_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix}$$

Rotation about the x axis by the angle θ_x is performed with an analogous transformation matrix

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix}$$

All the three rotations about x, y, and z axes can be concatenated to form a rotation about an arbitrary axis.

Graphical Representation of Image Data

As discussed in the introduction to this section, one of the major advantages of CAD is its ability to display the design interactively on the computer display screen. Wireframe representations, whether in 2D or 3D, can be ambiguous and difficult to understand. Because mechanical and other engineering designs often involve three-dimensional parts and systems, CAD systems that offer 3D representation capabilities have quickly become the most popular in engineering design.

In order to generate a 2D view from a 3D model, the CAD software must be given information describing the viewpoint of the user. With this information, the computer can calculate angles of view and determine which surfaces of the design would be visible from the given point. The software typically uses surfaces that are closest to the viewer to block out surfaces that would be hidden from view. Then, applying this technique and working in a direction away from the viewer, the software determines which surfaces are visible. The next step determines the virtual distance between viewer and model, allowing those areas outside the boundaries of the screen to be excluded from consideration. Then the colors displayed on each surface must be determined by combining considerations of the user's preferences for light source and surface color. The simulated light source can play a very important role in realistically displaying the image by influencing the values of colors chosen for the design and by determining reflection and shadow placements (see Fig. 13.18). Current high-end CAD software can simulate a variety of light sources, including spot-lighting and sunlight, either direct or through some opening such as a door or window.

Some systems will even allow surface textures to be chosen and displayed. Once these determinations have been made, the software calculates the color and value for each pixel in the raster-scan display terminal. Since these calculations are computationally intensive, the choice of hardware is often just as important as the software used when employing solid-modeling programs with advanced surface representation features.

13.9 CAD STANDARDS AND TRANSLATORS

In order for CAD applications to run across systems from various vendors, four main formats facilitate this data exchange:

- IGES
- STEP
- DXF
- ACIS (American Committee for Interoperable Systems)

IGES (Initial Graphics Exchange Specification)

IGES is an ANSI standard for the digital representation and exchange of information between CAD/CAM systems. 2D geometry and 3D constructive solid geometry (CSG) can be translated into IGES format. New versions of IGES also support boundary representation (B-rep) solid modeling capabilities. Common translators (IGES-in and IGES-out) functions available in the IGES library

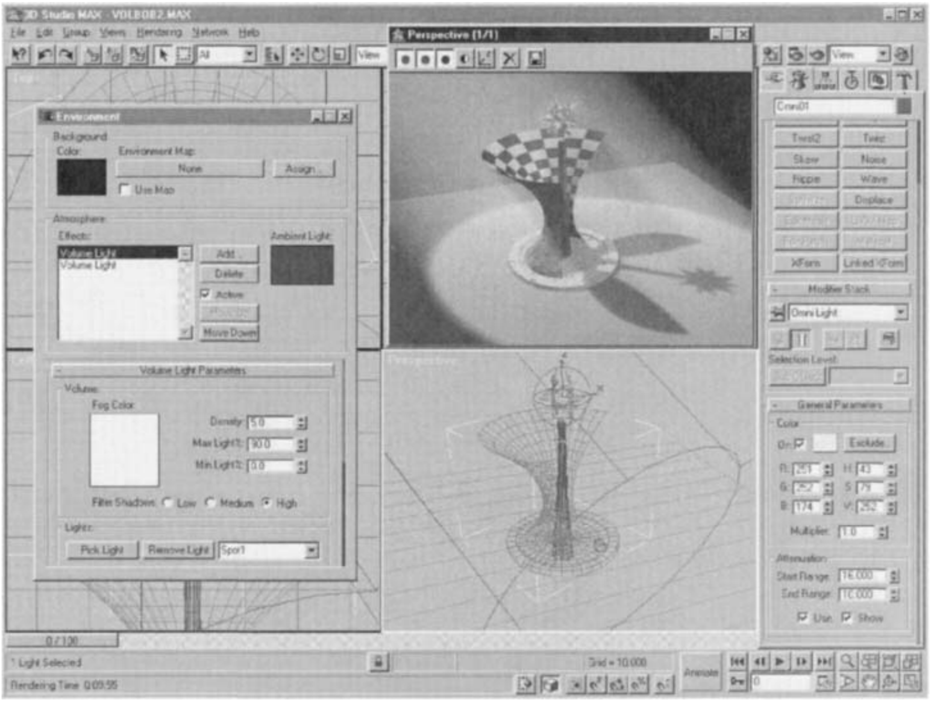

Fig. 13.18 Vase design rendered with directed spot-lighting and shadows in Autodesk 3-D StudioMax software (courtesy of Autodesk Inc.).

include IGES file parsing and formatting, general entity manipulation routines, common math utilities for matrix, vector, and other applications, and a robust set of geometry-conversion routines and linear approximation facilities.

STEP (Standard for the Exchange of Product Model Data)

STEP is an international standard. It provides one natural format that can apply to CAD data throughout the life cycle of a product. STEP offers features and benefits that are absent from IGES. STEP is a collection of standards. The user can pull out an IGES specification and get all the data required in one document. STEP can also transfer B-rep solids between CAD systems. STEP differs from IGES in how it defines data. In IGES, the user pulls out the spec, reads it, and implements what it says. In STEP, the implementor takes the definition and runs it through a special compiler that then delivers the code. This process assures that there is no ambiguous understanding of data among implementors. The conformance testing for STEP will eventually be built into the standard.

DXF (Drawing Exchange Format)

DXF, developed by Autodesk, Inc. for AutoCAD software, is the de facto standard for exchanging CAD/CAM data on a PC-based system. Only 2D drawing information can be converted into DXF, either in ASCII or in binary format.

ACIS (American Committee for Interoperable Systems)

The ACIS modeling kernel is a set of software algorithms used for creating solid-modeling packages. Software developers license ACIS routines from the developer, Spatial Technology Corp., to simplify the task of writing new solid modelers. The key benefit of this approach is that models created using software based on ACIS should run unchanged with other brands of ACIS-based modelers. This eliminates the need to use IGES translators for transferring model data back and forth among applications. ACIS-based packages have become commercially available for CAD/CAM and FEA software packages. Output files from ACIS have the suffix "*.SAT."

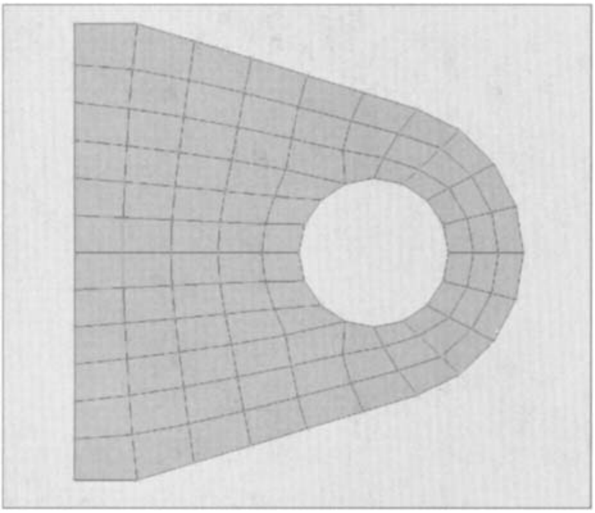

Fig. 13.19

13.9.1 Analysis Software

An important part of the design process is the simulation of the performance of a designed device. A fastener is designed to work under certain static or dynamic loads. The temperature distribution in a CPU chip may need to be calculated to determine the heat transfer behavior and possible thermal stress. Turbulent flow over a turbine blade controls cooling but may induce vibration. Whatever the device being designed, there are many possible influences on the device's performance.

The load-types listed above can be calculated using finite element analysis (FEA). The analysis divides a given domain into smaller, discrete fundamental parts called elements. An analysis of each element is then conducted using the required mathematics. Finally, the solution to the problem as a whole is determined through an aggregation of the individual solutions of the elements. In this manner, complex problems can be solved by dividing the problem into smaller and simpler problems upon which approximate solutions can be obtained. General-purpose FEA software programs have been generalized such that users do not need to have detailed knowledge of FEA.

A finite element model can be thought of as a system of solid blocks (elements) assembled together (Fig. 13.19). Several types of elements that are available in the finite-element library are

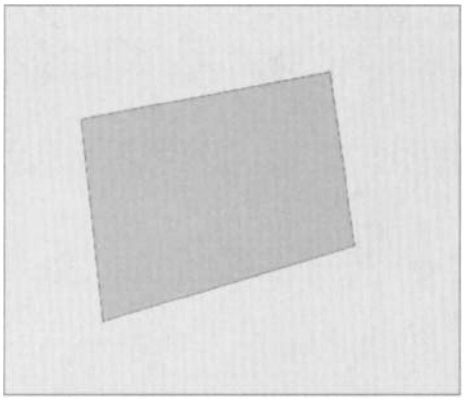

Fig. 13.20

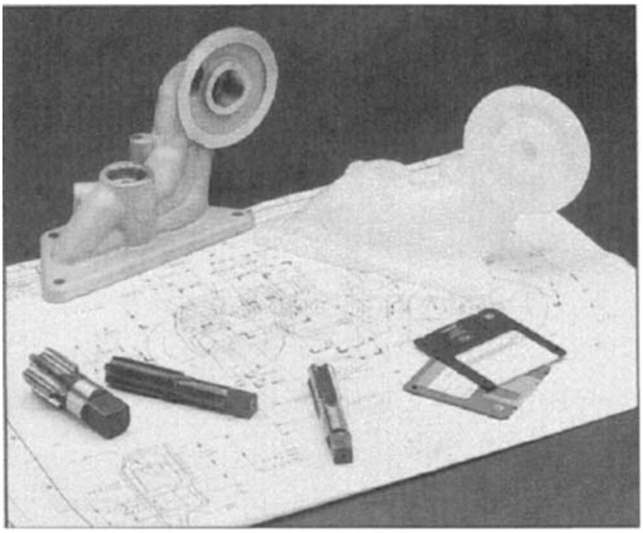

Fig. 13.21 Rapid Prototype (R) and Part cast from Prototype (L) (courtesy of ProtoCam, Inc.).

given below. Well-known general purpose FEA packages, such as NASTRAN and ANSYS, provide an element library.

To demonstrate the concept of FEA, a two-dimensional bracket is shown divided into quadrilateral elements, each having four nodes. Elements are joined to each other at nodal points. When a load is applied to the structure, all elements deform until all forces balance. For each element in the model, equations can be written that relate displacement and forces at the nodes. Each node has a potential of displacement in x and y directions under F_x and F_y (x and y components of the nodal force) so that one element needs eight equations to express its displacement. The displacements and forces are identified by a coordinate numbering system for recognition by the computer program. For example, d^I_{xi} is the displacement in the x direction for element I at node i, while d^I_{yi} is the displacement in the y direction for the same node in element I. Forces are identified in a similar manner, so that F^I_{xi} is the force in the x direction for element I at the node i.

A set of equations relating displacements and forces for the element I should take the form of the basic spring equation, $F = kd$.

$$\begin{aligned}
k_{11}d^I_{xi} + k_{12}d^I_{yi} + k_{13}d^I_{xj} + k_{14}d^I_{yj} + k_{15}d^I_{xk} + k_{16}d^I_{yk} + k_{17}d^I_{xl} + k_{18}d^I_{yl} &= F^I_{xi} \\
k_{21}d^I_{xi} + k_{22}d^I_{yi} + k_{23}d^I_{xj} + k_{24}d^I_{yj} + k_{25}d^I_{xk} + k_{26}d^I_{yk} + k_{27}d^I_{xl} + k_{28}d^I_{yl} &= F^I_{yi} \\
k_{31}d^I_{xi} + k_{32}d^I_{yi} + k_{33}d^I_{xj} + k_{34}d^I_{yj} + k_{35}d^I_{xk} + k_{36}d^I_{yk} + k_{37}d^I_{xl} + k_{38}d^I_{yl} &= F^I_{xj} \\
k_{41}d^I_{xi} + k_{42}d^I_{yi} + k_{43}d^I_{xj} + k_{44}d^I_{yj} + k_{45}d^I_{xk} + k_{46}d^I_{yk} + k_{47}d^I_{xl} + k_{48}d^I_{yl} &= F^I_{yj} \\
k_{51}d^I_{xi} + k_{52}d^I_{yi} + k_{53}d^I_{xj} + k_{54}d^I_{yj} + k_{55}d^I_{xk} + k_{56}d^I_{yk} + k_{57}d^I_{xl} + k_{58}d^I_{yl} &= F^I_{xk} \\
k_{61}d^I_{xi} + k_{62}d^I_{yi} + k_{63}d^I_{xj} + k_{64}d^I_{yj} + k_{65}d^I_{xk} + k_{66}d^I_{yk} + k_{67}d^I_{xl} + k_{68}d^I_{yl} &= F^I_{yk} \\
k_{71}d^I_{xi} + k_{72}d^I_{yi} + k_{73}d^I_{xj} + k_{74}d^I_{yj} + k_{75}d^I_{xk} + k_{76}d^I_{yk} + k_{77}d^I_{xl} + k_{78}d^I_{yl} &= F^I_{xl} \\
k_{81}d^I_{xi} + k_{82}d^I_{yi} + k_{83}d^I_{xj} + k_{84}d^I_{yj} + k_{85}d^I_{xk} + k_{86}d^I_{yk} + k_{87}d^I_{yl} + k_{88}d^I_{yi} &= F^I_{yl}
\end{aligned}$$

The k parameters are stiffness coefficients that relate the nodal deflections and forces. They are determined by the governing equations of the problem using given material properties such as Young's modulus and Poisson's ratio and from element geometry.

The set of equations can be written in matrix form for ease of operation as follows:

$$\left\{\begin{matrix}
k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} & k_{17} & k_{18} \\
k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} & k_{27} & k_{28} \\
k_{31} & k_{32} & k_{33} & k_{34} & k_{35} & k_{36} & k_{37} & k_{38} \\
k_{41} & k_{42} & k_{43} & k_{44} & k_{45} & k_{46} & k_{47} & k_{48} \\
k_{51} & k_{52} & k_{53} & k_{54} & k_{55} & k_{56} & k_{57} & k_{58} \\
k_{61} & k_{62} & k_{63} & k_{64} & k_{65} & k_{66} & k_{67} & k_{68} \\
k_{71} & k_{72} & k_{73} & k_{74} & k_{75} & k_{76} & k_{77} & k_{78} \\
k_{81} & k_{82} & k_{83} & k_{84} & k_{85} & k_{86} & k_{87} & k_{88}
\end{matrix}\right\} \times \left\{\begin{matrix}
d^I_{xi} \\ d^I_{yi} \\ d^I_{xj} \\ d^I_{yj} \\ d^I_{xk} \\ d^I_{yk} \\ d^I_{xl} \\ d^I_{yl}
\end{matrix}\right\} = \left\{\begin{matrix}
F^I_{xi} \\ F^I_{yi} \\ F^I_{xj} \\ F^I_{yj} \\ F^I_{xk} \\ F^I_{yk} \\ F^I_{xl} \\ F^I_{yl}
\end{matrix}\right\}$$

When a structure is modeled, individual sets of matrix equations are automatically generated for each element. The elements in the model share common nodes so that individual sets of matrix equations can be combined into a global set of matrix equations. This global set relates all of the nodal deflections to the nodal forces. Nodal deflections are solved simultaneously from the global matrix. When displacements for all nodes are known, the state of deformation of each element is known and stress can be determined through stress-strain relations.

For a two-dimensional structure problem, each node displacement has three degrees of freedom, one translational in each of x and y directions and a rotational in the (x,y) plane. In a three-dimensional structure problem, the displacement vector can have up to six degrees of freedom for each nodal point. Each degree of freedom at a nodal point may be unconstrained (unknown) or constrained. The nodal constraint can be given as a fixed value or a defined relation with its adjacent nodes. One or more constraints must be given prior to solving a structure problem. These constraints are referred to as *boundary conditions.*

Finite element analysis obtains stresses, temperatures, velocity potentials, and other desired unknown variables in the analyzed model by minimizing an energy function. The law of conservation of energy is a well-known principle of physics. It states that, unless atomic energy is involved, the total energy of a system must be zero. Thus, the finite element energy functional must equal zero.

The finite element method obtains the correct solution for any analyzed model by minimizing the energy functional. Thus, the obtained solution satisfies the law of conservation of energy.

The minimum of the functional is found by setting to zero the derivative of the functional with respect to the unknown nodal point potential. It is known from calculus that the minimum of any function has a slope or derivative equal to zero. Thus, the basic equation for finite element analysis is

$$\frac{dF}{dp} = 0$$

where F is the functional and p is the unknown nodal point potential to be calculated. The finite element method can be applied to many different problem types. In each case, F and p vary with the type of problem.

Problem Types

Linear Statics. Linear static analysis represents the most basic type of analysis. The term *linear* means that the stress is proportional to strain (i.e., the materials follow Hooke's law). The term *static* implies that forces do not vary with time, or that time variation is insignificant and can therefore be safely ignored.

Assuming the stress is within the linear stress-strain range, a beam under constant load can be analyzed as a linear static problem. Another example of a linear statics is a steady-state temperature distribution within a constant material property structure. The temperature differences cause thermal expansion, which in turn induces thermal stress.

Buckling. In linear static analysis, a structure is assumed to be in a state of stable equilibrium. As the applied load is removed, the structure is assumed to return to its original, undeformed position. Under certain combinations of loadings, however, the structure continues to deform without an increase in the magnitude of loading. In this case, the structure has buckled or become unstable. For elastic, or linear, buckling analysis, it is assumed that there is no yielding of the structure and that the direction of applied forces does not change.

Elastic buckling incorporates the effect of differential stiffness, which includes higher-order strain displacement relationships, that are functions of the geometry, element type, and applied loads. From a physical standpoint, the differential stiffness represents a linear approximation of softening (reducing) the stiffness matrix for a compressive axial load and stiffening (increasing) the stiffness matrix for a tensile axial load.

In buckling analysis, eigenvalues are solved. These are scaling factors used to multiply the applied load in order to produce the critical buckling load. In general, only the lowest buckling load is of interest, since the structure will fail before reaching any of the higher-order buckling loads. Therefore, usually only the lowest eigenvalue needs to be computed.

Normal Modes. Normal modes analysis computes the natural frequencies and mode shapes of a structure. Natural frequencies are the frequencies at which a structure will tend to vibrate if subjected to a disturbance. For example, the strings of a piano are each tuned to vibrate at a specific frequency. The deformed shape at a specific natural frequency is called the *mode shape.* Normal modes analysis is also called *real eigenvalue analysis.*

Normal modes analysis forms the foundation for a thorough understanding of the dynamic characteristics of the structure. In static analysis, the displacements are the true physical displacements due to the applied loads. In normal modes analysis, because there is no applied load, the mode shape components can all be scaled by an arbitrary factor for each mode.

Nonlinear Statics. Nonlinear structural analysis must be considered if large displacements occur with linear materials (geometric nonlinearity), or if structural materials behave in a nonlinear stress-strain relationship (material nonlinearity), or a combination of large displacements and nonlinear stress-strain effects occurs. An example of geometric nonlinear statics is shown when a structure is loaded above its yield point. The structure will then tend to be less stiff, permanent deformation will occur, and Hooke's law will not be applicable anymore. In material nonlinear analysis, the material stiffness matrix will change during the computation. Another example of nonlinear analysis includes the contacting problem, where a gap may appear and/or sliding may occur between mating components during load application or removal.

Dynamic Response. Dynamic response in general consists of frequency response and transient response. Frequency response analysis computes structural response to steady-state oscillatory excitation. Examples of oscillatory excitation include rotating machinery, unbalanced tires, and helicopter blades. In frequency response, excitation is explicitly defined in the frequency domain. All of the applied forces are known at each forcing frequency. Forces can be in the form of applied forces and/or enforced motions. The most common engineering problem is to apply steady-state sinusoidally varying loads at several points on a structure and determine its response over a frequency range of interest. Transient response analysis is the most general method for computing forced dynamic response. The purpose of a transient response analysis is to compute the behavior of a structure subjected to time-varying excitation. All of the forces applied to the structure are known at each instant in time. The important results obtained from a transient analysis are typically displacements, velocities, and accelerations of the structure, as well as the corresponding stresses.

13.10 APPLICATIONS OF CAD

Computer-aided design has been presented in terms of its applicability to design, the hardware and software used, and its capabilities as an entity unto itself. The use of CAD data in conjunction with specialized applications is now reviewed. These applications fall outside the realm of CAD software in a strict sense; however, they provide opportunities for the designer to use the data generated through CAD in new and innovative techniques that can similarly affect design efficiency. Many of the items discussed in this section apply to the evaluative stage of design. Some of the basic analytical methods that can be used in CAD to optimize designs have already been presented. Options open to the design engineer using information from a CAD database and special applications are now presented.

13.10.1 Optimization Applications

As designs become more complex, engineers need fast, reliable tools. Over the last 20 years, finite element analysis has become the major tool used to identify and solve design problems. Increased design efficiency provided by CAD has been augmented by the application of finite element methods to analysis, but engineers still often use a trial-and-error method for correcting the problems identified through FEA. This method inevitably increases the time and effort associated with design because it increases the time needed for interaction with the computer. As well, solution possibilities are often limited by the designer's personal experiences.

Design optimization seeks to eliminate much of this extra time by applying a logical mathematical method to facilitate modification of complex designs. Optimization strives to minimize or maximize a characteristic, such as weight or physical size, that is subjected to constraints on one or more parameters. Either the size, shape, or both determines the approach used to optimize a design. Optimizing the size is usually easier than optimizing the shape of a design. Optimizing the thickness of a plate does not significantly change its geometry. On the other hand, optimizing a design parameter, such as the radius of a hole, does change the geometry during shape optimization.

Optimization approaches were difficult to implement in the engineering environment because the process was somewhat academic in nature and not viewed as easily applicable to design practices. However, if viewed as a part of the process itself, optimization techniques can be readily understood and implemented in the design process. Iterations of the design procedure occur as they normally do in design up to a point. At that point, the designer implements the optimization program. The objectives and constraints upon the optimization must first be defined. The optimization program then evaluates the design with respect to the objectives and constraints and makes automated adjustments in the design. Because the process is automatic, engineers should have the ability to monitor the progress of the design during optimization, stop the program if necessary, and begin again.

The power of optimization programs is largely a function of the capabilities of the design software used in earlier stages. Two- and three-dimensional applications require automatic and parametric meshing capabilities. Linear static, natural frequencies, mode shapes, linearized buckling, and steady-state analyses are required for other applications. Because the design geometry and mesh can change during optimization iterations, error estimate and adaptive control must be included in the optimization program. Also, when separate parts are to be assembled and analyzed as a whole, it is often helpful to the program to connect different meshes and element types without regard to nodal or elemental interface matches.

Preliminary design data used to meet the desired design goals through evaluation, remeshing, and revision. Acceptable tolerances must then be entered along with imposed constraints on the optimization. The engineer should be able to choose from a large selection of design objectives and behavior constraints and use these with ease. Also, constraints from a variety of analytical procedures should be supported so that optimization routines can use the data from previously performed analyses.

Although designers usually find optimization of shape more difficult to perform than of size, the use of parametric modeling capabilities in some CAD software minimizes this difficulty. Shape optimization is an important tool in many industries, including shipbuilding, aerospace, and automotive manufacturing. The shape of a model can be designed using any number of parameters, but as few as possible should be used, for the sake of simplicity. If the designer cannot define the parameters, neither design nor optimization can take place. Often, the designer will hold a mental note on the significance of each parameter. Therefore, designer input is crucial during an optimization run.

13.10.2 Virtual Prototyping

The creation of physical models for evaluation can often be time-consuming and provide limited productivity. By employing kinematic and dynamic analyses on a design within the computer environment, time is saved and often the result of the analysis is more useful than experimental results from physical prototypes. Physical prototyping often requires a great deal of manual work, not only to create the parts of the model, but to assemble them and apply the instrumentation needed as well. Virtual prototyping uses kinematic and dynamic analytical methods to perform many of the same tests on a design model. The inherent advantage of virtual prototyping is that it allows the engineer to fine-tune the design before a physical prototype is created. When the prototype is eventually fabricated, the designer is likely to have better information with which to create and test the model.

Physical models can provide the engineer with valuable design data, but the time required to create a physical prototype is long and must be repeated often through iterations of the process. A second disadvantage is that through repeated iterations, the design is usually changed, so that time is lost in the process when parts are reconstructed as a working model. Too often, the time invested in prototype construction and testing reveals less useful data than expected.

Virtual prototyping of a design is one possible solution to the problems of physical prototyping. Virtual prototyping employs computer-based testing so that progressive design changes can be incorporated quickly and efficiently into the prototype model. Also, with virtual prototyping, tests can be performed on the system or its parts in a way that might not be possible in a laboratory setting. For example, the instrumentation required to test the performance of a small part in a system might disrupt the system itself, thus denying the engineer the accurate information needed to optimize the design. Virtual prototyping can also apply forces to the design that would be impossible to apply in the laboratory. For example, if a satellite is to be constructed, the design should be exposed to zero gravity in order to simulate its performance properly.

Prototyping and testing capabilities have been enhanced by rapid prototyping systems with the ability to convert CAD data quickly into solid full-scale models that can be examined and tested. The major advantage of rapid prototyping is in the ability of the design to be seen and felt by the designer and less technically adept personnel, especially when esthetic considerations must come into play. While rapid prototyping will be discussed further below, even this technology is somewhat limited in testing operations. For example, in systems with moving parts, joining rapid prototype models can be difficult and time-consuming. With a virtual prototyping system, connections between parts can be made with one or two simple inputs. Since the goal is to provide as much data in as little time as possible, use of virtual prototyping before a prototype is fabricated can strongly benefit the design project.

Engineers increasingly perform kinematic and dynamic analyses on a virtual prototype because a well-designed simulation leads to information that can be used to modify design parameters and characteristics that might not have otherwise been considered. Kinematic and dynamic analysis methods apply the laws of physics to a computerized model in order to analyze the motion of parts within the system and evaluate the overall interaction and performance of the system as a whole. At one time a mainframe computer was required to perform the necessary calculations to provide a realistic motion simulation. Today, microcomputers have the computational speed and memory capabilities necessary to perform such simulations on the desktop.

One advantage of kinematic/dynamic analysis software is that it allows the engineer to overload forces on the model deliberately. Because the model can be reconstructed in an instant, the engineer can take advantage of the destructive testing data. Physical prototypes would have to be fabricated and reconstructed every time the test was repeated. There are many situations in which physical prototypes must be constructed, but those situations can often be made more efficient and informative by the application of virtual prototyping analyses.

13.10.3 Rapid Prototyping

One of the most recent applications of CAD technology has been in the area of rapid prototyping. Physical models traditionally have the characteristic of being one of the best evaluative tools for influencing the design process. Unfortunately, they have also represented the most time-consuming and costly stage of the design process. Rapid prototyping addresses this problem, combining CAD data with sintering, layering, or deposition techniques to create a solid physical model of the design or part. The rapid prototyping industry is currently developing technology to enable the small-scale production of real parts, as well as molds and dies that can then be used in subsequent traditional manufacturing methods. These two goals are causing the industry to become specialized into two major sectors. The first sector aims to create small rapid prototyping machines that one day might become as common in the design office as printers and plotters are today. The second branch of the rapid prototyping industry is specializing in the production of highly accurate, structurally sound parts to be used in the manufacturing process.

Stereolithography

A stereolithography machine divides a 3D CAD model into slices as thin as 0.0025 in. and sends the information to an ultraviolet laser beam. The laser traces the slice onto a container of photocurable liquid polymer, crosslinking (solidifying) the polymer into a layer of resin. The first layer is then lowered by the height of the next slice. The process is then repeated and, with each repetition, the solid resin is lowered by an increment equal to the height of the next slice until the prototype has been completed. The workspace on one large stereolithography machine has a workspace of $20 \times 20 \times 20$ in.

Early stereolithographed parts were made of acrylate resins using the *Tri-Hatch* build style. This resulted in fabricated parts being very brittle, with rough surface finishes, and significantly less accuracy than provided by today's stereolithography machines. Since 1990, advances in hardware, software, polymers, and processing methods have resulted in improved accuracy. The standard measurement for accuracy used in stereolithography applications is the $\varepsilon(90)$ value, which indicates the degree of 90th percentile error. Early machines were capable of an $\varepsilon(90)$ accuracy of about 400 microns. In 1990, a new technique called the *Weave* build style increased the $\varepsilon(90)$ accuracy to 300 microns. The increased accuracy led the application of the models for verifying designs in checking for interference, tolerance build-ups, and other potential design flaws. The next advance in stereolithography at 3-D Systems was the release of the "Star-Weave" structure, when $\varepsilon(90)$ reached 200 microns. At this accuracy, engineers could begin to use stereolithographed parts in iterations of the design process and for optimization routines. There are some applications for which computational simulations are simply not accurate enough; for example, airflow through an inlet manifold on an automobile engine. Chrysler Corporation, for example, has used stereolithography to create manifold parts and subsequently tested them in the laboratory to identify the optimal design.

All of the benefits realized from the fabrication of plastic parts could potentially be greater with the ability to use the technology to create metal parts. The initial focus at 3-D Systems was on using the stereolithographed part in an investment casting system. The problem was that early stereolithographs were solid, causing thermal expansion to place stress on the ceramic shell, breaking the mold. The problem was addressed with the QuickCast build geometry. The structure of the stereolithograph is not solid, but about two-thirds hollow, with an open lattice structure. The internal lattice provides the structural strength and the somewhat hollow properties of the model generate a smooth surface definition. The key to investment casting using this approach is to ensure that any liquid polymer left within the lattice structure of the pattern has an escape route to avoid its solidification to a thickness that would cause enough thermal expansion to break the ceramic mold. Therefore, a drain hole is usually provided to allow whatever resin is left in the mold to escape before the final UV-curing process. A vent hole prevents a complete vacuum in the internal lattice.

A further advance in stereolithography technology was developed by Thomas Pang, an organic chemist at 3-D Systems, who developed an epoxy resin with significant advantages over the acrylate plastics originally used. The acrylates have high viscosity but not very high green strength. The lattice triangles in acrylates could not be too large, or they would sag, and the high viscosity of the liquid meant that the resin flowed very slowly through the drain holes in the part. Epoxy resin offers one-tenth the viscosity of acrylates coupled with a fourfold increase in green strength. Also, the linear shrinkage effective upon hardening is decreased with the epoxy technology. Acrylates show 0.6–1.0% shrinkage, while epoxy resin offers only 0.06% linear shrinkage. With the epoxy resin, $\varepsilon(90)$ values began to approach 100μm. For these reasons, epoxy resin is viewed as a major improvement over acrylate systems.

The QuickCast system, using first acrylate plastics and then epoxy resin, has opened up the market to rapid manufacturing of accurate functional prototypes in aluminum, stainless steel, carbon steel, and others. The accuracy of the prototypes keeps growing greater, as shown by an $\varepsilon(90)$ value of

91μm achieved in December 1993 with the SLA-500/30 machine. Current research focus is on using RP technology to create investment-cast steel tool molds for low-level manufacturing purposes. Ford Motor Co. implemented such technology in the production of a wiper-blade cover for its mid-year 1994 Ford Explorer. Other RP companies have also jumped into the rapid tooling market. DTM Corp. of Austin, Texas, has introduced a laser sintering process capable of sintering precursor powders into nylon, polycarbonate, and casting wax parts. The biggest advantage at DTM is the use of laser sintering to create metal prototypes. In the process, called *RapidTool* at DTM, the powder used in their Sinterstation machines is a combination low-carbon steel and thermoplastic binder. The laser uses the data from a CAD file to trace the incremental slices of a part onto the powder, causing the plastic to bind with the metal, holding the shape of the part. A low-temperature furnace burns off the plastic binder and then the temperature is raised to lightly fuse the steel particles, leaving an internal steel skeleton. The steel skeleton is subsequently infused with copper to provide a composite metal part slightly more than 50% iron by weight. The tools created using RapidTool technology use similar processes as those created using aluminum tooling. Accuracy using RapidTool is projected to reach 0.003 in. on features and 0.010 in. on any dimension.

Other materials for use with RP are currently being tested and implemented by various other companies involved in the growing rapid-prototyping industry. The technology presents not only the opportunity, but the realistic opportunity, for further automatization in the design and implementation environments. One major factor impeding the implementation of such technology on a large scale is cost. Currently, even desktop systems, such as those from 3-D Systems, range from about $100,000 to $450,000, thus limiting their use in small and mid-size corporations. The cost associated with materials is also high, meaning that virtual prototyping, at least for the time being, can cut the cost of the rapid prototyping and testing process before investing in the fabrication of even a rapid prototype. As with most technologies, however, prices are expected to drop with time, and when they do, it is expected that RP technology will become as much of a fixture in the design engineering environment as CAD itself has become.

13.10.4 Computer-Aided Manufacturing (CAM)

Although this chapter is primarily about CAD, we would be terribly remiss not to mention CAM in terms of our applications discussion. The two techniques are so integrally related in today's manufacturing environment that often, one is not mentioned without the other. Acronyms such as CAD/CAM, CADAM (computer-aided design and manufacturing), and CIM (computer-integrated manufacturing) are often used to describe the marriage between the two. In essence, CAM uses data prepared through CAD to streamline the manufacturing process through the use of tools such as computer numerical control and robotics. CAM will be discussed in much further detail in another chapter of this handbook. We strongly refer the reader to that chapter to foster some basic understanding of this related subject.

BIBLIOGRAPHY

Ali, H., "Optimization for Finite Element Applications," *Mechanical Engineering,* 68–70 (December 1994).

Amirouche, F. M. L., *Computer-Aided Design and Manufacturing,* Prentice-Hall, Englewood Cliffs, NJ, 1993.

Ashley, S., "Prototyping with Advanced Tools," *Mechanical Engineering,* 48–55 (June 1994).

"Basics of Design Engineering," *Machine Design,* 47–83 (February 8, 1996).

"Basics of Design Engineering," *Machine Design,* 83–126 (July 1995).

"CAD/CAM Industry Report 1994," *Machine Design,* 36–98 (May 23, 1994).

Deitz, D., "PowerPC: The New Chip on the Block," *Mechanical Engineering,* 58–62 (January 1996).

Dvorak, P. (ed), "Engineering on the Other Personal Computer," *Machine Design,* 42–52 (October 26, 1995).

———, "Windows NT Makes CAD Hum," *Machine Design,* 46–52 (January 10, 1994).

"Engineering Drives Document Management," Special Editorial Supplement, *Machine Design,* 77–84 (June 15, 1995).

Foley, J. D., et al., *Computer Graphics: Principles and Practice,* 2nd ed., Addison-Wesley, New York, 1990.

Groover, M. P., and E. W. Zimmers, Jr., *CAD/CAM: Computer-Aided Design and Manufacturing,* Prentice-Hall, Englewood Cliffs, NJ, 1984.

Hanratty, P. J., "Making Solid Modeling Easier to Use," *Mechanical Engineering,* 112–114 (March 1994).

Hodson, W. K. (ed), *Maynard's Industrial Engineering Handbook,* 4th ed., McGraw-Hill, New York, 1992.

Hordeski, M. F., *CAD/CAM Techniques,* Reston, Reston, VA, 1986.

Krouse, J. K., *What Every Engineer Should Know About Computer-Aided Design and Computer-Aided Manufacturing,* Marcel Dekker, New York, 1982.

Lee, G., "Virtual Prototyping on Personal Computers," *Mechanical Engineering,* 70–73 (July 1995).

Masson, R., "Parallel and Almost Personal," *Machine Design,* 70–76 (April 20, 1995).

Microsoft Windows NT from a UNIX Point of View, Business Systems Technology Series, Microsoft Corp.

Norton, R. L., Jr., "Push Information, Not Paper," *Machine Design,* 105–109 (December 12, 1994).

Puttre, M., "Taking Control of the Desktop," *Mechanical Engineering,* 62–66 (September 1994).

Shigley, J. E., and C. R. Mischke, *Mechanical Engineering Design,* 5th ed., McGraw-Hill, New York, 1989.

Stallings, W., *Computer Organization and Architecture: Designing for Performance,* 4th ed., Prentice-Hall, Englewood Cliffs, NJ, 1996.

Teschler, L. (ed.), "Why PDM Projects Go Astray," *Machine Design,* 78–82 (February 22, 1996).

Wallach, S., and J. Swanson, "Higher Productivity with Scalable Parallel Processing," *Mechanical Engineering,* 72–74 (December 1994).