Mapped out is the field of interactive computer graphics technology. The author surveys the range of applications from the visual arts to the visualization of theoretical mathematical models to the simulation of aircraft and ship navigation. Hardware and software are explored. Also outlined are interactive graphics data bases, data structures, and proposed standards that apply to them.

Interactive graphics today

by R. S. Burchi

Computer-oriented interactive graphics represents the combination of a computer and a graphics output (and input) device as a medium by which a user manipulates visual information. The end result may be the design of an automobile, the teaching of a lesson in electromagnetic field theory, the training of an airplane pilot, entertainment such as animated cartoons, or the manipulation of colors, masses, and forms to produce purely artistic designs.

This journal last had an issue dedicated to graphics[1] in 1968. In the meantime, a number of its covers have been artistic designs produced by the interaction of the designer with the medium.[2] Since 1968, the use of graphics devices has expanded into almost all industries. Three changes that have fostered this increased usage are the following: 1. the availability of a wide variety of displays of all prices and capabilities, including home microprocessor-based cathode ray tube (CRT) systems; 2. the availability of turn-key or complete end-user application packages; and 3. an abundance of documentation in the literature about successful applications, payoffs, and implementation techniques. New users of graphics systems are no longer the "pioneers." This paper discusses a variety of applications and implementation techniques.

Range of graphics applications

Graphics display devices are used almost everywhere that computers are used. It is useful, however, to differentiate among their applications and among the advantages they provide. Carl Machover[3] breaks down computer graphics applications into the following six general areas:

- Management information.
- Scientific graphics.
- Command and control.
- Image processsing.
- Real-time image generation.
- Electrical and mechanical design.

He points out that each of these application areas can require markedly different hardware and software.

Another way to look at graphics applications is in terms of the visualization functions or services provided by computer graphics devices, some of which are the following:

- Graphics display of computation results as they occur.
- Replacement of paper as a drafting medium.
- Rapid presentation of large quantities of information.
- Observing and influencing change.
- Visualization of nonexistent or unbuilt objects for the rapid study of design options.
- Enhancement of interpretation or impact of data.
- Visual communication to replace alphanumeric man-machine communication.
- Process simulation and verification before committing real resources.
- Simulation of real-world scenes.
- Graphic representation and observation of simulated theoretical models.
- Creation of artistic designs.
- Entertainment.

Most graphics applications that have been implemented exploit one or more of these functions or services, which are now expanded with examples.

display of computed data

Results of computations can be displayed graphically as they occur and before all computations are complete. This provides real-time feedback to a user who may be doing a parameter study for a new design such as an airplane wing. He can terminate a particular computation if the partial results show that different parameters are required. Such a dynamic technique is common with typewriter-like terminals, but much more immediately useful in-

formation can be provided graphically. Users report significant savings in main computer time when computations can be aborted early and when several options can be explored in the time required to make one hand drawing.

drafting medium

The replacement of paper as a drafting medium is perhaps one of the most commonly exploited advantages of a graphics system. A graphics system that presents the proper tools at the screen for image creation and manipulation can make a user much more productive. Among the many documented studies of increased productivity, Reference 4 cites productivity time ratios ranging from 1.9 to 17 times improvement where engineering drawings and/or machine parts are the end-products. Even though a graphics console may be used as a tool to create images, the end-product may still be a drawing on paper for wide communication of the same viewed information to those without graphics consoles. Shepherd[5(a)] describes a system of text and artwork composition that normally would be accomplished with paper and pencil. Pooch[5(b)] points out further reasons why the graphics console is faster than paper and pencil in an engineering drawing environment: repetitive construction entities do not have to be redrawn, but are instantaneously called forth from storage; any symmetry occurring on the drawing is immediately displayed rather than having to be redrawn, and analytic geometry constructions are performed by the computer and do not have to be calculated and drawn manually. Also, design accuracy is often better than can be achieved manually.

Certain laborious processes become trivial to the user with a graphics system. Given the proper data base design, a new view of an object requires only that the user state the request in precise form, such as rotation about one, two, or three axes. Also, rotation or translation of one object relative to another in drawing space or in a real-world coordinate system is also easily performed. The system design and data base design for such manipulations are complex, but once implemented, with ease of use in mind, interactive computer graphics becomes a versatile tool that can increase a user's productivity by an order of magnitude.

information display efficiency

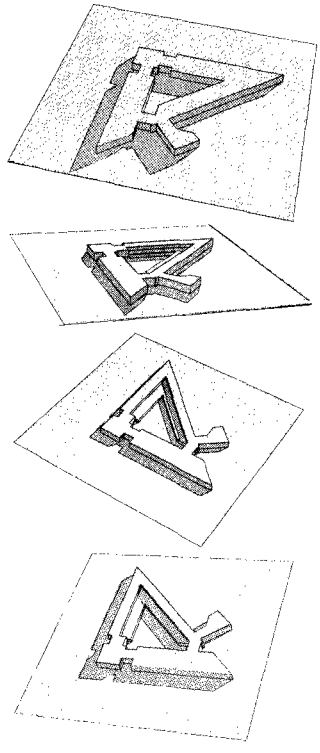
Large amounts of information can be presented in a short space of time. Many graphics applications are based on the fact that a graphics console can display a large quantity of information at one time and, for a regenerated cathode ray tube, the console can display changes every display cycle. (Regenerated CRT displays usually cycle at 40 to 60 times per second to avoid flicker.)

Resolution of displays varies. For example, Handelman[6] discusses high-resolution graphics, and Williams[7] describes a color system of 512 by 512 picture elements (pels). Each pel can be any one of 128 colors at one time out of a total of 16^3 (or 4096) possible

hue/intensity choices. The author points out that higher resolution and more colors would be possible with a larger refresh memory.

Highest-resolution CRTs tend to be monochromatic, but 1000 lines are available in color.[3] Tannas and Goede[8] describe plasma displays of 512 by 512 addressable-point resolution, and the PLATO[9] system uses a 512 by 512 addressable-point plasma panel with optional rear-projected color photographs. Commercially available monochromatic displays[10(a)] are available with up to 4096 by 4096 addressable points.

Figure 1 A study of an unbuilt building with seasonal change of light from three points of view



Applications that exploit the high information content of a display abound. Mallary and Ferraro[10(b)] describe ECOSITE, by which complex views of land sites are displayed and manipulated to study land use and land reclamation. McCleary[11] describes the display of ocean data where color aids in the differentiation of data, such as depth and shipping density.

One can create change or the illusion of change at a graphics console, as exemplified by the work done on the creation of animated cartoons. Levoy[12] describes a system that employs multiple planes of information, where each plane can contain a portion of an overall cartoon. An automatic interpolation technique is used to provide for the illusion of continuous motion. Catmull[13] points out the great problems of automatic interpolation, or "in-betweening," due primarily to the insufficiency of information available to translate a 2-D view of a cartoon character to another 2-D view without a 3-D model (data) of the character.

Other monitoring applications where changing information is presented include the status information of oil refineries, power distribution systems, and rocket launchings. Some air traffic control systems combine both radar inputs and digitized information about specific flights on the same screen.

visualizing unbuilt objects

A graphics representation of an unbuilt object can provide insight into potential problems before construction. As an example, Gonin and Moffett[14] describe a graphics program for designing highways. A major innovation is that the designers can "drive" down the highway while sitting before the display screen, and look for such potential hazards as dangerously poor visibility. Another example is the viewing of an unbuilt building from various viewpoints, under changing seasonal lighting conditions and architectural modifications as shown in Figure 1.

visual enhancement

Many techniques have been created to emphasize or highlight certain characteristics of data. Reference 15 describes many of these techniques, including the use of histograms, empirical density functions, pie charts, contour plots, discriminant analysis

displays, cluster analyses, Chernoff "faces," and Andrews' sine curves. A number of cartographic techniques are also described. Reference 16 describes a system for studying crime data and for presenting it in various forms of maps for greater comprehension. The use of a color graphics system as a programming support tool is discussed in Reference 17. The authors state that the programmer works with a two-dimensional graphics representation of the program (instead of a linear text string) that exhibits the meaning of a program more clearly and results in better coding and improved programmer productivity.

pictorial communication

Pictorial communication can replace alphanumeric man-machine communication. Many uses of computers before the advent of interactive graphics employed alphanumeric input and batched plotted output. In this older graphics technology, for example, the output for numerically controlled drawing machines and machine tools would be directed by input statements created by parts programmers. They would describe the geometry of parts to be produced using special languages. Today, the use of graphics consoles allows the designer to construct the parts pictorially, thus eliminating the need for geometric languages. That is, the end-user works entirely with geometric construction tools at the screen in a manner similar to working at a drafting table, except that he can be much more productive.

process simulation

Processes can be simulated and observed for verification before committing real resources. Directions to be given to a numerically controlled machine tool can be graphically and visually verified at the screen of a graphics console before driving the real machine. For example, some systems can picture a "tool" traversing its programmed path with continuous motion, providing verification that can avoid costly errors on the machine floor.

real-world simulation

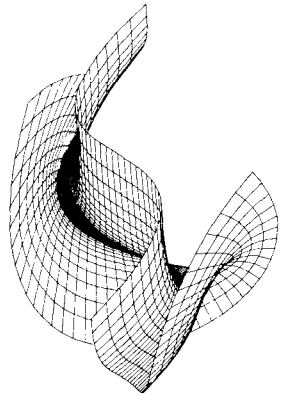
The simulation of real-world scenes is perhaps the area of greatest challenge. Computational and storage capacity to produce photographic quality is justified and is required in areas such as aircraft and ship training simulators. Elaborate airplane simulators have been in existence for many years, and have offered the advantages of training pilots at less cost and less danger than using real airplanes. Various schemes have been employed to present to a pilot trainee a realistic view of the terrain he is "flying" over. A computer-controlled television camera moving in real time over a miniature landscape is one example of a method of simulating a real world in which unprogrammed events can take place. Contrast this with a prerecorded video program through which—even when combined with other visual materials—only a limited range of unanticipated events can be depicted.

As a further improvement, work has been going on to create images in real time by means of a digital computer receiving input

from the controls in the hands of the pilot. Weinberg[18] describes a system that has been created to show the pilot of the Space Shuttle his earthbound approach, with views through simulated windows of the earth from space. In addition, Shuttle maneuvers in space together with other spatial objects can be depicted. Weinberg speculates that students could be taught to drive automobiles in similar systems once the cost of the simulators has been sufficiently reduced. Other systems have been created to display harbors for ship's captain trainees at the controls of ship simulators.[19]

Simulated theoretical models may be graphically represented and observed. One particularly good set of examples of such visualization is in education, where graphics systems are used for the teaching of complex subjects and concepts. Here theoretical models in science, engineering, and other disciplines are represented visually by pictures at a graphic console. Figure 2 shows a high-order algebraic surface of the class of surfaces that are also known as "catastrophe surfaces." Such surfaces can represent multiparameter functions (such as one might find in economics or biology) in which sudden changes of state occur. Smith and Sherwood[9] describe uses of the PLATO educational system in such an environment.

Figure 2 A high-order algebraic surface



Huggins[20] presents a fascinating discussion of visual and symbolic thinking and shows many of the problems of presenting concepts pictorially rather than verbally. He points out "how dependent we are on words—that one word is worth a thousand pictures" but that "good graphics probably ought to involve an effort to avoid the words and labels to begin with." And Sagan[21] points out that computer graphics permits important and novel kinds of learning experiences in arts and sciences.

graphics design

The graphics console can be an inventive tool for artists and designers. Csuri[22] provides a survey of computer art, and Musgrave[2] describes a color system that is used for, among other things, creative design. The nonprogrammer user of the system described by Musgrave can experiment with many different patterns and color combinations. Reference 2 also provides a brief synopsis of related current work in the field of graphics consoles for graphic design.

entertainment

Wide use is made of CRTs for entertainment by way of black boxes that augment home TVs. Games on CRTs are available in public places, and home computers offer canned games and the opportunity to create one's own games. Kahn and Lieberman[23] discuss some of the more sophisticated games one can play on such systems.

Comparison of computer graphics and other data processing

A reasonable question to ask might be this: How does computer graphics differ from other data processing applications? There have been plotters for years and drawing with them is very straightforward.

The creation of an interactive display environment for certain kinds of applications can cause one to face many new problems in system design and application design. There are several major considerations of a graphics application design that may not exist in a nongraphics data processing environment.

system and application design problems

Creation of user-oriented pictures. Graphics applications can range in output requirements from simple alphanumeric messages to pictures that are photographic in quality. As computer graphics output requirements grow closer to pictures of photographic quality, the required algorithms and processing increase dramatically. Algorithms may be required, for example, to remove hidden surfaces, produce shaded figures and shadows, and introduce colors in many hues. Considerations for realistic pictures are discussed later in this paper under special techniques.

Creation of user-oriented input mechanisms. In many graphics applications, the user makes decisions at a CRT. To make a user's work environment convenient for his comfort and productivity, his decisions must be given to the system in as natural and easy a manner as possible. The mere procuring of hardware input mechanisms is not sufficient. They must be programmed for good human factors and ease of use. For example, when a user points at something on a display, he should receive immediate feedback about what the system perceives has been pointed to rather than for the system to begin a calculation on perhaps a wrong element. Such an element may be falsely sensed by the system for such reasons as parallax, user change-of-mind, or user mistake. An immediate feedback mechanism would aid in making a forgiving system.

Other aspects of input mechanisms deal with the problem of easily inputting large amounts of 2-D or 3-D information. Designing a hardware/software package that is straightforward to use, prompt with messages, forgiving, and provides feedback on system perceptions requires careful design.

Tying information together with a data base. A prime consideration in any data processing application is the design of data structures that are to be processed by the application. The structure of the data in a data base of a graphics application may have to represent data or relationships that would not normally be considered in other data processing applications.

One level of consideration is that a data base may contain information in normalized or graphics-device-independent form.[24] For many applications, such a data base is said to have *world coordinates*. That is, the data are related to the application in real-world terms, measurements, etc., rather than in the coordinate system of a particular display. Also, some data may be kept about the current display on the screen. Such data can, for example, help an application manage the contents of the screen or help decide which element has been selected by using a light pen or crosshairs.

Key to the design of a world data base is the consideration of the relationships to be kept in the data base. A simple example of graphics relationships is the corner created by the termination of three lines in space. Should the data base contain simply data describing each line as a wholly independent entity or should the data base also note that the lines converge to a point and represent a corner? Should the data base also contain information that the corner is part of a greater entity that may contain other geometric entities? The data base designer's answers to these questions affect the characteristics of the application at the screen. For example, an application programmer cannot easily present a user option to move a corner if the data base does not present corners. Reference 24 considers such matters from a machine-independent point of view.

Supporting information processing requirements for graphics systems. Graphics applications can put a strain on the computing facilities available to a graphics console. Some of the effects of strain are a result of 1. a need for fractional second response (i.e., feedback to a user whose hand is poised, awaiting a response at the screen); 2. a need for a system to handle consistently short user think times. Times of one to two seconds are not unusual for sustained periods, according to Dill and Thomas,[25] who refer to several hundred to a thousand interactions per hour in Computer Aided Design (CAD); and 3. a need for complex pictures with high information content to be drawn and changed rapidly. As images become more complex (whether dense wire images or real-world solid images), the computing time and disk storage accesses required to create an image can increase dramatically. Processing required for hidden-line or hidden-surface elimination, shading, coloration and other real-world effects can cause significant delays in response time. Support for such data processing requirements can be costly; insufficient support may make some graphics applications infeasible.

Addressing the aforementioned considerations is necessary to the implementation of many graphics applications. Solutions can be time-consuming and expensive; improper solutions can defeat a project.

Graphics hardware and software

In the past decade, graphics hardware and software have both made significant strides, certain of which have been chosen for discussion here. Unfortunately, many implementers of graphics applications find too late that they are solving the same problems others have solved before them. They may create a data base structure, session monitor, screen monitor, or a hidden-line algorithm that has been previously published. To attempt to avoid some of this duplication of effort and to plan for the portability of applications and components of applications, a group of individuals and organizations have proposed a set of graphics standards.[26,27]

proposed standards

A Core System has been proposed by the Graphics Standards Planning Committee of the ACM Special Interest Group of Graphics (SIGGRAPH). The basic concepts of the Core Systems are described in a document containing the proposed standard[26] and are summarized here.

1. The separation of input and output functions.
2. The minimization of the differences between producing output on a plotter and on an interactive display.
3. The concept of two coordinate systems, the *world coordinate system* in which the picture for display is constructed, and the *device coordinate system* in which data to be displayed are represented.
4. The concept of a *display file*—containing device coordinate information—used by all but the least interactive graphics systems.
5. The notion of display file *segments*, each of which can be independently modified as a unit.
6. The provision of functions to transform world coordinate data into device coordinates, by invoking a *viewing transformation*.

Because specific graphics devices vary greatly in their capabilities, and because application rquirements can vary so much, the Core System has been created with multiple levels of potential implementation. Michener and Van Dam[27] describe these levels as follows.

A Level 1 implementation of the Core System supports the graphics output needs of applications with no requirements for incremental picture modification, only total picture replacement. Capabilities provided at this level include all two- and three-dimensional output primitives and their attributes, viewing transformations, and control functions. Only nonretained picture segments can be used. Neither retained segment nor interactive capabilities are provided.

A Level 2 implementation of the Core System provides additional capabilities over Level 1 that allow incremental picture modification. Specifically, retained segments can be created, deleted, and renamed. The visibility and highlighting segment attributes are supported. The detectability and image transformation segment attributes and interactive capabilities are not supported.

A Level 3 implementation of the Core System provides the detectability segment attribute and interactive capabilities in addition to capabilities provided by Level 2.

A Level 4 implementation supports all Core System capabilities, that is, it supports image transformations in addition to the capabilities provided by Level 3.

Michener and Van Dam[27] discuss and diagram these four levels. Keller, Reed, and Solem[28] discuss an implementation involving a variety of graphics devices. Chappell and Bono[29] give a status report on a number of different organizations' efforts to implement some version or portion of the standards.

Michener and Van Dam[27] discuss major issues that were faced in creating the proposed standards. Vinberg[30] comments on the proposed standards and points out their emphasis on engineering graphics and insufficient emphasis on data representation graphics.

hardware types

Machover[3] gives an up-to-date description of types of graphics devices and their capabilities, and Orr[10(a)] lists manufactured devices comprehensively. Although most output equipments used in interactive environments are cathode ray tubes (CRTs), there is great diversity among them. Regarding imaging techniques, they may be raster-scan or random beam drive. The random beam may be a regenerative type or a storage device. The resolution of graphics hardware types varies greatly from one to another. A device may be capable of drawing only alphanumeric and special characters or it may have an addressability and resolution capability to create images of photographic quality. A graphics device may have one or many beam intensities and/or colors.[2]

Also, a graphics device may have *function generators,* that is, hardware features to draw certain functions rather than having software produce those functions. As a simple example, consider a circle generator. This requires only X and Y coordinate data and a radius for a wired-in circle generator to position and move the beam of electrons on the face of the CRT. Without the generator, software must calculate the coordinates of many straight-line segments to simulate the circle. Added to function generators, graphics systems may offer rear-screen projection[9] and large-screen projection.[31]

Raster-scan-regenerated CRTs are being offered for sale at computer stores at quite attractive prices. Television is a common example of a raster-scan-regenerated CRT in which both color and high resolution are available. Because of the low cost of these displays and their display buffers, there is much interest in developing their use. Dungan, Stenger, and Sutty[32] and Pavlidis[33] describe techniques for filling in or texturing surfaces on a raster graphics display.

Since raster-scan CRTs take a fixed time to "paint" the screen once, quite complex shaded and multicolored images can be displayed without flicker. Conversely, random-beam displays can flicker because the time to paint once depends on the content of the picture. Real-life displays are usually high in information content and are well suited to raster-scan CRT displays.

Color CRT displays are growing in popularity because they present the capability of highlighting information in a text environment,[17] of helping create greater realism,[18] and of presenting a large amount of information.[11]

In this issue, McManigal and Stevenson[34] discuss a graphics system employing a CRT pair for each user, a regenerating-raster-scan CRT that is used for alphanumeric communication between the user and the computer, and an adjacent high-resolution-storage CRT that is used for graphics viewing. Among the system's many advantages is that alphanumerical communication with the computer does not interfere with pictures being created on the screen.

Although the CRT is the primary type of display device in use today, other graphics devices exist and are used. Tannas and Goede[8] discuss flat-panel or solid-state devices. Plasma display devices are also available as part of an overall display system capability. In many cases, these devices offer a display of only a few characters or have low resolution. Some of the problems that still have to be worked out to make plasma display devices competitive with CRTs are those of availability of gray scales and availability of color. In addition, their cost is often higher than that of CRTs.

A wide variety of noninteractive graphics output devices are also available,[10(a)] including ink and electrostatic plotters and microfilm output equipment. They often augment an interactive graphics system by providing hard copy as a communication medium. In other cases, where graphics interaction is not required or justified, noninteractive devices may be the only graphics output for a system.

Input

Perhaps the one area in which there has been the greatest change, innovation, invention, problem, and controversy has been that of

graphics input technology. Many types of graphics input have been introduced because of the inherent difficulty of efficiently and easily supplying graphics information to a computer. Because of the complexity of choice, users have often been forced to make choices based on intangibles and personal preferences.

In 1964, Jacks[35] specified the need for a static form of input and used a 35-mm flying-spot scanner CRT to digitize drawings. The same author also specified the need for a dynamic input mechanism and used an input pen to point at the CRT display. Today, input devices are used either to input graphics information to a system or to point at a particular area of the screen to initiate some action.

Light pens are commonly used for pointing at a screen to initiate an action, and, to a more limited degree, to draw many data points. These input devices work with regenerated CRTs (either raster-scan or random-beam) as follows. When the beam comes under the light-sensitive transducer in the pen, regeneration momentarily stops to record information about the object being pointed to. Light pens do not work with storage-type CRTs because there the image is painted only once, and thus no beam appears under the pen when the user points. Light pens can be designed for easy use, and with proper programming (to give feedback, to provide symbol tracking, and to allow changes), can approach the flexibility of a pencil on a piece of paper.

Light pens, however, are not the only graphics input mechanism. For pointing purposes, the finger can be used. Smith and Sherwood[9] describe a terminal where the finger is used to indicate choices. A 16 by 16 array of infrared light-emitting diodes and sensors determine the position of the finger. Herot and Weinzapfel[36] describe a much more powerful Touch Sensitive Digitizer (TSD) that detects finger pressure, location, and torque.

Storage-type screens employ other input devices such as a joy stick to move a cross-hair over the display area. Cross-hairs on a screen can also be directed by the input produced from mechanical wheels under a "mouse" moved by hand. Similar devices with static cross-hairs can be used to trace a drawing for input to a system.

More elaborate mechanisms are available for digitizing drawings.[10(a)] These include both manual and semiautomatic digitizers for tracing drawings mounted on a special board that gives position data to produce digitized drawings. Users sometimes wish to extract data from a three-dimensional model. Elaborate schemes using tactile probes or measurement mechanisms employing light or sound are in use today.[37] All these methods extract data from the real world for further processing in a graphics system.

Graphics application systems and components

As with other data processing environments, solutions to graphics problems can be specific and closed in design or general and open-ended. The advantages of closed designs are smaller short-term costs and speed of implementation of an initial application. The advantages of open-ended designs are smaller long-term costs per application and improved speed of implementation in later applications. Both approaches have been taken in various graphics implementations.

A computer-aided design system, the CADAM® system (a registered trademark of the Lockheed Corporation), is designed specifically to aid engineers, designers, and draftsmen in creating engineering drawings and directions for numerically controlled machine tools. By focusing on the creation of the drawing, the CADAM® system was initially limited to a 2-D environment (with tools for combining views into perspective views, called $2^1/_2$-D). A 3-D version is now available. A 2-D system can have many simplifying assumptions in its data base and thus the user can experience very good performance due to short search times.

Weller, Palermo, and Williams[38,39] describe a picture-building system that consists of high-level applications, building tools, and a relational data base. The system is designed for an applications programmer to interactively create and modify graphics applications that are device-independent. Data within the data base are self-describing, and therefore if the user wishes only to display pictures, no application programs need be written. This system is significant in its general purposefulness and its high level of user interface.

Some of the main ingredients in a graphics system are data base (world and picture), session supervisor, and graphics support, which consists of components, system considerations, and special graphics effects. These are now explored.

data base

Consider first a world data base, which has two primary aspects: 1. the data in the system; and 2. relationships recorded and kept among the data elements.

A simple example involves the matter of describing and storing line information about a mechanical part in a data base. Assume a requirement to design a part that can be represented by three orthogonal (orthographic) projections. The system might allow construction of three unrelated representations (views) that contain a line AB. The result is a relatively simple data structure to design and manage, one that is used in some 2-D drawing systems. The disadvantage here is that a later change in a line AB in one view

does not automatically change the other representations of that line. The designer has to change the other views as well.

Another way to store these data is to describe a line mathematically in a three-dimensional coordinate system. That is, the data base contains only one representation of line AB in 3-D space. A drawing probably does not exist in the data base. To make a drawing, the user must request multiple views of the one 3-D model to be projected.

The next level of design is that of the relationships to be kept in the data base. Should a line in the 2-D drawing model be logically connected to other lines that are part of the same view? Should a line in the 3-D model be logically connected to other lines in the same surface? Should surfaces be logically connected together if they belong to the same part? Should common elements of two surfaces be logically connected?

The application designer faces these questions and others. The degree of connectivity he chooses affects the complexity of the final data base and determines the kinds of operators or tools that can be created for the end user at the screen. For example, if in the 2-D drawing data base all geometric objects that are part of one view are so recorded, one can implement a screen tool that allows the end user to move an entire view to different areas on the drawing surface.

A key consideration in the design of a data base is the degree of data changes to be allowed or required at the screen. If an application calls only for viewing a complex object in different views—perhaps even rotating in real time—the data base probably does not have to be complex and have many relationships. If, on the other hand, the user expects to be able to change the design of the object on the screen and requires sophisticated tools for making changes, the data base has to be more complex.

Athay[40] discusses data base requirements for Computer Aided Design (CAD) and looks at the problems of structuring the data so that small amounts of large data bases may be passed on to small computers where resources are limited.

Edson and Lee[41] report on ways to structure data for a digital cartographic data base. Such data bases can be very bulky and require special considerations.

picture data bases

An application or graphics subsystem normally keeps track of the current contents of the screen, in order to provide services to the application program. Examples of services are those of indicating which element by name in the display had been pointed at by a pen (for regenerating CRTs), allowing deletion of single display

elements or logical groups of display elements (as previously defined by the applications program), or allowing operations on display entities or entity groups such as brightening, changing color, or disabling light pen detection. Picture data bases are organized for these purposes.

session supervisor

As with typewriter terminal environments, a terminal monitor or session supervisor is very useful. Such a supervisor is responsible for soliciting from the terminal user names of applications to be run, with names of data files to be used, and aiding in application-to-application transition. In addition, session supervisors are available when a programmer wishes to debug a program. A good supervisor offers a set of high-level debugging tools for the application programmer.

graphics support

Graphics support is one of the richest and most varied areas of computer graphics because it embraces graphics services, languages, and algorithms. It is progress in this area that has allowed the definition of the ACM Proposed Standard on Graphics.[26,27] There are hardware/software systems for handling simple character-oriented displays and advanced graphics systems that can produce detailed pictures that are three-dimensional in appearance. The cost of the graphics components varies widely. Also, the processing cost or time to create a three-dimensional illusion is one or two orders of magnitude greater than that for character displays. Of course, there are not yet truly three-dimensional displays. Such illusions should more properly be called photographic quality, since no attempt is made to give a real third dimension to the image.

proposed Core standards

The ACM Graphics Standards Planning Committee surveyed eight graphics software packages for line drawing graphics systems, and the results of that survey have been published.[26] The elements of a graphics system that were evaluated are outlined here. The major subjects of the analysis are the following:

- Input facilities.
- Output facilities.
- Control facilities.
- Extensions.

Expanded, the subjects are broken down into questions of the capabilities available within particular implementations as follows:

- *Input facilities.* What input devices are supported with input device independence? If menu or graphics item picking is supported, what information is supplied to the application? What is provided about a device's location on the screen? Can input be graphic, alphanumeric, or through buttons?

- *Output facilities*. What coordinate systems are there? What object transformations (change to a graphics object) are supported? What viewing transformations (projection, windowing, etc.) are supported? What logical structure exists for picture elements? What picture element attributes are supported (line width, intensity, blink, and color)? What drawing primitives are supported (points, lines, curves and lines, surfaces, and text)? What is the picture data structure?
- *Control facilities*. What are the requirements for initialization, error reporting, and termination?
- *Extensions (special facilities)*. What special graphing facilities are available, i.e., continuous, discrete, or 3-D? What curve-fitting and data-smoothing facilities are available?

Note that the outline does not ask about any application data base or data structure. The ACM study and proposed standards do not apply to that aspect of a graphics system.

The subject of image transformation (projection onto planes, rotations, etc.) has been treated in many texts. Chasen[42] devotes an entire text to this subject, including 2-D and 3-D geometry and curve and surface fitting (i.e., fitting to constraints, such as raw input data).

Lucido[43] describes a number of software packages that are available for both passive and interactive graphics. He also describes capabilities and actual programmer interfaces in detail.

systems considerations

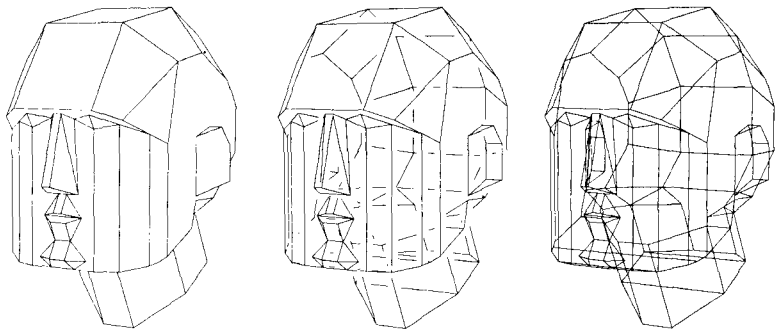
Little and Williams[44] introduce a novel approach to the problem of driving a high-bandwidth device, such as a graphics terminal, over low-speed transmission lines. Essentially, the person at the screen indicates those pictures or picture elements that should be retained in the terminal storage device for later recall. This eliminates the need for the central CPU's application program to plan ahead for recall.

Dill and Thomas[25] report on a study to remotely operate graphics systems and the considerations for placing information within the storage of a satellite CPU. Primary considerations in their design include providing for good human factors (such as immediate brightening for feedback) and for keeping to a minimum the amount of remote CPU storage for the image and light-pen correlation logic. The remote system does not have disk storage.

Special graphics effects

Applications that require photographic quality (3-D) images must employ additional techniques to create the illusion of reality.

Figure 3 A wire head shown with all lines visible, haloing, and hidden-line elimination



The application programmer is concerned about creating the illusion of solid figures (not stick figures), shadows, use of multiple light sources, colors of many hues, and texture. Many papers have been written on these subjects, some of which are outlined here.

References 23, 45, and 46 describe excellent examples of the current state of the art and include problems of the following kinds:

- Hidden-line and hidden-surface elimination.
- Illusion of solids.
- Texture.
- Shadows.

hidden-line and hidden-surface elimination

The hidden-line problem is that of algorithmically determining the data that are not now to be displayed, since those data points, lines, or surfaces lie behind those of another object, based on the current line of sight of the screen. Automatic algorithms have been created by a number of authors. Sutherland, Sproull, and Schumacker[46] compare ten different techniques of hidden-line or hidden-surface elimination. They conclude that sorting is at the heart of the hidden-surface problem, that is, the sorting of objects into those to be displayed and those not to be displayed. Giloi[47] classifies hidden-surface algorithms and shows how specific ones vary in CPU time required for a specific sample object. Many applications, such as Computer Aided Design (CAD), do not eliminate lines that would have been hidden from the line of sight of the display because of the overhead involved.

haloing

Appel, Rohlf, and Stein[48] present a novel approach to the hidden line problem. Figure 3 shows three models of heads to illustrate the method presented in Reference 48. The left representation is the head without hidden-line elimination. The right figure is the result of a standard hidden-line elimination algorithm. The middle

figure shows the result of an approach called "haloing." Lines behind other lines are broken up to give the impression of a halo around the lines closest to the viewer. This technique is simple to implement and is especially suitable for the display of finite-element grids, three-dimensional contour maps, and ruled surfaces.

illusion of solids

When one fills in a description of an object with color or shades of gray the result is the illusion of solidity. Automatic algorithms for doing this are described by Pavlidis.[33] These algorithms follow known boundaries that define shapes and attempt to fill in the display area with color or shades of gray, yet keep separate shapes or areas distinct.

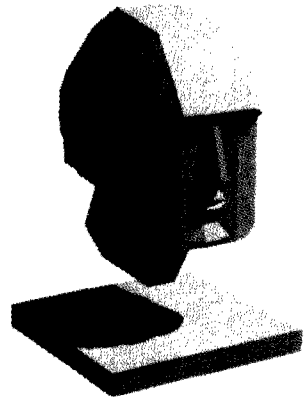
texture

Most real objects have texture. To create a realistic illusion on the screen, algorithms have been devised to add texture to computer graphics. Dungan, Stenger, and Sutty[32] describe a scheme that involves using digitized arrays of stored information that is replicated on a surface within an image. Blinn[49] creates wrinkles on a surface for a very realistic illusion of randomized texture. He does this by changing the parameters used in the intensity calculations; that is, he does not have wrinkles represented in the picture or world data base. He points out that typical pictures take three to seven minutes each to produce.

shading

Blinn[50] also describes algorithms that have been created to calculate light reflections off objects to create the illusion of shading. His work is based on experimental measurement of light reflectance from real objects and is used to differentiate, for example, between light reflections from metallic and nonmetallic surfaces. Blinn and Newell[51] describe algorithms for applying texture and reflections to curved 3-D objects. Objects are broken into patches, and light intensity is calculated per picture element. Patterns to be applied to objects can be mathematically defined, and may come from a digitized hand drawing or from a scanned and digitized photograph of a real scene. The resulting effects are very realistic.

Figure 4 A computer-generated sculptured image



Shadow creation, which aids depth perception, can be used to create more realistic images; it may also be important to an understanding of spatial relationships or whether objects are invisible because of shadows. Figure 4 employs solid-filled planes, hidden-plane elimination, shading, and shadows to create a sculptured image.

As with hidden-line/hidden-surface elimination, a number of techniques have evolved to create shadows in graphic systems. Some techniques work only with objects made of planar polygons,[52] whereas others work with curved surfaces.[53] Other questions that may be dealt with in the creation of shadows are the following: Are single or multiple light sources supported? Does a light

source yield divergent rays? Or is it a point light source at infinity that yields parallel rays? Is the light source of finite size, thus yielding a penumbra instead of a geometric shadow? Crow[54] describes three classes of shadow algorithms and discusses their relative advantages and disadvantages.

Many of the aforementioned shadowing techniques are costly in terms of CPU time. Therefore, they are used in creating still output or are used to create the frames that make up a movie (i.e., for animation). To be able to use these techniques in a dynamic, interactive graphics console environment necessitates either very large processors or special hardware assistance at the graphics console. Some of these techniques, therefore, are implemented with special hardware function generators.

Concluding remarks

Many obstacles to creating productive graphics applications have been removed or reduced. Significant effort is still required to make the following items available:

- A variety of end-user-oriented graphics applications that require no additional programming.
- A variety of display devices and systems varying widely in capability and price.
- A variety of high-level graphics support packages for creating one's own applications.
- A well-documented list of users who attest to productivity gains.
- Documentation of solutions to many of the more complex problems in graphics,[5(b),55] such as data base design and special effects.
- Computer systems that offer special advantages to graphics applications, such as virtual storage, virtual machines, and response-oriented operating systems.

In a philosophical vein, Sagan[21] summarizes the potential of interactive graphics by observing that "Computer graphics have now reached a state of sophistication that permits important and novel kinds of learning experiences in arts and sciences, and in both cerebral hemispheres. There are individuals, many of them analytically extremely gifted, who are impoverished in their abilities to perceive and imagine spatial relations, particularly three-dimensional geometry. . . . (Computer graphics) is an extraordinary tool for improving our ability to visualize three-dimensional forms—a skill extremely useful in graphic arts, in science and in technology. It also represents an excellent example of coopera-

tion between the two cerebral hemispheres: The computer, which is a supreme construction of the left hemisphere, teaches us pattern recognition, which is a characteristic function of the right hemisphere."

ACKNOWLEDGMENT

The author thanks Arthur Appel of the IBM Thomas J. Watson Research Center for providing all the figures used in this paper.

CITED REFERENCES

1. *IBM Systems Journal* **7**, Nos. 3 and 4 (1968).
2. J. F. Musgrave, "Experiments in computer-aided graphic expression," *IBM Systems Journal* **17**, No. 3, 241-259 (1978).
3. C. Machover, "Graphic displays," *IEEE Spectrum* **14**, No. 8, 24-32 (August 1977), and **14**, No. 10, 22-27 (October 1977).
4. A. Feder, *Test Results on Computer Graphics Productivity for Aircraft Design and Fabrication*, Paper No. 75-967, presented at the AIAA 1975 Aircraft Systems and Technology Meeting, Los Angeles, CA, Aug. 4-7, 1975, American Institute of Aeronautics and Astronautics, New York, NY (1975).
5. (a) B. J. Shepherd, "Experimental page makeup of text with graphics on a raster printer," *IBM Systems Journal* **19**, No. 3, 345-355 (1980, this issue).
   (b) U. W. Pooch, "Computer graphics, interactive techniques and image processing 1970-1975: A bibliography," *Computer* **9**, No. 8, 46-64 (August 1976).
6. S. Handelman, "A high-resolution computer graphics system," *IBM Systems Journal* **19**, No. 3, 356-366 (1980, this issue).
7. R. Williams, *Image Processing and Computer Graphics*, Research Report RJ 2336, IBM Research Laboratory, San Jose, CA 95193 (1978).
8. L. E. Tannas and W. F. Goede, "Flat-panel displays: A critique," *IEEE Spectrum* **15**, No. 7, 26-32 (July 1978).
9. S. G. Smith and B. A. Sherwood, "Educational uses of the PLATO computer system," *Electronics: The Continuing Revolution*, American Association for the Advancement of Science, Washington, DC (1977).
10. (a) J. N. Orr, "Computer graphics," *Mini-Micro Systems* **12**, No. 12, 66-78 (December 1977).
    (b) R. Mallary and M. Ferraro, "An application of computer-aided design to the composition of landforms for reclamation," *Computer Graphics* **11**, No. 2, 1-7 (Summer 1977).
11. L. E. McCleary, "Techniques for the display of ocean data on a raster-driven color CRT," *Computer Graphics* **11**, No. 2, 98-101 (Summer 1977).
12. M. Levoy, "A color animation system based on the multiplane technique," *Computer Graphics* **11**, No. 2, 65-71 (Summer 1977).
13. E. Catmull, "The problems of computer-assisted animation," *Computer Graphics* **12**, No. 3, 348-353 (August 1978).
14. M. Gonin and T. Moffett, "ARTES, an interactive highway design program," *Computer Graphics* **10**, No. 2, 268-274 (Summer 1976).
15. R. K. Lohrding, M. M. Johnson, and D. E. Whiteman, *Computer Graphics for Extracting Information from Data*, Report LA-UR-77-2456, Los Alamos Scientific Laboratory, Los Alamos, NM, 1977.
16. K. Brassel, J. Utano, and P. Hanson, III, "The Buffalo crime mapping system: A design strategy for the display and analysis of spatially referenced crime data," *Computer Graphics* **11**, No. 2, 78-85 (Summer 1977).
17. H. P. Frei, D. L. Weller, and R. Williams, "A graphics-based programming-support system," *Computer Graphics* **12**, No. 3, 43-49 (August 1978).
18. R. Weinberg, "Computer graphics in support of space shuttle simulation," *Computer Graphics* **12**, No. 3, 82-86 (August 1978).

19. N. Grove, "Supertankers, giants that move the world's oil," *National Geographic* **154,** No. 1, 102-124 (July 1978).
20. W. H. Huggins, "What is needed?", *Proceedings of the Battelle Computer Graphics Conference, Computer Graphics* **8,** No. 1, 32-44 (Spring 1974).
21. C. Sagan, *The Dragons of Eden: Speculations on the Evolution of Human Intelligence*, Random House, Inc., New York, NY (1977), p. 212.
22. C. Csuri, "Computer graphics and art," *Proceedings of the IEEE* **62,** No. 4, 503-515 (April 1974).
23. K. Kahn and H. Lieberman, "Computer animation: Snow White's dream machine," *Technology Review* **80,** No. 1, 34-46 (October/November 1977).
24. D. L. Weller, E. D. Carlson, G. M. Giddings, F. P. Palermo, R. Williams, and S. N. Zilles, "Software architecture for graphics interaction," *IBM Systems Journal* **19,** No. 3, 314-330 (1980, this issue).
25. J. C. Dill and J. J. Thomas, "On the organization of a remote low cost intelligent graphics terminal," *Computer Graphics* **9,** No. 1, 1-8 (Spring 1975).
26. *Computer Graphics* **11,** No. 3 (Fall 1977).
27. J. C. Michener and A. Van Dam, *ACM Computing Surveys* **10,** No. 4, 381-387 (December 1978).
28. R. G. Keller, T. N. Reed, and A. V. Solem, "An implementation of the ACM/SIGGRAPH Proposed Graphics Standard in a multisystem environment," *Computer Graphics* **12,** No. 3, 308-312 (August 1978).
29. G. Chappell and P. Bono, "Core system implementation—a status report," *Computer Graphics* **12,** No. 4, 53-66 (December 1978).
30. A. Vinberg, "Position paper on graphics standards," *Computer Graphics* **12,** No. 4, 46-52 (December 1978).
31. W. C. Donelson, "Spatial management of information," *Computer Graphics* **12,** No. 3, 203-209 (August 1978).
32. W. Dungan, A. Stenger, and G. Sutty, "Texture tile considerations for raster graphics," *Computer Graphics* **12,** No. 3, 130-134 (August 1978).
33. T. Pavlidis, "Filling algorithms for raster graphics," *Computer Graphics* **12,** No. 3, 161-166 (August 1978).
34. D. F. McManigal and D. A. Stevenson, "Architecture of the IBM 3277 graphics attachment," *IBM Systems Journal* **19,** No. 3, 331-344 (1980, this issue).
35. E. L. Jacks, "A laboratory for the study of graphical man-machine communication," *1964 Fall Joint Computer Conference Proceedings* **26,** Part 1, 343-350 (1964).
36. C. F. Herot and G. Weinzapfel, "One-point touch input of vector information for computer displays," *Computer Graphics* **12,** No. 3, 210-216 (August 1978).
37. R. W. Decker, "Computer aided design and manufacturing at GM," *Datamation* **24,** No. 5, 159-165 (May 1978).
38. D. Weller and R. Williams, "Graphic and relational data base support for problem solving," *Computer Graphics* **10,** No. 2, 183-189 (Summer 1976).
39. F. P. Palermo and D. L. Weller, *Picture building system*, Research Report RJ 2436, IBM Research Laboratory, San Jose, CA 95193 (1979).
40. R. J. Athay, "Object models for computer aided design—an overview," *Computer Graphics* **12,** No. 3, 239-244 (August 1978).
41. D. T. Edson and G. Y. G. Lee, "Ways of structuring data within a digital cartographic data base," *Computer Graphics* **11,** No. 2, 148-157 (Summer 1977).
42. S. H. Chasen, *Geometric Principles and Procedures for Computer Graphics Applications*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1978).
43. A. P. Lucido, "Software systems for computer graphics," *Computer* **9,** No. 8, 23-32 (August 1976).
44. W. D. Little and R. Williams, "Enhanced graphics performance with user controlled segment files," *Computer Graphics* **10,** No. 2, 179-182 (Summer 1976).
45. *Proceedings of SIGGRAPH '78, Computer Graphics* **12,** No. 3 (August 1978).

46. I. Sutherland, R. Sproull, and R. Schumacker, "A characterization of ten hidden-surface algorithms," *ACM Computing Surveys* **6,** No. 1, 1-55 (1974).
47. W. Giloi, *Interactive Computer Graphics*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1978).
48. A. Appel, F. Rohlf, and A. Stein "The Haloed Line Effect for Hidden Line Elimination," *Computer Graphics* **13,** No. 2, 151-157, (Summer 1979).
49. J. F. Blinn, "Simulation of wrinkled surfaces," *Computer Graphics* **12,** No. 3, 286-292 (August 1978).
50. J. F. Blinn, "Models of light reflection for computer synthesized pictures," *Computer Graphics* **11,** No. 2, 192-198 (Summer 1977).
51. J. F. Blinn and M. E. Newell, "Texture and reflection in computer generated images," *Communications of the ACM* **19,** No. 10, 542-546 (October 1976).
52. P. Atherton, K. Weiler, and D. Greenberg, "Polygon shadow creation," *Computer Graphics* **12,** No. 3, 275-281 (August 1978).
53. L. Williams, "Casting curved shadows on curved surfaces," *Computer Graphics* **12,** No. 3, 270-274 (August 1978).
54. F. C. Crow, "Shadow algorithms for computer graphics," *Computer Graphics* **11,** No. 2, 242-248 (Summer 1977).
55. G. F. Schrack, "Current literature references," *Computer Graphics* **12,** No. 4, 114-123 (December 1978).

The author is located at the IBM Systems Research Institute, 205 East 42nd Street, New York, NY 10017.

Reprint Order No. G321-5126.