

Tekniques

Smalltalk: Fifth-Generation Programming

System Workspace
The Smalltalk-80tm System V
Copyright (c) 1989 Xerox Corp.
All rights reserved.
Enhancements
Copyright (c) 1984, 1985 Tektronix,
All rights reserved.

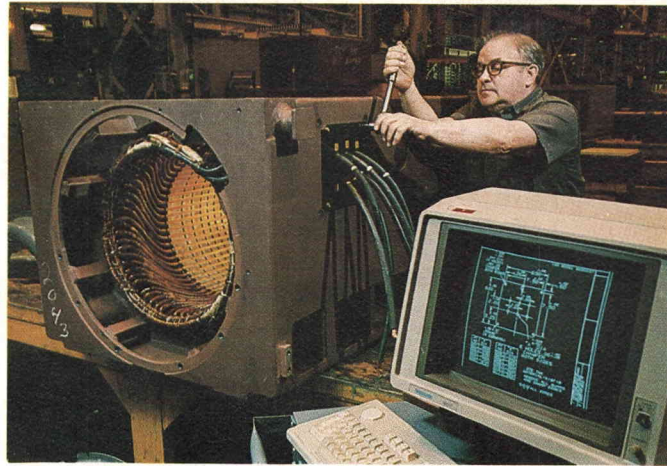
System Browser
Graphics-Primitives
Graphics-Display Objects
Graphics-Paths
Graphics-Views
Graphics-Editors
Graphics-Support
Kernel-Objects
Kernel-Classes
Kernel-Methods
Kernel-Processes
Kernel-Support

Create File System
instance creati
examples

Pen
Point
Quadrangle
Rectangle

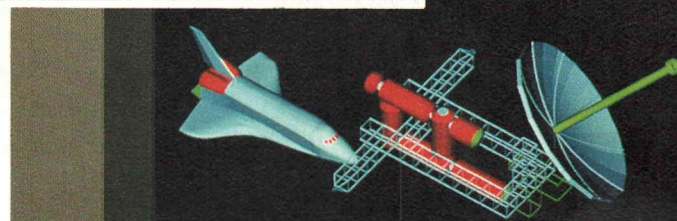
instance class

example
"Draws a spiral in gray with a pen that is 4 pixels wide."
| bic |
bic + Pen new.
bic mask: Form gray.
bic defaultNib: 4.
bic combinationRule: again | der.
1 to: 50 do: [:i | bic
Pen example | copy | cut | paste | do it | print it | accept | cancel | format | spawn | explain
undo | redo | bic turn: 89]



▲ On the Shop Floor at G.E.

Viewing ▶ Solid Models



Tek Terminals on the Shop Floor

At GE's Large Motor and Generator Dept., the 4107's zoom and pan, local storage and rugged construction are useful in manufacturing data display.

4107 Helps in GE's Move Toward Paperless Manufacturing

Economic conditions and international competition have put pressure on American manufacturers to cut production costs and improve product quality. To achieve these goals, many manufacturers are turning toward increased computerization. Thus, in addition to CAD and CAM, companies are also taking steps toward CIM—computer-integrated manufacturing.

“One advantage is greater accuracy through improved revision control. . . You're assured of having the right drawing, the latest drawing.”

CIM involves the sharing of data—using the information generated during the design process to provide a basis for product manufacturing itself, as well as for associated processes such as manufacturing planning and product evaluation.

Typically, the design data is generated on engineering workstations. However, many

other individuals need to view the data, and the engineering workstations used to create the designs are not appropriate for those who only need to examine the drawings. They're expensive, their capabilities are more than are required for graphic data display, and they may be too delicate to hold up under the environmental conditions on the shop floor.

Many companies are finding that Tek's 4100A Series Computer Display Terminals represent an affordable solution for data display throughout the manufacturing organization. They have the the display quality and graphics features needed in the manufacturing environment, their VT100* compatibility means they can double as general-purpose terminals, and they are rugged enough for a majority of shop floor environments.

4107s Provide Solution for GE's Large Motor Dept.

One organization that is using 4100 Series terminals for manufacturing data display is the Large Motor and Generator Depart-

ment (LM&G) of General Electric. Located in Schenectady, New York, LM&G employs over 800 people and produces induction and synchronous motors ranging from about 500 horsepower to 10,000 horsepower, and synchronous generators from 1,000 KW to 6000 KW. (See Figure 1.) LM&G's products, including its popular new Phoenix™ motors and generators, are used in a variety of industrial applications.

“We've been using computers for product design a long time,” says Chris Chartrand, LM&G's manager of systems development. “Now, we're taking advantage of that data for manufacturing as well. We're drawing on our design data base and pushing that information on to the shop floor.”

At LM&G, engineering design work is done on a VAX* 11/780 minicomputer. Drawings are stored on a second VAX, along with the software that controls their access and display. In some cases, a design can go to a numerical control

On the cover:

At General Electric's Large Motor and Generator Dept., Joseph LaPlante uses a 4107 terminal to view an engineering design.

MCS's OMNISOLIDS™ and Tek's 4129 workstation teamed to produce this representation of the space shuttle and space station. Three articles on solid modeling begin on page 8.

Smalltalk on Tek's 4400 Series AI workstations is a fifth-generation programming language ideally suited for AI research, and application development and prototyping. Stories begin on page 20.

In This Issue

4107 Helps in G.E. Manufacturing	2
Tek's Newest Terminal	4
TekniCAD Version 8.1	5
Using Your Terminal with a 6130 . . .	6
Solid Modeling Overview	8
4129 Viewing System	12
Solution Profile: OMNISOLIDS ..	16
Programming with Smalltalk . . .	20
4400 Series Architecture	23
Program Animation	24
4510A Rasterizer	25
Software Development on a 6130 27	
Software for IBM®	
Environments	29
In Brief	29
Programming Tips	30
Training Workshops	31

Tekniques, the IDG Applications Newsletter, is published by the Information Display Group of Tektronix, Inc., Mail Stop 63-635, P.O. Box 1000, Wilsonville, OR 97070. It is distributed to users of TEKTRONIX computers and graphics products.

The information in this newsletter is subject to change without notice and should not be taken as a commitment by Tektronix, Inc. Tektronix disclaims all liability for errors that may appear in this publication.

Copyright © 1986, Tektronix, Inc.
All rights reserved

(NC) lathe or other machine via tape, and the part can be made with minimal operator intervention. More frequently, the part must be machined by conventional methods.

Employees in a number of LM&G manufacturing areas use Tek's 4107 Computer Display Terminal to view engineering designs. In the final motor assembly area, for example, workers use 4107s to call up the drawings to be assembled. Other 4107s are used by manufacturing planners, manufacturing engineers, and quality control inspectors. Producibility engineers use the 4107s to look at designs and evaluate their producibility - whether the design can be built to engineering tolerances using current equipment. In the Incoming Parts Inspection Areas, inspectors use the terminals for reference in comparing actual parts with the drawing specifications.

"The local zoom and pan is a key feature. . . You certainly couldn't do this kind of work on a personal computer."

The 4107s are also part of "manufacturing information centers" strategically located in the factory. Tek's 4695 Color Graphics Copiers are also part of the centers, ready for the occasional need for a hard copy.

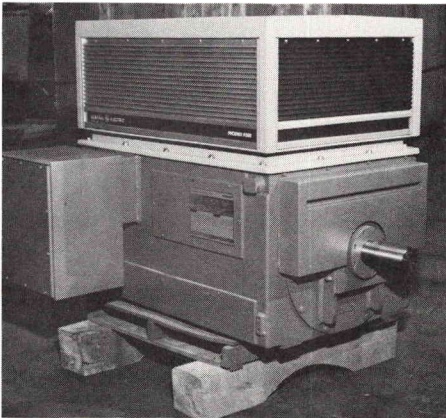


Figure 1. GE's Large Motor and Generator Dept. designs and produces motors such as this 900-horsepower P211 Frame Phoenix™ motor.

Improved Accuracy, Productivity

Chartrand notes several benefits from having manufacturing access to online drawings. "One advantage is greater accuracy through improved revision control," he comments. "Communicating design changes in a paper drawing environment requires someone tracking down each print on the shop floor and

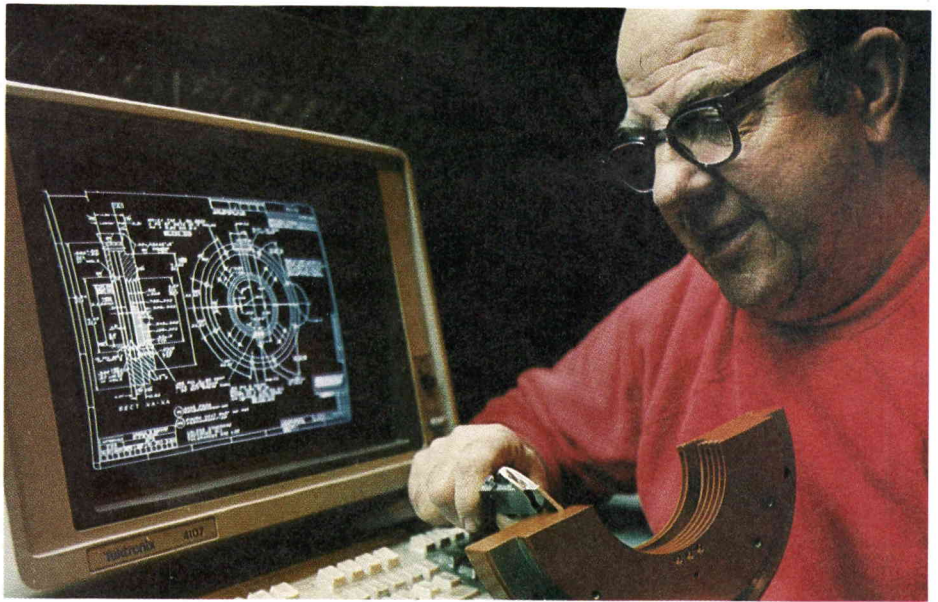


Figure 2. Quality-control inspector Gill Staring uses a micrometer to compare a part's actual dimensions to those specified by the design drawing.

replacing it with the updated version. With the drawing online, you're assured of having the right drawing and the latest drawing—and if a design change is in process while an operator attempts to retrieve that drawing, the software issues an alert and prevents access."

"Obviously, another benefit is that you eliminate the shuffling of paper," Chartrand continues. Accordingly, he has found that the use of online drawings saves time: "It's much quicker to download a drawing to the terminal at the work station than to go to another location, find the drawing and make a copy."

4107 Makes the Job Easier

Readability is an important consideration when the drawings being displayed are complex and the costs of misreading a figure are high. Chartrand has found that the 4107's graphics capabilities are helpful to operators viewing complex design and manufacturing drawings.

"The local zoom and pan is a key feature of the terminal," he explains. "It's very important to be able to pan over a drawing and zoom in on a specific part of it, particularly when you're dealing with large drawings. Having the print or drawing local—that is, downloaded to the terminal rather than stored in the mainframe computer—allows this to occur quickly. It also minimizes the CPU drain on the central system."


In addition, the 4107's 640 x 480 addressability and optional color display combinations help ensure good readability. "You

certainly couldn't do this kind of work on a personal computer," says Chartrand.

Chartrand has found the 4107s to be sturdy and reliable. "When we were looking at terminals, we were told that the Tek terminals were rugged, and we've found that to be true," he says. "The construction of the terminal and the keyboard is heavy."

"We were told that the Tek terminals were rugged, and we've found that to be true."

An Evolving Process

Tek graphics products have been part of LM&G for the past several years, and as the need for computer data display has moved from the office to the shop floor, it's been natural for Tek terminals to move there as well. Chartrand sees computerized manufacturing as an evolutionary process. "There's still a lot we want to do," he says. But he believes the 4107's intelligence and memory options give it the flexibility to keep pace as requirements continue to evolve. 

Phoenix is a trademark of General Electric Corporation. VAX and VT100 are trademarks of Digital Equipment Corporation.

Photography for this article courtesy of G.E.

The 4111's 32-bit virtual graphics space, 1024×768 pixel display and 16 displayable colors make it a cost-effective choice for architectural engineering, cartography, and computer-aided design.

Tek's Newest Terminal—A Low-Cost, High-Performance Tool for CAD Engineers

CAD professionals have recognized the need for a mid-range computer display terminal that provides sufficient speed and resolution for low-end CAD applications. With the introduction of the new Tektronix 4111 Computer Display Terminal, that need has been met (Figure 1).

The 4111, newest member of the Tek 4100 Series, neatly rounds out the Tektronix family of high-performance desktop graphics terminals and provides a low-cost vehicle for architectural engineering, cartography, and CAD applications such as electrical engineering, and mechanical drafting. Because it provides the high resolution and advanced capabilities usually found on much higher priced systems, the 4111 offers superior price/performance value. Features such as the 1024 by 768 pixel display, 32-bit addressability, and 16-color graphics make the 4111 a practical addition to any engineering group looking for cost-effective CAD solutions.

At the same time, the 4111 has a high degree of compatibility with 4100 Series terminals and therefore offers the many graphics functions users have come to expect from Tek terminals—multiple views, local picture segments, surface support, and true zoom and pan, to name a few. It also comes standard with 256K bytes of local RAM for temporary storage of graphics elements, with provisions for adding an optional megabyte of RAM.

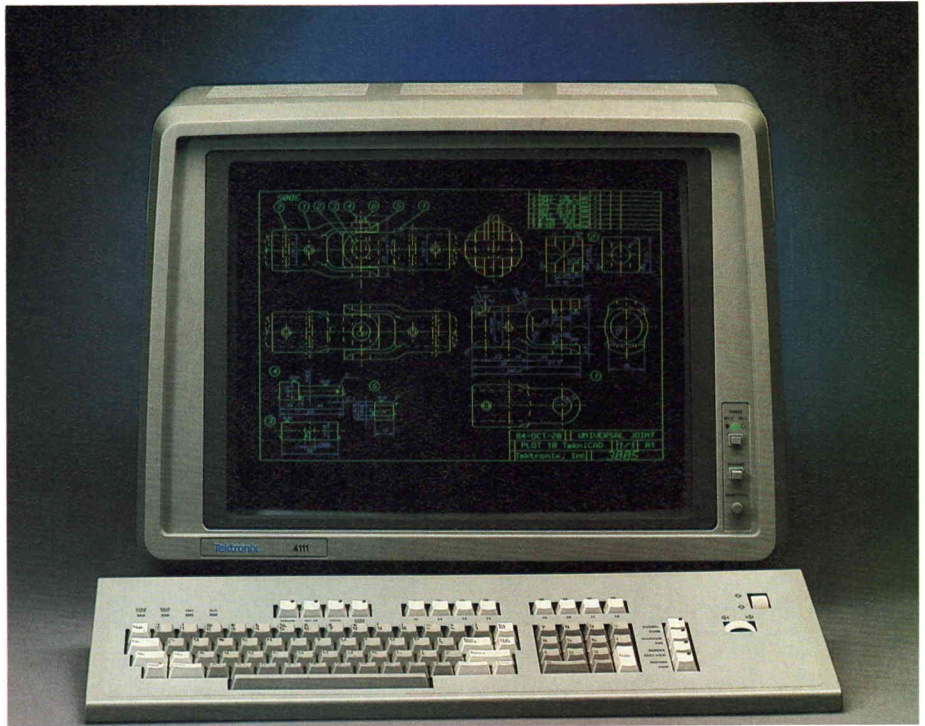


Figure 1. Fitting neatly between the 4109A terminal and the high-performance 4120 Series graphics workstations, the 4111 is ideally suited for personal use by CAD professionals.

Powerful, Affordable CAD

The 4111 offers powerful capabilities essential to many CAD applications. One advanced 4111 feature is its 32-bit virtual graphics space, which provides over four billion by four billion addressable points. Using this capability, CAD engineers can easily load drawings of entire buildings, for example, and zoom in on the details of a particular room.

For superior image quality, 4111 graphics are displayed in a 1024 by 768 pixel matrix on a 19-inch color raster display that is built around a precision, in-line gun CRT. This fits nicely between the 640×480 resolution of the 4109A and the 1280×1024 resolution of the 4125 Color Graphics Workstation.

Four bit planes come standard with the 4111, offering a choice of up to 16 colors that can be displayed simultaneously on

the screen, from a palette of 4096 shades. Each bit plane can be treated as a separate display surface, allowing data manipulation by layers—a feature that is especially useful for IC and printed circuit board design.

CAD professionals will also appreciate the 4111's thumbwheels, which provide the high interactivity necessary in most CAD programs. Adding to the 4111's enhanced user interactivity are a numeric keypad with 10-key functionality; ports for connecting a joystick, mouse and graphics tablet; and eight dedicated programmable function keys, each with four possible macro definitions. Tek's 4957 and the larger 4958 Graphics Tablets are both supported.

Two peripheral ports also come standard with the 4111. One port can be configured as either an RS-232 port or an RS-422 in-

Using Tek's 6130 Intelligent Graphics Workstation with your existing Tek terminal gives you a full workstation configuration at low additional cost.

6130 Plus Your Tek Terminal— An Economical Workstation Solution

The benefits of engineering workstations have received widespread attention, both in *Tekniques* (see Vol. 8, numbers 2, 3 and 4) and in many trade publications. Typically, workstations foster engineering and scientific productivity by providing distributed processing, a dedicated computing resource, and local mass storage.

With many workstation vendors, however, a workstation is a "package deal." Application software supports only the display subsystem that is specifically designed for that workstation, and in order to get a workable configuration, you have to purchase both the display and the processing module. Any displays that have already been purchased have to be put to other uses—or put out to pasture. Any software you have developed must be rewritten—or thrown away.

Unbundling the Workstation

Tek took a different approach in developing its 6130 Intelligent Graphics Workstation (Figure 1). Preserving a customer's investment in existing hardware and software is a cornerstone of Tek's approach to providing graphics solutions. Consequently, the 6130 can be used with the full spectrum of Tektronix graphics terminals. The 6130 can connect via RS-232-C with Tek terminals, and via RS-232-C or RS-422 with the new 4111 terminal and 4120 Series graphics workstations.

This "unbundled" approach means that if you currently own a Tektronix graphics terminal and would like a scientific or engineering workstation, you're already halfway there. For just the cost of the 6130 processing module, you can have full workstation benefits while preserving your existing software and hardware. And because it's a Tek-supplied solution, you're ensured of superior graphics performance and excellent service and support—all at far less cost than if you were purchasing an entirely new workstation.



Figure 1. The 6130 is a flexible, configurable system built to handle a wide range of engineering and scientific tasks.

6130: The Workstation for You?

The question that remains to be answered is whether the 6130 can meet your particular application requirements. Chances are, the answer is "yes." Designed for high performance at low cost, the 6130 is a highly flexible, configurable workstation that can handle a wide range of engineering and technical applications.

The 6130 is a powerful, single-bus system built around National Semiconductor's NS32016, a 32-bit processor with a 16-bit external data path and 32-bit registers. A 32-bit floating point processor and 32-bit memory management unit augment the CPU. The operating system, UTeK*, is an enhanced Unix* implementation that provides significant compatibility with both AT&T's System V and Berkeley's 4.2 versions.

Preconfigured 6130S Application Systems bring together the hardware and software needed for specific application tasks such as computer-aided drafting, software engineering, and data analysis and presentation. Or you can build your own workstation configuration, choosing memory options from 1 to 7 megabytes, disk storage from a 360-kilobyte flexible disk to an 80-megabyte hard disk. Available I/O ports include DMA, RS-232-C, high-speed serial, high-speed GPIB, SCSI, and

Centronics*-compatible general-purpose parallel. The hardware summary table lists standard and optional hardware.

Workstations are interconnected via the 6130's integrated Local Area Network (LAN), so that you can share the use of peripherals such as copiers, printers and mass storage devices. In addition, you can use the 6000 Family's electronic mail facilities and Distributed File System (DFS) to exchange information and to share files and programs with other users.

Software Support

Using your Tek terminal with a 6130 gives you a large pool of software to draw on. Software written using the PLOT 10® Interactive Graphics Library (IGL) and PLOT 10 Terminal Control System (TCS) can be migrated directly to the workstations. The Casual User Interface (CUI)* provides an alternate system interface.

For application software, Tek offers PLOT 10 Computer-Aided Drafting (TekniCAD) and PLOT 10 Computer-Aided Presentation Software (TekniCAP). In addition, a number of software packages are available through Tek's Solution Vendor Program, including:

- PicSure®, DI-3000®, DI-TEXTPRO®, GRAFMAKER* Metafile and the Contouring System, from Precision Visuals

- Visual:ProChart*, Visual:C-chart* and Visual:GKS*, from Visual Engineering
- QUIK*, from Sierra Geophysics

For developing new software, you can choose from C, Pascal or a highly-optimized FORTRAN '77. All three compilers share a common code generator, so a program written in any of the three languages can call a subroutine written in one of the others. Thus, you can choose the language that is most appropriate for each task a program will perform instead of having to use one language for the entire program. The new ANSI BASIC is also available.

For 4120 Series workstations, the PLOT 10 Software Terminal Interface (STI), which provides a library of 4120-specific graphic routines, includes I/O routines for the UTeK operating system.

Standard and High-Speed Serial Interfaces Available

The 6130's standard, dual RS-232-C port (see Figure 2) supports a single user at up to 9600 baud. For higher-speed RS-232-C communications, you can choose 6130 Option 3A (61KR01), which can handle multiple users at up to 38.4K baud. High-speed RS-422 communications (at up to 400K bits/second) are available with the 6130's Option 3C, the Serial Synchronous/Asynchronous Interface board

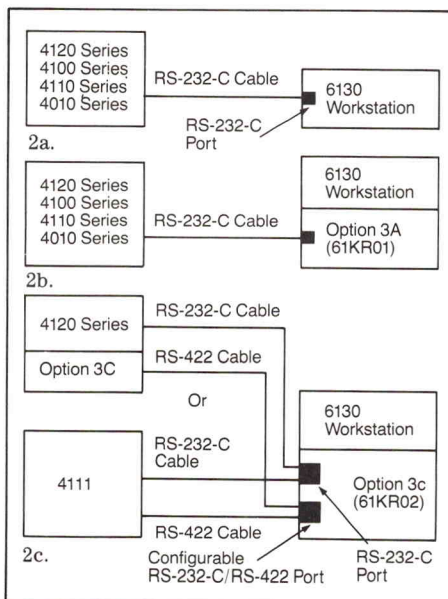


Figure 2. Standard and high-speed RS-232-C interfaces are available, as well as high-speed RS422 for the 4111 terminal and 4120 Series workstations. The standard RS-232-C interface (2a) supports a single user at up to 9600 baud. The high-speed RS-232-C interface (2b) can handle multiple users at rates up to 38.4K baud. RS-422 communications (2c) occur at up to 400K bits per second.

6130 Hardware Summary

	Standard	Optional/Additional
Processors	32016 CPU 32081 Floating Point Processor 32082 Memory Management Unit	
Memory	1 MB	1 MB, 2 MB, 4 MB or 6 MB
Storage	360 KB diskette 20 MB hard disk	40 MB or 80 MB hard disk (substitute) External 45 or 60 MB cartridge tape and 40 MB hard disk
Interface Ports	RS-232-C (2) GPIB LAN	High-speed RS-232-C (2) High-speed RS-232-C/RS-422 High-speed GPIB Dual Centronics SCSI
Peripherals		4644 Dot Matrix Printer 4691, 4692 4695 Color Graphics Copiers* 4510 Color Graphics Rasterizer Support for line printer, letter-quality printer

*Attached to a 4100 or 4120 Series terminal/workstation.

(61KR02). If you have a 4115B or 4120 Series workstation, you'll need the workstation's Option 3C; for the 4111 terminal, one of the two peripheral ports can be configured as an RS-422 interface.

The high-speed RS-422 interface is designed for the transfer of large amounts of data for complex and dense graphics, as in applications such as CAD/EE, logic design, and CAD/ME. The RS-232-C interface is used to control RS-422 communications. An application can pick the most appropriate interface by switching back and forth between them under software control. The RS-232-C interface is used for interactive work composed of short data streams, such as initializing the terminal, graphic input, menu selection, or software development.

Using the Standard RS-232-C Interface

Linking the 6130 to your graphics terminal is a straightforward task. To use the RS-232-C interface, the terminal requires nothing beyond standard RS-232-C cabling. Connect the terminal and the 6130, and your hardware is ready to go. All software configuration can be done through the 6130's interactive, menu-driven **sysadmin** interface, a standard part of UTeK.

Like most Unix-based systems, UTeK has an **/etc/termcap** (or 'terminal capabilities') file that contains encoded descriptions of terminals that are recognized by the system. UTeK's **/etc/termcap** file includes entries for Tek terminals, so you simply need to respond to the **sysadmin** question about terminal type by entering your terminal type—4107, 4128, etc.


If more than one terminal will be connected to the 6130, you will also need to configure the RS-232-C port. Again, the **sysadmin** interface will lead you through the required steps. (The primary RS-232-C is already configured.)

Adding the RS-422 Interface

To add the RS-422 interface, you must first configure the RS-232-C interface, which will act as a communications controller. For the 4115B/4120 Series workstations, you'll need the Option 3C Interface board, which includes RS-422 cabling. Follow the steps outlined above for software configuration; then configure the RS-422 port. If there will be other users on the system, use the UTeK **assign** and **deassign** commands to assign the port to yourself and release it when you log out. (This is a good practice to follow even if you're the only user.)

Lower levels of the RS-422 device driver are standardized and included in UTeK and in the 4120 Series firmware. Application-level software must handle the RS-422 I/O channel interface and the higher levels of the device driver.

PLOT 10 is a registered trademark of Tektronix. CUI and UTeK are trademarks of Tektronix.

Centronics is a trademark of Centronics Corp. DI-3000, DI-TEXTPRO, GRAFMAKER and Picture are trademarks of Precision Visuals Corporation. QUIK is a trademark of Sierra Geophysics. Unix is a trademark of AT&T Bell Laboratories. Visual:C-chart, Visual:GKS and Visual:ProChart are trademarks of Visual Engineering. 

A Solid Modeling Overview

The capacity to capture an object's solid and void parts, and to distinguish between its inside and outside are two major reasons solid modeling is considered the key to integrating CAD/CAM.

Solid Modeling—Key to Integrated CAD/CAM

Solid modeling is evolving as a powerful computerized tool for integrating mechanical product development. From the first concept of a product through its final production, solid modeling promises to perfect a central database from which all engineering processes can operate.

Although the realistic, color-shaded displays produced by solid modeling systems are often thought to be its focus, graphic rendering is just one of the applications drawing from the central computerized model. In this article we will briefly review solid modeling concepts, components, and terminology, and look at the role played by computer graphics.

Eliminating Ambiguity

Developing a product is an iteration of steps—conceptualizing; creating, modifying, and analyzing. At each iteration, object shapes, spatial relationships, and attributes are considered. Before the advent of computers, the information generated at each step was communicated in engineering line drawings as a collection of two-dimensional lines and arcs, annotated with tolerances and dimensions. These drawings relied on the human ability to process the data—the ability to recognize patterns, interpret production procedures based on existing knowledge, identify interference problems, and so forth.

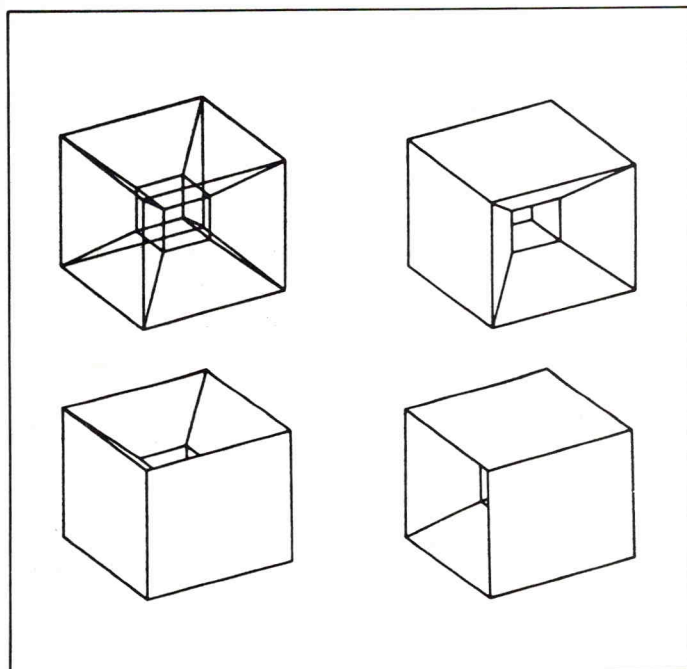


Figure 1. Ambiguity of wireframe model leads to three possible interpretations.

More recently, computer-aided drafting has replaced many of the manual drawing tasks, and two dimensions have been generalized to three dimensions. But the computerized shape information usually remains as graphic data, mainly in lists of lines and arcs, and still requires human interpretation. Commonly known as *wireframe* models, the lines and arcs represent a *view* of the object rather than the object itself. Information about the object's surface and interior is missing from the computerized data. This leads to ambiguity that is resolved only through human translation. Two different physical objects, for example, can be inferred from the same wireframe representation (Figure 1). Wireframe models are potentially incomplete or invalid as well, and depend on the human interface to catch any missing pieces (Figure 2).

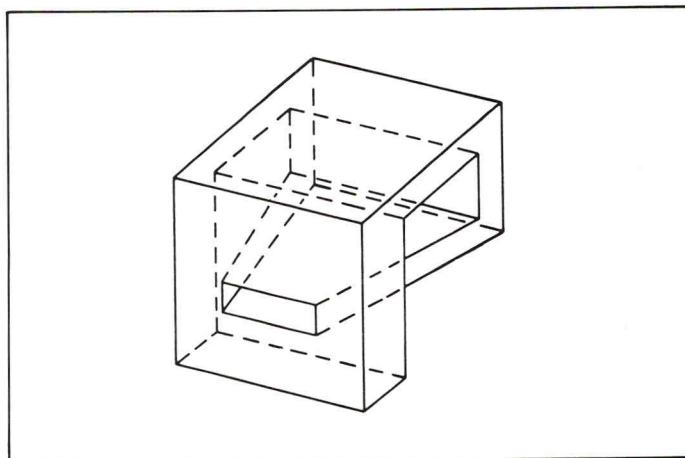


Figure 2. Wireframe models can result in invalid object description (a Klein bottle).

Surface models add information about the pieces of the object that enclose its volume; these models thus provide the ability to suppress hidden lines in a graphic display. But surface models still lack information about the object's interior.

Because of the limitations of wireframe and surface models, it is often difficult or impossible to pass model information electronically between the engineering processes during product development. Consequently, to provide the information needed by the different processes, data must be entered several times: in one form as the conceptual design, another form for engineering drawings, another for finite element analysis, and another for numerical control. This is not only redundant, but also open to potentially inconsistent descriptions of the same object.

By producing one central model with all the information required by the various engineering processes, solid modeling

strives to unify product development into a coherent, and often automated, process. The complete solid model serves as the database source for all subsequent applications.

Informationally Complete Models

Writing for *IEEE Computer Graphics and Applications*,¹ Requicha and Voelcker define solid modeling as encompassing "an emerging body of theory, techniques, and systems focused on 'informationally complete' representations of solids—representations that permit (at least in principle) any well-defined geometrical property of any represented object to be calculated automatically." Solid modeling, then, is a combination of methods to represent object information in a way that enables applications to interpret and use the data without human intervention or interpretation.

But what constitutes an informationally complete representation of a solid? First of all, the representation guarantees a valid object. That is, its volume is totally enclosed; all of its faces and edges are defined; it is realizable. The representation includes the relationships among an object's faces: how they connect with one another, how they lie in three-dimensional space, how they are associated to enclose the object's volume.

"Informationally complete" also means that all nongeometric attributes of an object are specified, even though such information may be pertinent to only one or two of the processes. Such attributes could include the finish and color of an object's surface, the material(s) beneath the surface, the object's function, and many others.

In addition, a complete model automatically maintains consistency of object description in different representation schemes. Although the ideal solid model contains all the information needed by any application in the product development process, the model may have to be converted to another form to suit an application. Special data conversion routines in the solid modeling system "manage" the central solid model for the various application programs. By sharing the same model, all applications are accessing identical information. When modifications are made to an object, the changes are thus propagated through the central solid model to all representations used for all applications.

The benefits of an informationally complete solid model lie in the wide variety of automatic calculations to which the model lends itself. Mass properties, including weight, moment of inertia, center of gravity and volume, can be computed. Moving parts of an assembly can be checked for interference. Tool paths for numerical control applications can be determined. Meshes for finite element analysis can be generated. And, theoretically, all calculations can be accomplished without human involvement.

Defining the Solid Model


Because the solid model serves as the sole source of information for the separate engineering disciplines, its data structure must include the information used in all processes from design through manufacturing. Although a single data representation scheme has yet to be developed whose data structure satisfies all applications, two data representation schemes are clearly predominant: constructive solid geometry, and boundary representation. Two lesser schemes are gluing and sweeping. Another scheme is cell decomposition. Most solid modeling systems on the market today are hybrids of two or more data representation schemes.

Editors Note

A Special Report from the S. Klein Newsletter on Computer Graphics stated it succinctly: "Considerable confusion surrounds solid modeling."⁽¹⁾

One source of confusion is that many 3D CAD systems claim to be solid modelers, but are actually a combination of wireframe and surface modelers. A solid modeler is distinguished by the fact that its database contains information about an object's *interior*. A surface modeler displays objects by defining their "skins" as a collection of special elementary surfaces, often known as polygons or facets. All points of the object's surfaces are precisely known, which enables shaded surfaces to be produced quickly and easily; but information about the volume occupied by the object is missing.

As the Klein report goes on to say, solid modeling is also more than colorful, shaded-image displays: "Many people think solid modeling is merely a new way of making pretty pictures on a CRT," they state. "Often overlooked are other . . . benefits of solid modeling: simplified mass-property computation, interference checking, finite element analysis and Numerical Control (NC) tape preparation."

In this group of three articles, *Tekniques* attempts to dispel some of the confusion. The first article describes the differences in wireframe, surface and solid modeling, and offers basic terminology and concepts that provide a basis for evaluating solid modeling software and graphics devices. The second article discusses the role of the 4129 Color Graphics Workstation in displaying the output from host-driven solid modeling packages, including how the 4129 stores a 3D image and how it shades an object. The third article looks at an example of a solid modeling package whose results can be displayed on the 4129—OMNISOLIDS*, the solid modeling module for the ANVIL-4000* package from Manufacturing and Consulting Services. 

(1) "Solid Modeling in Computer Graphics: Tehnology, Applications, Supply Sources; A Special Report from The S. Klein Newsletter on Computer Graphics," Sudbury, Mass.: Technology and Business Communications, Inc., 1984.

Constructive Solid Geometry (referred to as CSG) uses a set of Boolean logic operations to build solid models from primitive, three-dimensional geometric shapes—cubes, cylinders, spheres, wedges, cones, truncated cones, and usually several complex forms. Each CSG primitive is a prototype form where certain characteristics specific to each type are left variable. For example, the cube-type primitive has variable height, length, width, and spatial location. The engineer produces an *instance* of a primitive by specifying all variable characteristics. Thus, each primitive carries a set of parameters defining its location, orientation, and size. Attributes of material, function, etc., can easily be included with the geometric information.

The set operations include *union*, which combines two primitives; *difference*, which subtracts one primitive from another; and *intersection*, which defines a volume shared by both primitives (Figure 3). Using successive operations, internalized in the computer as a binary tree (Figure 4), the engineer creates complex models. Recording each step of the design creation in a binary tree permits quick editing of an object: simply eliminate one or more of the steps. Because CSG constructions

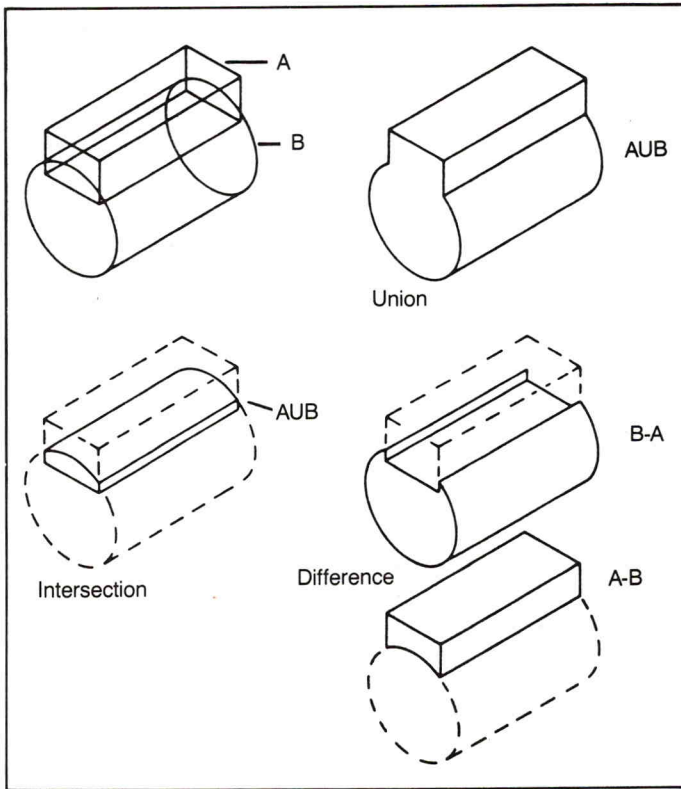


Figure 3. CSG object operators and their functions.

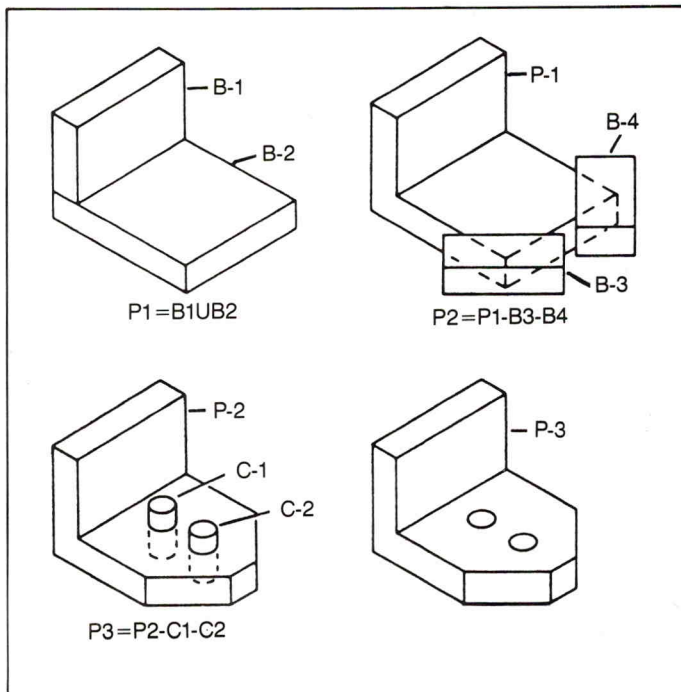


Figure 4a. A sample part description sequence.

use complete three-dimensional primitives, a CSG-constructed model always results in a valid object.

CSG can define the shapes comprising a large percentage of real-world part designs. These are the common shapes produced by machining operations such as cutting, drilling, milling, and so forth. CSG is also a natural method to accommodate the way machining operations occur. For instance, to "drill" a hole in a cube during CSG construction, a cylinder the size of the hole

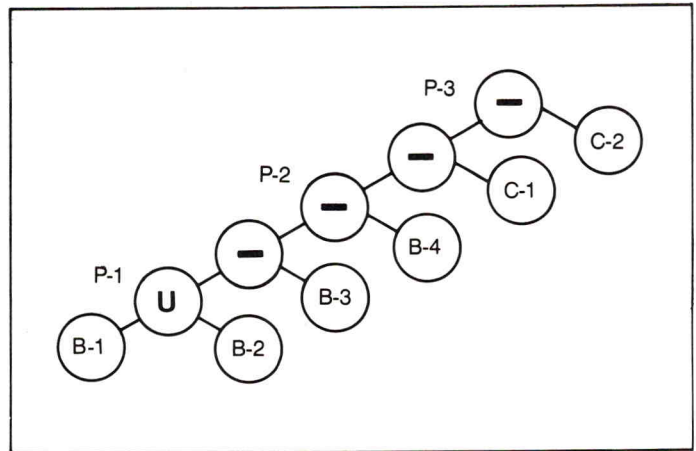


Figure 4b. A binary tree for the part shown in Figure 4a.

is subtracted from the cube. This reflects the actual drilling operation, which subtracts a specified amount of material from a part.

The main drawbacks in CSG construction are its lack of flexibility in surface description, and the intensive computation required to convert a CSG solid model to a boundary representation model required by certain applications.

Boundary representation (B-rep) is, perhaps, a more natural method for defining shapes. Faces, edges, and vertices compose the building blocks of B-rep, and their topology and connectivity define the solid (Figure 5). Special operators allow bending, stretching, or blending of the surfaces. Parameters are used to define in which plane a face lies, and which is its front and back. Attributes such as surface finish can easily be attached to each of the faces in a B-rep model. And once an object is defined through B-rep, set operations can be performed on it as if it were one of the CSG primitives.

Since B-rep is not restricted to certain primitive shapes, it can produce a wider variety of objects than CSG. Multicurved surfaces, known as sculptured or free-form surfaces, are more easily represented using the B-rep scheme. These surfaces include objects such as automobile fenders, airplane wings, propeller blades, ship hulls, and ornamental objects.

The major advantage of the B-rep scheme is that the faces, edges, and vertices of an object are represented explicitly. Since applications such as graphic rendering and numerical control programming are concerned with explicit boundaries, this scheme has real appeal. B-rep schemes, like CSG, have certain limitations, however. Because its primitives are not complete shapes, B-rep construction doesn't necessarily produce a valid object. Therefore, such solid modeling systems must check the validity of object's geometry—that faces intersect only at edges, that a volume is completely enclosed—or depend on the engineer to do it. When Boolean operators are used to combine two boundary representations into a third, determining the intersection of the nonplanar surfaces of these free-form objects is often slow, and possibly inexact. And since B-rep construction lacks information about the order in which a model is created, reversing certain operations can place a heavy burden on the user.

Gluing, sweeping, and cell decomposition are other methods for defining solid shapes.

Gluing joins two previously created solids at a common surface (Figure 6). *Sweeping* moves a two-dimensional surface

through space along a path to "sweep out" a solid. For example, to create a part with rotational symmetry, the two-dimensional surface is rotated about an axis (Figure 7). Gluing and sweeping operations are easy to use, but the number of shapes they can define is limited. Both are useful as a subset of CSG or B-rep schemes.

Cell decomposition represents solid objects using a restricted class of "simple" solid objects, called *cells*, and combines these cells by gluing. The cells may form an object by sharing boundaries—vertices, edges, and/or faces—but objects cannot be formed by cells sharing volumes. (The CSG method shares volumes.) Two variations exist for cell decomposition: voxels and tricubic hyperpatch cells.

Voxels are cubic volume elements that can only approximate the desired object, although any degree of precision can be achieved with sufficiently small elements (Figure 8). Because all surface features of an object are represented uniformly and only approximately, information as to whether a surface is cylindrical (important in numerical control programming) is not exact.

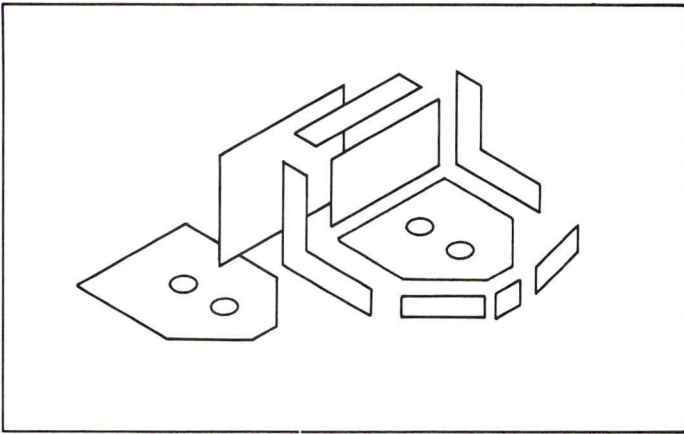


Figure 5. B-rep building blocks.

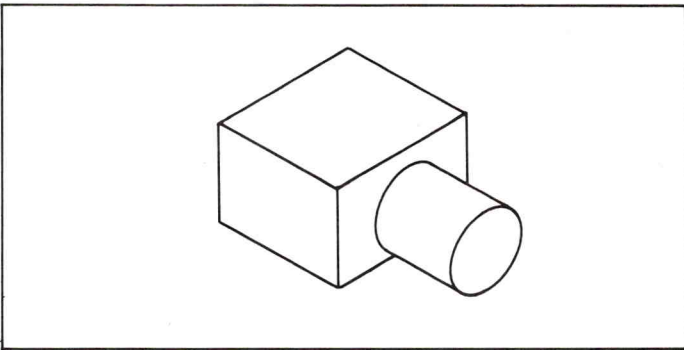


Figure 6. Gluing joins block and cylinder.

Tricubic hyperpatch cells permit accuracy with curved surfaces; however, the user is generally responsible for decomposing the conceptual shape into the cells required to define it. Because a tricubic hyperpatch is topologically equivalent to a cube (six faces), the user must decompose the shape into solids that are topological cubes (Figure 9). Certain modifications are difficult with tricubic hyperpatches.

Regardless of the data representation scheme or schemes used, the solid modeling interface prompts the engineer for the data needed and translates the engineer's entries into an internal

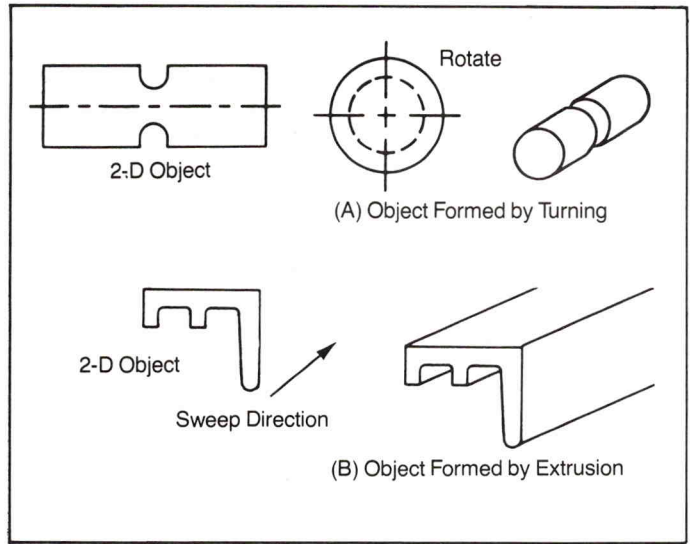


Figure 7. Examples of objects naturally describable as swept volumes.

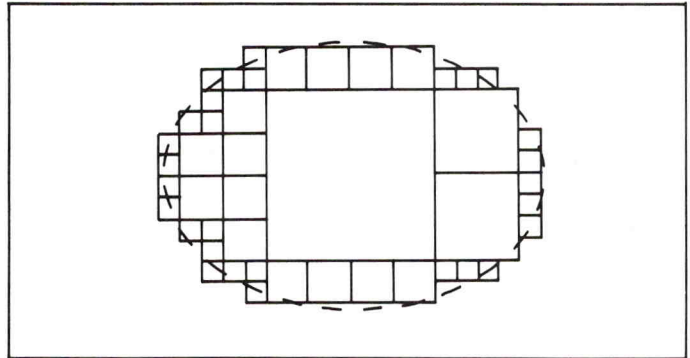


Figure 8. A solid object approximated with volume elements (voxels).

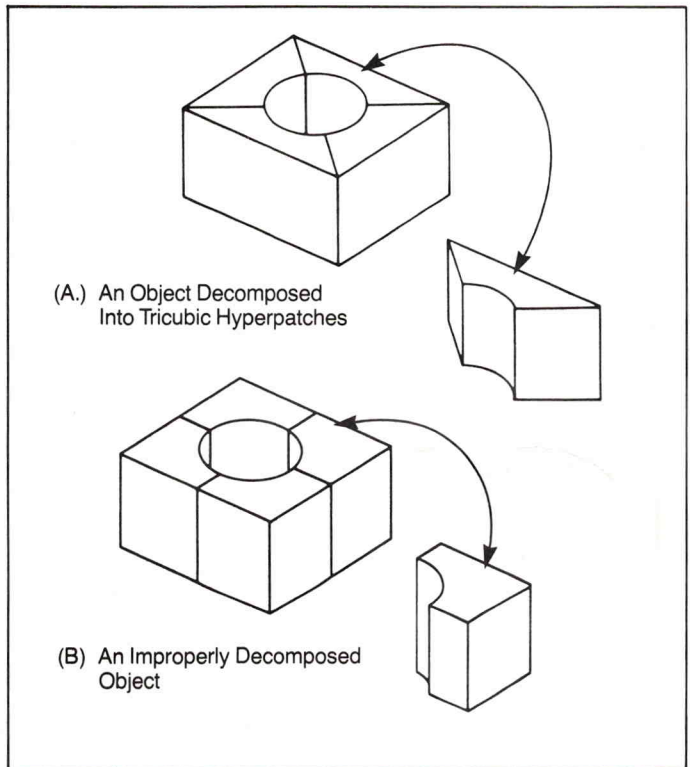


Figure 9. Tricubic hyperpatch cells approximating an object must be topologically equivalent to a cube.

representation that is a complete solid model. The burden of how the object is organized and its mathematics are handled in the software, not placed on the engineer. The engineer is free to conceptualize, create, analyze, and modify the object.

The Graphics Interface

Most of the solid modeling systems on the market today depend on a graphics interface to communicate with the engineer. And almost all of these graphics interfaces use color-shaded, three-dimensional graphics.

In the conceptual stage, the engineer selects geometric elements from a database, or creates them on the screen to bring his concept to life. Graphics provide immediate feedback, helping the engineer visualize the object. Realistic images depict the object's aesthetic appeal and reveal its flaws or design inconsistencies. Color defines relationships between object assemblies, piping layouts, and other types of elements.

During finite element analysis interpretation, color shading highlights the stress points of an object. Reviewing the shaded image, the engineer can readily isolate areas of the design that need refining.

Interference studies call upon graphics to provide realistic displays of the moving parts of an object throughout the sequence of its possible configurations. Color clearly identifies the various components of the moving system.

Numerical control applications display the programmed tool paths with color defining the individual paths.


To facilitate interaction with the user, most graphic interfaces have built-in controls that enhance the displays. Zooming and panning select certain areas of the image for expanded display. Multiple viewports on the screen permit comparison of differing views of an object, or collections of different data. Sectioning cuts parts of an object away to reveal its interior.

Illumination helps show off surface properties of an object. Different projection types permit perspective or parallel views. Wireframe, hidden line, or color-shaded renderings can be chosen to fit the application.

How such displays are produced from the solid model depends on the data representation scheme used and the graphics device being used. For the most part, data structures are designed to model solid objects mathematically rather than to produce images rapidly. Therefore, model conversion is almost always required.

A B-rep description lends itself to being approximated by planar polygons and can be easily displayed. A CSG description can be converted to B-rep, then displayed as planar polygons. A direct graphical display method for CSG objects, known as ray tracing, is becoming viable. And hardware enhancements of the display devices such as depth buffers and special shading processors are facilitating display of solid objects.

Conclusion

The full realization of what solid modeling promises is yet to come. Currently, all applications in product development cannot secure the needed object information from one central solid model. But as research improves the solid modeling architecture, that single-source goal becomes closer. Accompanying the solid modeling developments will be the graphic display system technology that will permit display devices to work with the solid modeling data structure to streamline the object's portrayal on the screen. 

¹A.A.G. Requicha and H.B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment," *IEEE Computer Graphics and Applications*, March, 1982.

Innovative display techniques of the 4129 Color Graphics Workstation enhance the graphic link between host and engineer in solid modeling systems.

4129's Viewing System Produces Realistic Displays for Solid Modeling Programs

As solid modeling unifies CAD/CAM, realistic graphics displays are playing a critical role in helping the engineer visualize complex geometries. The Tektronix 4129 Color Graphics Workstation offers rapid display of color-shaded, three-dimensional surfaces to depict the realism needed by solid modeling systems.

Database Holds Key

The complexity of the geometric data influences how an object is portrayed. In general, objects rendered on a graphics screen take four forms: wireframe, wireframe with hidden line suppression, shaded surface, and sectioned solid. A good way to sort out the different forms is by examining the underlying

database formats of the models which use them.

In many existing CAD systems, objects are represented internally as a sequence of 3D coordinates defining the end points of lines. The lines represent the edges of the object. Each edge is essentially one-dimensional—it has no width or depth, only length. When the edges making up the image are projected and transformed on a screen, it gives the illusion of the manipulation of a complete object. Yet there is no information in the database about the object's surface or interior. Lines cannot be suppressed, and shaded surfaces cannot be portrayed. Thus, the only display form the object can take is wireframe—that is, edge rendition.

Additional information in the database can define a *3D surface model*. Such information includes the 3D coordinates, plus how surfaces connect to edges, how edges are related, and so forth. In addition to a wireframe display of the object, the surface description permits wireframe with hidden-line suppression and shaded surface renderings of the object. Although the display may seem realistic, however, the underlying database is still not complete. What happens, for instance, when the object is sectioned?

A *solid model database* contains information about the object's shape, edges, surfaces, and *interior*. This means that given any point in space, a solid modeler can determine if the point is part of an object or not. Even advanced surface modelers lack this capability. A solid model, therefore, can be rendered as wireframe, wireframe with hidden lines suppressed, shaded surface, or sectioned to reveal its true interior.

Realism vs. Computing Time

When would a solid modeling system use each of the four types of graphic renderings? Wireframe is a common display technique since it takes less time to generate and display than the other techniques. Each successive form of graphic rendition requires more computing time and slows down the graphic feedback to the designer. Another reason for using wireframe might be to display all the components of a CSG constructed object. (A related article in this issue of *Tekniques* discusses the different solid modeling methods.)

Wireframe with hidden lines suppressed provides a compromise between very realistic objects and computing time. It reduces the display complexity of an object without the attendant time required to shade the object's surfaces.

Too often, however, real objects consist of many assembled parts constructed from hundreds of individual graphic primitives, and even hidden line suppression doesn't remove the clutter. Color-shaded surface display of these objects helps the engineer visualize and distinguish the contours and profiles of the object. Sectioning, of course, is used to see into an object. It cuts away portions of the object to open its interior to view.

The Tektronix 4129 Color Graphics Workstation takes over the computing burden from the host to display objects as wireframe, wireframe with hidden line suppression, and color-shaded surfaces. Although the workstation does not retain information about an object's interior, it supports sectioning and translucency display through special commands. A local viewing transform system in the workstation gives the engineer control over how an object is displayed on the screen. Using the thumbwheels and keyboard keys, the engineer can view the object from any angle and control its display form.

In this article, we'll look at how the 4129 stores a 3D object definition and how it shades an object. The next issue of *Tekniques* will examine the 4129's Local Viewing System and how the user controls his view of a 3D object.

Storing an Image in the 4129

Segments

For the 4129 workstation to perform the calculations needed to shade or manipulate an object, the object must be stored in the workstation's memory. To do this, the host assigns the graphics primitives that define the object to a *segment*, which the workstation retains in memory. When the host sends in-

structions to the terminal to manipulate the image, it simply identifies the segment, rather than sending the complete collection of graphic primitives again. (Although this article is directed at the display of three-dimensional objects, the 4129 workstation also stores two-dimensional images as segments and manipulates them. In fact, both 2D and 3D segments can be displayed on the workstation's screen at the same time.)

Facets

For an object to be shaded, the information defining its surface must be sent to the 4129 workstation. Most graphics interfaces of solid modeling systems extract the shape of an object from the central solid model and approximate it with polygons (Figure 1). To accommodate the polygon approximation scheme, the 4129 accepts a special graphic primitive known as a *facet*, which is a special case of a polygon. Facets can reduce the amount of data the host program must send to define an object by up to 40%.

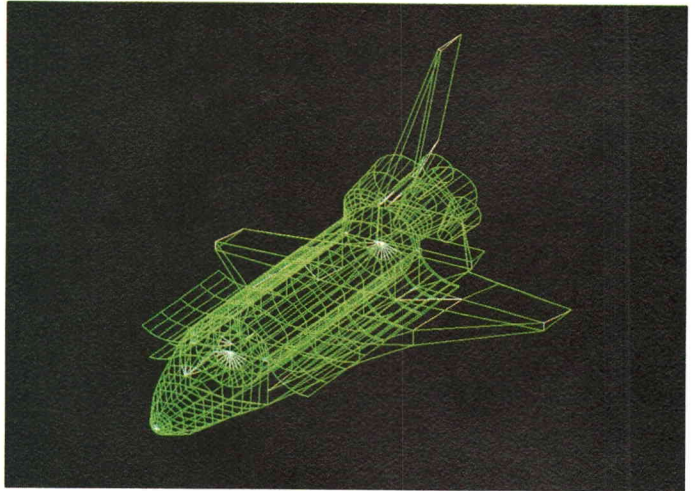


Figure 1. The object's shape can be approximated with facets.

As Tektronix uses the term, a facet is a triangle (three sides) or a quadrilateral (four sides), consisting of vertices and edges. A vertex is a point where two edges join; an edge is simply a line ending at two vertices.

As Figure 2 illustrates, five types of facets are available on the 4129 workstation. *Triangle lists* and *quadrilateral lists* let the host define many polygons with a minimum number of commands. *Triangle spokes* provide a convenient way to specify a sequence of triangles with a common "center" vertex. A *triangle strip* consists of a list of triangles in which the last edge of each one becomes the first edge of the next triangle. A *quadrilateral mesh* is an array of vertices with common edges. Spokes, strips, and meshes allow the host program to send fewer data points to define an object's surface, thus saving time and increasing performance. The host software determines which type of facets are used to best represent an object's surface.

Normal Vectors

To perform local shading, the 4129 must be able to determine which direction each facet is facing and how light is reflected. This is accomplished through *normal vectors*. Normal vectors are simply direction lines (known as vectors) perpendicular to (also called normal to) the piece of surface represented by each facet. Normal vectors can be specified in two ways: the normal vector to an entire facet surface, or the normal vector to

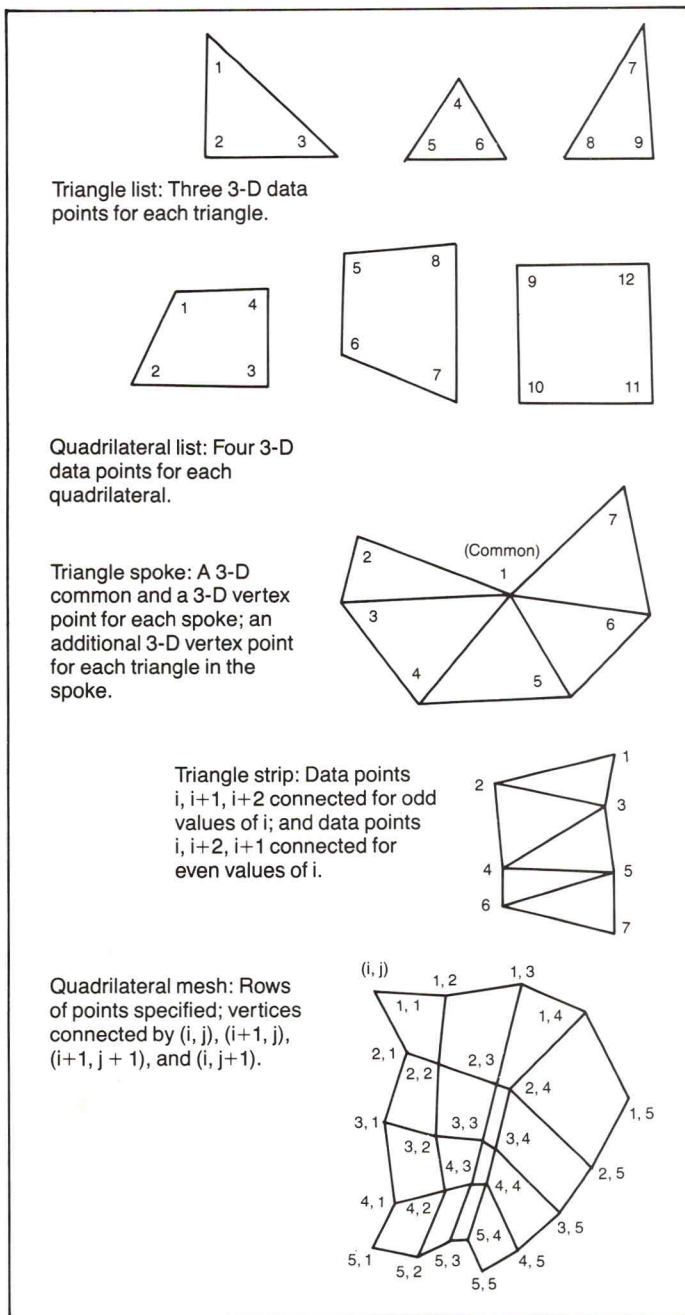


Figure 2. Five types of facets are available on the 4129. Host software determines which types of facets are used to best represent an object's surface.

each vertex of a facet (Figure 3). The vertex normal vectors should be perpendicular to the surface represented by the facets, not to the facet itself. Moreover, these normal vectors should coincide at vertices common to more than one facet, referred to as average normals.

Producing Shaded Surfaces

The 4129 workstation supports three methods for shading surfaces: constant, cosine, and Gouraud shading.

From a programmer's point of view, the simplest method is constant shading. The host program defines a color for an object, sends the facets that approximate the object, and lets the 4129 fill each facet with the same color. This allows removal of hidden surfaces and is useful for rapidly shading a three-

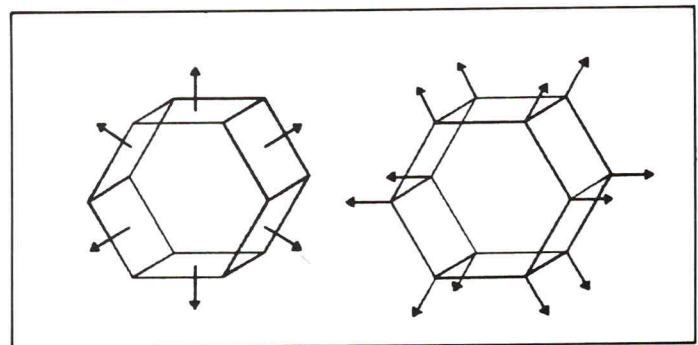


Figure 3. A vector can be normal to an entire facet (3a) or to each vertex of a facet (3b).

dimensional object in the development stage. However, since the entire object appears as exactly one color, without highlights, the object lacks realism.

With Cosine and Gouraud shading, the 4129's local lighting model can be used. The local lighting model takes into consideration the intensity of directed light sources (up to 16), and the reflectance capacity of the object (less reflectance for an orange than a ball bearing, for example).

Cosine and Gouraud shading are determined by the number of normal vectors used. For cosine shading a single surface normal for each facet is sent from the host. Each facet will be shaded with an even color (Figure 4). This is a reasonable approach to shading objects with flat surfaces, such as cubes or pyramids. For smoothly curved objects, however, the boundaries separating each facet will be clearly visible.

A better approach is for the host to send a color index or a normal vector for each vertex of the facet. This is known as Gouraud shading. When vertex normal vectors are sent, the 4129 calculates colors at each vertex of a facet and "linearly interpolates" colors across the facet (Figure 4). This provides smoother transitions in color and makes the faceted representation appear smooth, like the object it represents.

The three types of shading are most easily summarized by looking at the color of a single facet and the effect of the local lighting model. For constant shading, a facet is a single color across the surface and is not altered by the local lighting model. In cosine shading, the facet is again entirely one color, but the color varies and is affected by the local lighting model. Gouraud shading linearly interpolates color across the facet surface. Where normal vectors are provided in Gouraud shading, the lighting model can affect a single vertex or the entire surface of a facet to produce very smooth color transitions.

The colors used to shade an object are defined in a contiguous set of color map indices that represent a range of intensities. To shade an object with varying greens, for example, the range could run from blue-green to yellow-green hues. To shade an object with a single red but varying lightnesses, the range could run from almost black-red to a light pink. After calculating light intensities, the 4129's local lighting model extracts the appropriate color from the predefined color range for each normal vector.

An image is independent of the local lighting model, which means that the 4129 applies the local lighting model each time a segment is redrawn. The engineer thus can transform the segment or modify the direction or intensity of the light source

to see how an object reflects light under varying conditions or at different angles.

Although the local lighting model can save the host much computation time, it may not always be desirable to use. For applications that wish to portray shadows, mirror reflections, or some other special effect, the host can send a color for each vertex of the facet, and let the 4129 perform the shading for a smooth transition of colors across each facet. The host can also define the constant color for each individual facet.

Once the colors for the vertices of each facet are determined (either through the local lighting model or from the host), the 4129 calculates the color of each screen pixel. It then places the pixel data in the workstation's "Z-buffer"—a special depth buffer—where hidden-surface removal is performed by determining which pixel is "in front" of another with reference to the viewer.

Once this operation is complete, the pixels may be dithered for even smoother shading. Depending upon the location of a pixel, its color may be modified in order to minimize the Mach banding effect (where the edges of adjacent facets are perceived to be brighter than the centers of the facets) or color banding (where the facet has distinct bands of color).

Revealing the Object's Interior

One or two sectioning planes, oriented in any direction, can be used to "cut away" sections of the objects or, alternatively, to change the translucency for the sectioned piece of the object (Figure 5).

The effectiveness of sectioning and translucency depends on the host program. Remember that facets are sent to the workstation to approximate the object's *surface*. The 4129 has no knowledge of the interior of the object; that information remains with the host. The host should define the interior surfaces of the sectioned object and send them to the terminal.

A translucency pattern is used to determine which pixels are shaded and which are left unshaded. The unsectioned portion of an image can be displayed using a translucency pattern, while the sectioned portion is transparent. Or the sectioned portion can be displayed with the translucent pattern, while the unsectioned portion is opaque. The 4129 provides 16 predefined translucency patterns.

Normally, sectioning and translucency operations require a great deal of host computation. With relatively simple commands, however, the host program can cause the 4129 to accomplish those tasks.

Viewing an Object

In addition to removing the computational burden from the host for shading images, the 4129's viewing system also handles the calculations needed to transform an object, or to let the engineer "move around" an object.

Using just a few commands, the host can cause the 4129 to perform modeling transforms or viewing transforms on any object stored as a graphic segment. Most of the 3D transforms a program would ever require are available, including rotation, scaling, clipping, and skewing. A single command controls the three-dimensional space that can be viewed.

One of the most useful features for the engineer, perhaps, is the 4129's *Local Viewing System*, which lets the operator inter-

actively control his view of an object, the type of viewing projection, and the type of graphic rendering or shading. The Local Viewing System essentially gives the operator control over the orientation, size, and shape of a 3D "viewing volume." In Part 2 of this series, we'll see how the Local Viewing System works.



Figure 4. In constant and cosine shading (left), banding is visible. The Gouraud shading (right) generates smooth transitions of colors across an object's surface.

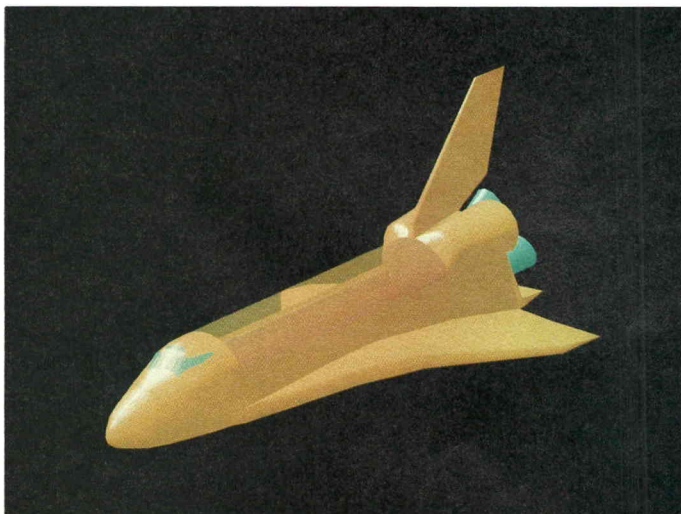


Figure 5. The shuttle bay doors are made translucent, allowing the interior to be seen. Note that the left wing now becomes visible through the translucent doors.

Using sophisticated construction techniques and intelligent graphics devices such as the 4129 workstation, ANVIL-4000 and OMNISOLIDS* permit automation of the CAD/CAM process from concept through machined part.*

Solution Profile: OMNISOLIDS Unifies the Product Development Process

Solid modeling. How does it really work? Related articles in this issue of *Tekniques* review the concepts of solid modeling and show how graphics display hardware supports the solid modeling effort. This article describes how the solid modeler component of one CAD/CAM software package integrates the engineering processes.

Solid Modeling with ANVIL-4000

At Manufacturing and Consulting Services, Inc. (MCS), Dr. Patrick J. Hanratty, president, affirms that the current generation of CAD/CAM systems is distinguished by having a solid modeler as an integral part of the system. And he notes that the ANVIL-4000 series of CAD/CAM software produced by MCS incorporates a state-of-the-art solid modeler known as OMNISOLIDS. OMNISOLIDS produces a true geometric model that all engineering disciplines can use, permitting automation of the CAD/CAM process from concept through machined part. Over 70 different construction techniques, ranging from the most elementary to the most advanced, are provided in the CSG/boundary hybrid solid modeler.

Realizing that no single CAD/CAM system could economically meet the diverse requirements of all industry segments involved in design and manufacturing, Hanratty and his design team developed ANVIL-4000 as a modular, extensible yet integrated package. Brought to market in 1981, ANVIL-4000 allows users to select the modules that fit the requirements of their operation at the time, then add other modules as their manufacturing operation grows. ANVIL-4000 runs on several 32-bit virtual computers and depends on intelligent graphics display devices such as Tek's 4129 Color Graphics Workstation to communicate with the user.

In its latest evolution, ANVIL-4000 incorporates the OMNISOLIDS module into its family, extending the designer's tools to handle solid objects.

Sophisticated Techniques

The "surface-enclosed solid" shown in Figures 1 and 2 illustrates the levels of sophistication and capability that OMNISOLIDS brings to solid modeling. Six different surfaces were used to create this very gnarled and knobby cube: ruled surface, generalized surface of revolution, tabulated cylinder, curve mesh surface, rational B-surface, and a Bezier surface. Figure 1 shows these original surfaces prior to forming the solid. Figure 2 shows the final solid—the cube with its faces trimmed, in three different display types: a) wireframe, b) wireframe image with hidden surfaces removed, and c) shaded, highlighted image.

The hole, which wanders through the solid like a worm hole,

was generated by subtracting (that is, using the Boolean "difference" operation) a solid formed by a "curve-driven surface" from the gnarled cube. Of particular interest, states Hanratty, is that the completed solid—six different warped surfaces enclosing a volume with a "worm hole" subtracted—is still a primitive to OMNISOLIDS. This shape can be used with the same flexibility as a cube, a cylinder, a wedge, or a cone. It can be combined with other primitives using the Boolean set operations of union, difference, and intersection. It can also be analyzed, meshed, annotated, and machined, all interactively, and all in ANVIL-4000.

Realistic Graphics

OMNISOLIDS uses the intelligence of the Tektronix 4129 workstation and its high-resolution color graphics to communicate effectively with the engineer. Both screen and tablet menus may be used. Screen menus are stored as segments on the 4129 to reduce transmission from the host. When the engineer requests a menu, it is instantaneously displayed.

By storing an image as facets in the 4129 workstation's memory, OMNISOLIDS gives the engineer local control over image display. The engineer can use all of the 4129's three-dimensional viewing functions to "walk around" an object, move through the object, enlarge it, shrink it, or change its position. OMNISOLIDS also lets the engineer modify the light sources, then use the 4129's local lighting model and viewing functions to capture the varying reflectances of an object at different angles. Although the 4129 does all of the shading, the engineer has control over the type of shading—constant, shading without dithering, or shading with dithering.

All of the shaded figures in this article were generated on the Tektronix 4129 workstation. The knobby cube uses OMNISOLIDS' lighting model and software shading; all others use the 4129's local lighting model and shading.

Creating a Solid Model

To illustrate the power of OMNISOLIDS, we'll follow a typical design-through-manufacture session. In this example, OMNISOLIDS is used to construct an aircraft wing, its supporting ribs, and the fuel and hydraulic lines which pass through it. After defining the model's geometry, one of the ribs is used to demonstrate finite element meshing, dimensioning, and numerical control tool path generation.

First, the shape of the wing, its skin and thickness are defined using typical engineering methods (Figures 3 and 4). Next, the wing's supporting rib is circumscribed by the region from a

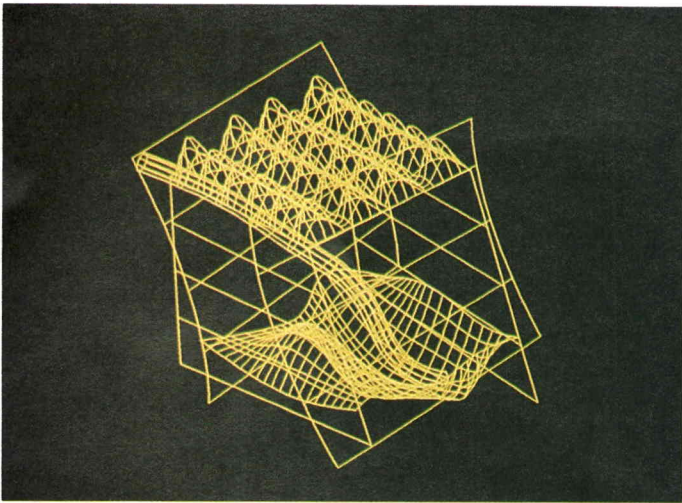


Figure 1. Six different types of surfaces are just a few elements in the OMNISOLIDS' tool kit.

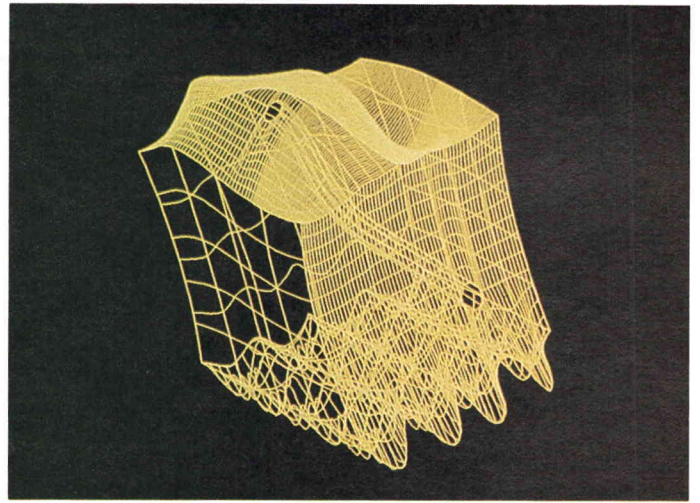


Figure 2. A gnarled cube comprised of six different surfaces represents the robustness of the "surface-enclosed solid."
a. wireframe

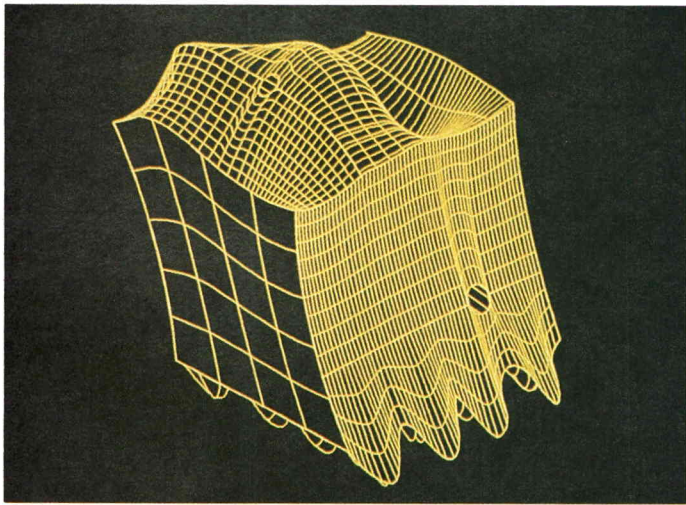


Figure 2b. Wireframe with hidden lines suppressed.

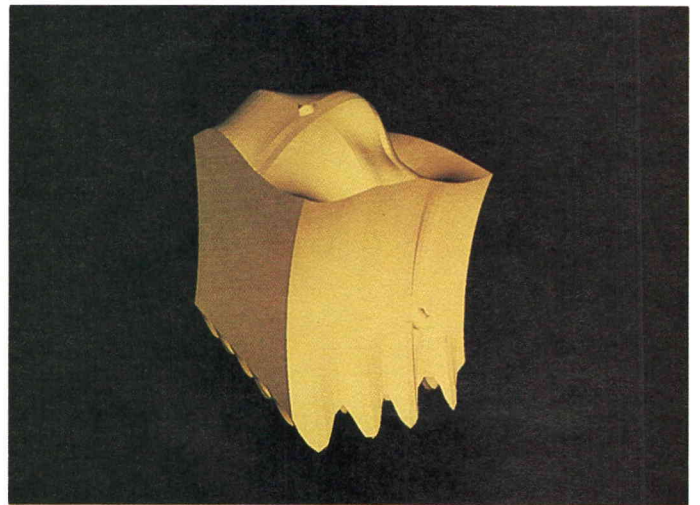


Figure 2c. Shaded surface.

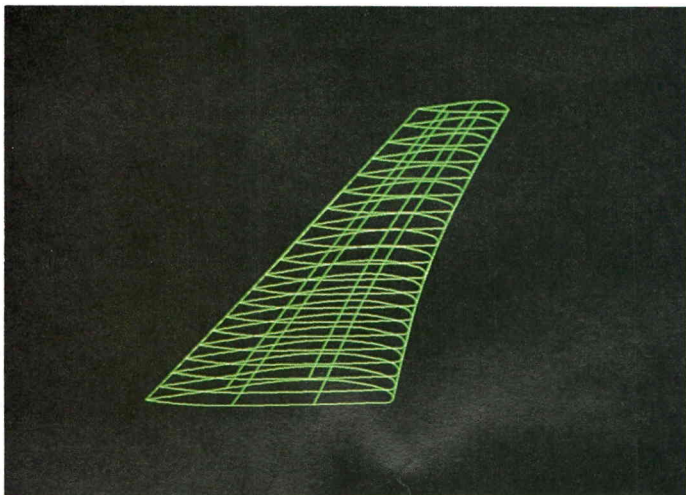


Figure 3. A "curve mesh surface" (a modified Coon's patch surface) defines the shape of the airfoil.

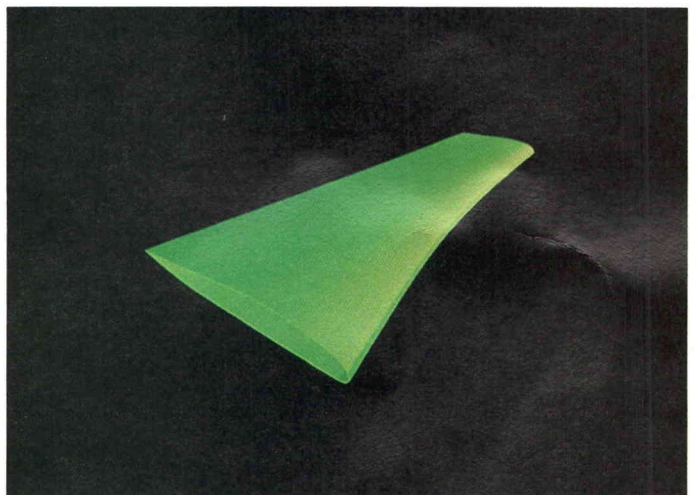



Figure 4. The skin of the wing is formed using the "warped surface normal offset" construction methods in which a solid is defined by the region from a base surface to a specified offset distance. To give the skin a thickness, the designer specifies an offset direction and offset distance from the airfoil.

“slab” (a set of closed planar curves) to its projection onto the airfoil surface (Figures 5 and 6). Primitive shapes and Boolean “not” operations cut six holes through the rib (Figures 7 and 8). The rest of the wing’s ribs are constructed in the same manner to complete the wing’s support structure (Figure 9).

To visualize the fuel and hydraulic lines that pass through the wing, curves are constructed that represent the center lines of the pipes and tubes; these center lines are then expanded to the pipe or tube outlines (Figure 10). The entire wing, with skin, ribs, and fuel/hydraulic components is still considered a primitive by OMNISOLIDS.

With the basic wing geometry complete, the next step in the design process is to evaluate the part’s structural integrity by generating a finite element mesh (Figure 11) and passing the mesh to a finite element analysis program such as ANSYS® or MCS NASTRAN™.

Dimensioning is performed in the ANVIL-4000 drafting module by selecting vertices on the solid (Figure 12). And the ANVIL-4000 NC module machines the surface geometry which is inherent in the solid model (Figure 13).

The versatility and flexibility of OMNISOLIDS is evident in this design sequence. One central geometric model was defined by OMNISOLIDS. The model was then automatically accessed by other ANVIL-4000 modules to perform the engineering processes required in the wing’s development. In addition, the shape information was passed to a non-ANVIL-4000 program without human intervention. All operations used one model, ensuring that the complete mathematics of the solid were used throughout the design, analysis, drafting and manufacturing processes. 

*ANVIL-4000 and OMNISOLIDS are trademarks of Manufacturing and Consulting Services. ANSYS is a registered trademark of Swanson Analysis Systems. MSC/NASTRAN is a trademark of the MacNeal-Schwendler Corp.

All photographs copyright by MCS.

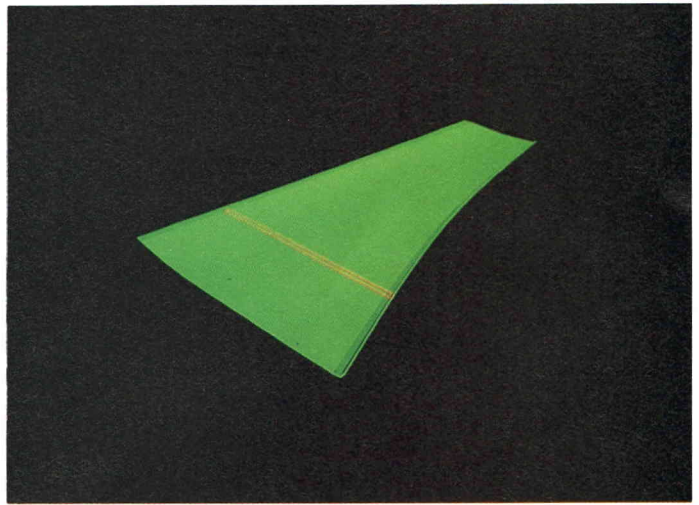


Figure 5. The engineer selects a quadrilateral that represents a cross-section of the rib through the wing chord.

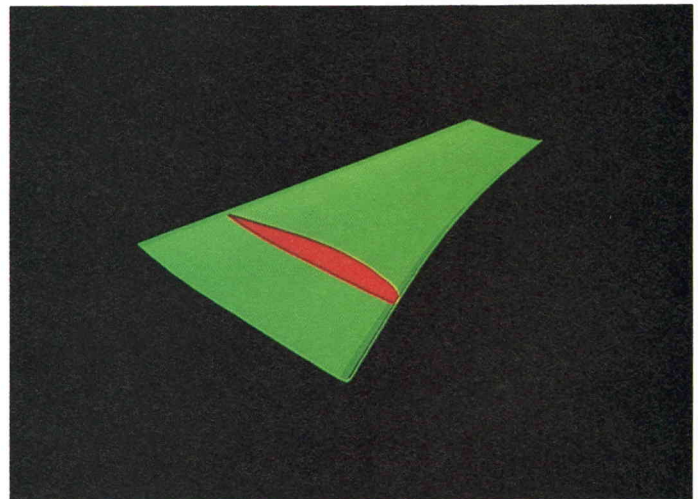


Figure 6. Using the “slab projected to a surface” construction method, the engineer projects the cross-section to the airfoil surface to generate the solid rib.

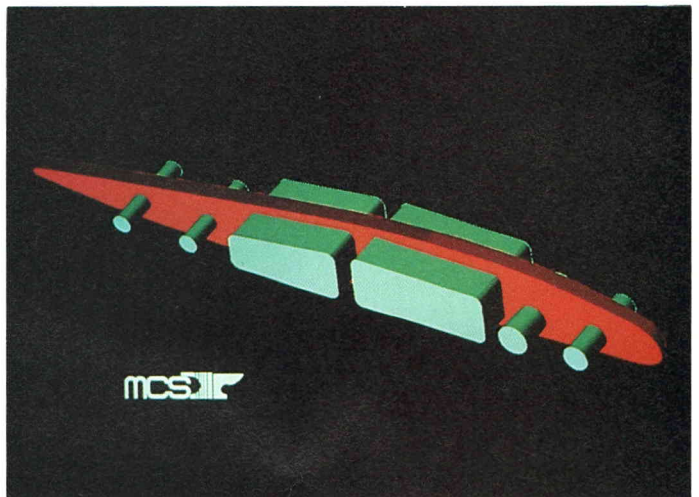


Figure 7. Two axial sweeps (linear projections of sets of curves) define the two non-circular shapes that are positioned on the rib, along with cylinders of the appropriate radius. Using the Boolean “not” operation (meaning the rib without the shapes), the engineer subtracts the six shapes, producing the holes in the rib.

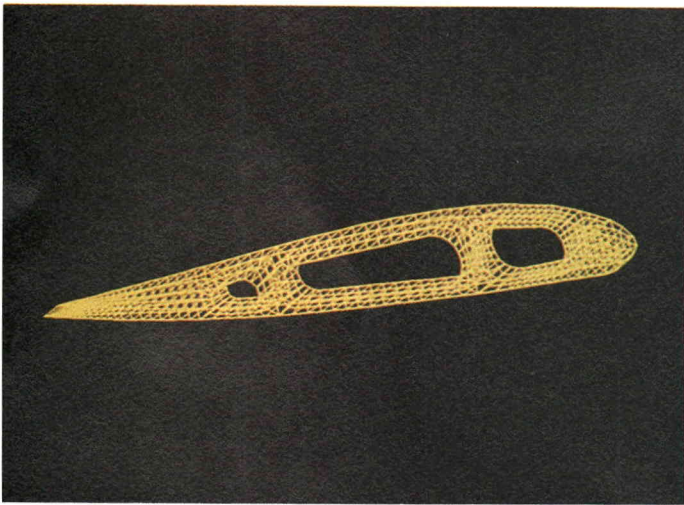


Figure 8. The rib with the holes cut through it.

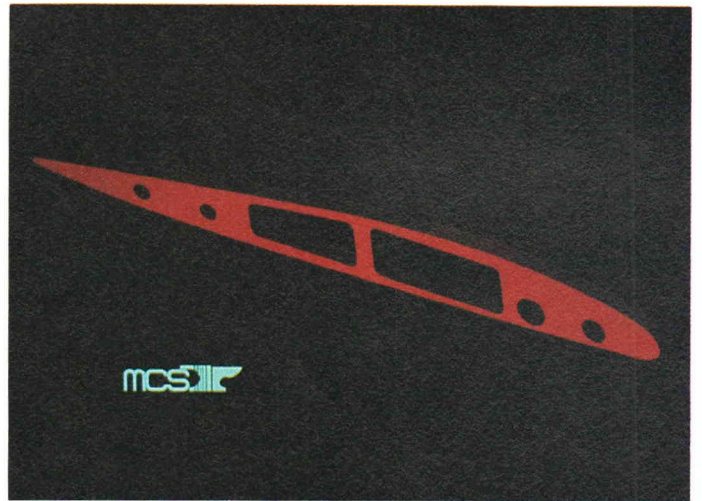


Figure 11. The ANVIL-4000 automatic mesh generator quickly generates a 3-D finite element mesh on the rib.

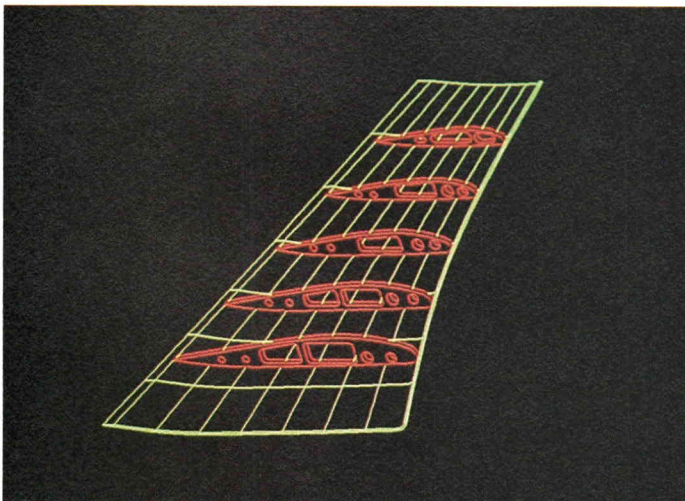


Figure 9. Repeating the processes shown in Figures 5 through 7 results in the wing's supporting structure.

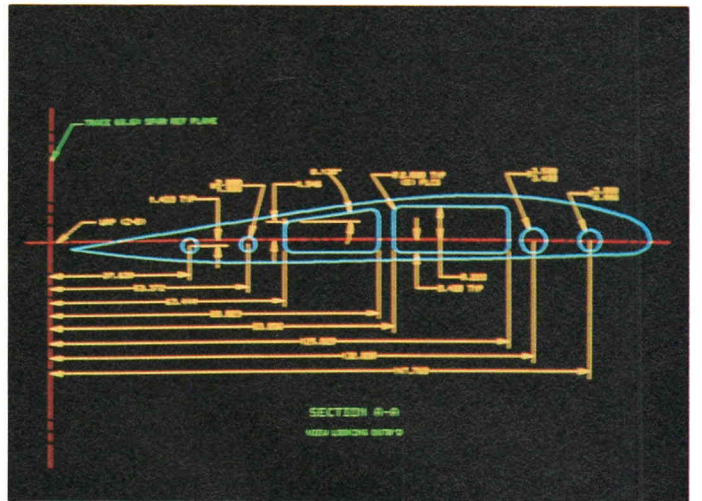


Figure 12. To annotate the drawing, the engineer "blanks" all of the solid except one of the ribs, and allows the ANVIL-4000 drafting module to generate the appropriate dimensions. He can then add notes and labels.

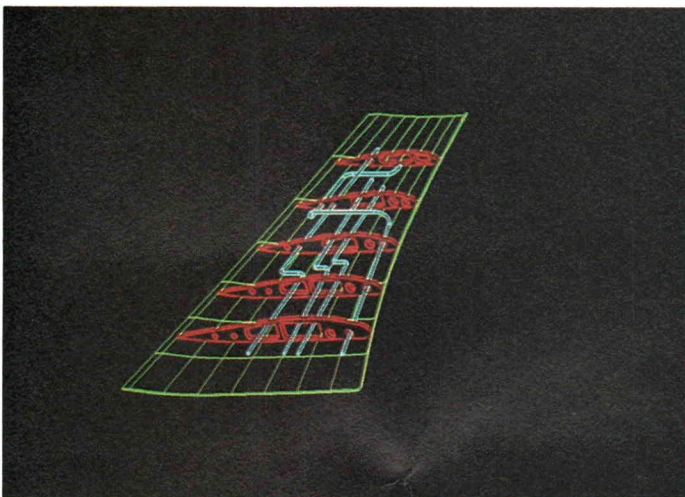


Figure 10. To expand center lines to circles which represent piping or tubing, the engineer uses the "curve-driven solid" construction method, specifying a radius along the center-line curves.

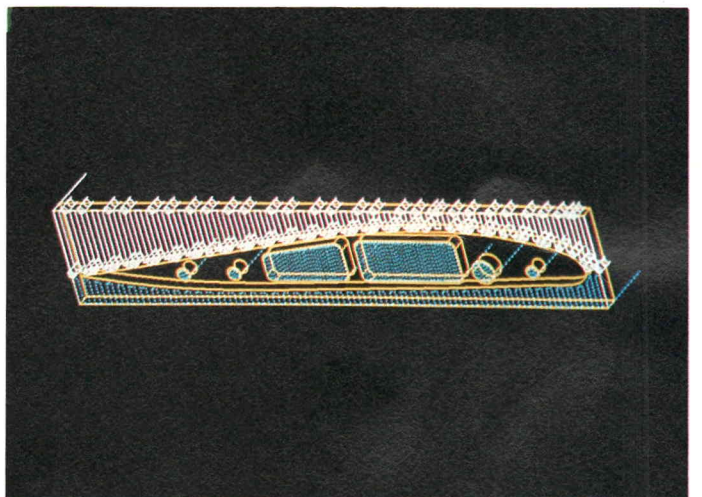


Figure 13. The solid rib is cut from a block of material using the numerical control function called "area clearance." The numerical control "pocketing" function removes the circular and non-circular holes.

Smalltalk, which comes standard on the 4400 Series AI Systems, offers a highly interactive, efficient, object-oriented programming environment.

Fifth-Generation Programming with Smalltalk-80*

by Juanita Ewing
Artificial Intelligence Machines
Tektronix, Inc.
Wilsonville, OR

With the high-speed Smalltalk implementation on its 4400 Series Artificial Intelligence (AI) Systems, Tek has brought the Smalltalk-80 system out of the research labs and into the hands of a wide variety of users. Originally developed at the Xerox Palo Alto Research Center, Smalltalk is an extensible, flexible system that lends itself to use in a number of application areas, including artificial intelligence and rapid prototyping. As a fifth-generation language, Smalltalk provides a fertile ground for development and increases programmer productivity.

An Object-Oriented Language

Smalltalk is different from most other programming languages—different enough that Byte magazine once devoted an entire issue (June, 1981) to explaining Smalltalk. When we speak of an ordinary computer language, we are generally talking about just a language. Sometimes the discussion goes so far as to refer to runtime libraries. When we refer to the Smalltalk system, though, we are talking about both a language and an environment. The syntax of the Smalltalk language is simple, and the language is small and readable. Smalltalk's environment, however, is extensive. The environment can be thought of as a huge collection of libraries and software whose purpose is to make programming in Smalltalk very convenient.

Fifth-generation languages embrace new paradigms such as functional programming, logic programming and *object-oriented* programming. Smalltalk is an object-oriented language. Objects are a way of encapsulating information. Object orientation means that everything in the environment is an object. Data is represented by objects, Smalltalk source code is made of objects, and an executing program is composed of objects.

In Smalltalk, objects send *messages* to each other; receivers of messages take some independent action to accomplish

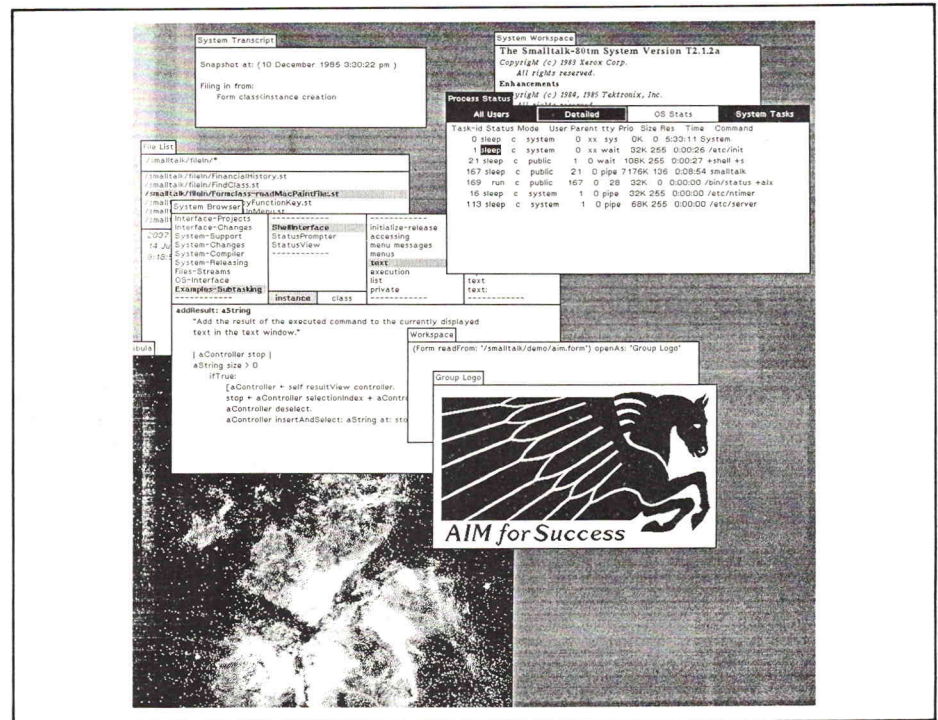


Figure 1. A typical Smalltalk screen consists of a number of windows, each containing text and/or graphics.

some goal. For example, if a programmer wants to display a rectangle on the screen, he sends the rectangle the message **display**. The rectangle knows what it has to do in order to display. The programmer can display a point the same way, by sending it the **display** message. This use of the same message in more than one situation (known as *overloading*) means that the internal details of the action of displaying are hidden. A programmer can change the definition of a rectangle, and, as long as the programmer updated the message definitions too, the same application program can ask a rectangle to display in the same way as before.

Since Smalltalk is not a strongly typed language, overloading is feasible and saves the programmer the overhead of determining exactly what kind of object is used before sending a message. In this example, both points and rectangles know how to display themselves. It would be inconvenient to have to send the message **rectangleDisplay** or **pointDisplay**, depending on the appropriate circumstance, when a programmer can get away with sending just the message **display**.

Overloading also hides the internal details of objects. This feature is useful for simplifying interfaces between portions of the system and applications and allows the programmer to use an object without having to learn the details of its internal structure.

Object-orientation demands that problem decomposition happen in a different way than in procedural languages. The decomposition must view problems so that solutions consist of objects sending messages and responding to messages. Simulations are especially easy to decompose this way.

Several features of the Smalltalk environment provide synchronization facilities and thus aid in simulations. These features include semaphores, shared queues and processes.

The Human Interface

The most visible features of Smalltalk are in its human interface. A Smalltalk display consists of a set of windows with varied number of panes. A three-button mouse is used for pointing and selection, and to direct the display of pop-up menus.

The menus are context sensitive; that is, they are different depending on what type of window the mouse pointer is in.

Graphics are an integral part of the Smalltalk environment. Simple graphics are easy to display since Smalltalk has classes representing points, pens, circles, lines, splines, etc. Smalltalk also provides low-level graphics in the form of bit block transfer (*BitBlt*), a way of writing bits from one location to another using various rules and masks.

Graphics allow visual display of concepts. The old adage 'a picture is worth a thousand words' is certainly true when attempting to explain abstract or difficult concepts. The visualization of algorithms is one area that makes good use of Smalltalk's graphical features and tools. (For an example, see Ralph London's article on program animation using Smalltalk.)

The windowing in Smalltalk is another example of visualization of concepts. Each window visually separates its contents from that of other windows (Figure 1). These multi-paned, overlapping windows allow greater information density on the screen, and help organize data. Windows may contain text and/or graphics. The typical Smalltalk application can use a window to display both text and graphics, usually each in a different pane of the window. Applications may also sketch anywhere on the screen and not be confined to a window. This ability is very useful in application development and prototyping.

The graphic interface provides a high degree of interactivity. Thus, expressions in Smalltalk-80 are executed by selecting the text denoting the expression and using the mouse to display a pop-up menu. For example, you'd select the menu item **do it** to execute some highlighted Smalltalk code, or the menu item **print it** to evaluate an expression and print the result. Input from a user can be gotten in a number of ways. A small window can pop up for the user to answer a question. Menus can be used for selecting among a number of items. The mouse can be used to designate locations on the screen that may be used as points, relative values or as binary feedback from the user. Some Smalltalk windows can look like buttons and perform as switches, for instance, so that only one button can be 'on' at a time.

Programming and Productivity

Smalltalk is an easy and enjoyable language to program in. One reason is that

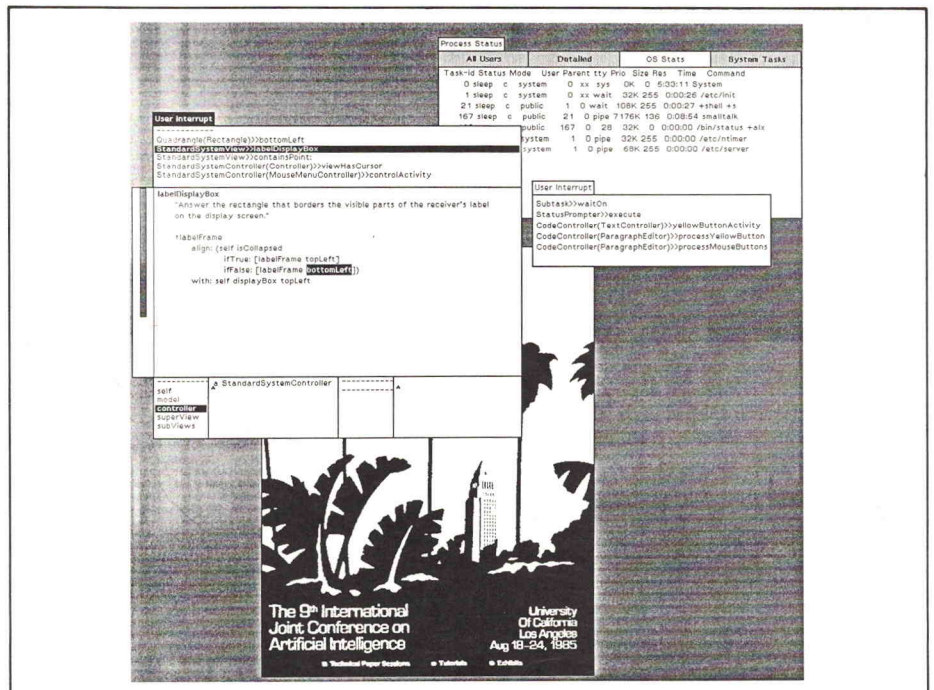


Figure 2. An example of programming efficiency: the user is informed that an error has occurred and can debug immediately.

it is incrementally compiled. This means the definition of a *method* (Smalltalk's equivalent to a procedure or a function) can be easily changed without the entire application's code having to be recompiled. Languages that are incrementally compiled allow immediate feedback and correction of errors.

Smalltalk also has a number of built-in tools that make programming interactive. One of these is the debugger. If a program breaks, or a break point has been set, or the user has interrupted the current process, a *notifier* appears informing the user of the error or interruption (Figure 2). The user can choose to debug at this time and can examine a trace of the execution stack and look at any method on the stack. Examination of methods occurs at the source-code level, and the expression currently being executed is highlighted. Highlighting focuses attention at the location of the error. Errors can be corrected in the debugger instead of having to switch to an outside editor and recompile. This tool makes immediate feedback and correction possible, and reduces the conventional compile/link/edit/debug cycle to an edit/debug cycle.

Two other tools that facilitate interactive programming are the *browser* and *inspectors* (Figure 3). The browser is a tool that organizes access to the large environment. Inspectors are used for examining and modifying objects. They are used in conjunction with the debugger so, for in-

stance, a programmer can modify the data in the method being debugged. The Smalltalk system also includes timing tools and a performance-monitoring tool.

Smalltalk is an exceptionally efficient language in which to program: some estimates of the increase in productivity from a conventional language to Smalltalk have ranged as high as ten times. The interactive nature of Smalltalk's environment accounts for a significant portion of the productivity improvement. The use of dynamic garbage collection frees the programmer from having to explicitly manage space, and also increases productivity. And, since Smalltalk is an extensible language, the programmer can personalize his programming environment, which further increases productivity.

Object-Oriented Programming

Another significant contribution to productivity comes from Smalltalk's object orientation.

A programmer works with information. The information, such as 10 ft. by 12 ft., contains data. The way the data is represented in a computer is known as a *data abstraction*. A particular data abstraction may help the programmer visualize the corresponding information.

Smalltalk's data abstraction descriptions are called *classes*. An instance of a class represents an instance of the data abstraction. For example, the information

about the size of a room, 10 ft. by 12 ft., could be represented by two numbers, 10 and 12. In Pascal, a programmer might represent this information by creating a structure called **room_dimension** that has two fields, length and width, and assign these fields the numbers 10 and 12. In Smalltalk, a programmer would create a class called **RoomDimension** that has two fields, length and width, called *instance variables*. An instance of the class **RoomDimension** would contain a length of 10 and a width of 12. This instance would respond to messages since it is an object and follows the object paradigm.

Smalltalk has a large number of existing classes. A programmer may 'borrow' already developed and debugged code from the existing classes. The nature of Smalltalk makes this borrowing very easy. These classes are arranged in a hierarchical order, so one class is a *subclass* of second, and consequently, the second is a *superclass* of the first. A class may have many subclasses, and a subclass inherits methods from its superclasses.

Besides inheriting methods, Smalltalk programmers may actually copy code from one class to another. Algorithms are most often copied, and they are often modified before incorporation into a new class. Smalltalk programmers live by the motto, 'Why reinvent the wheel if someone else has already debugged it?' Copying code is perfectly legitimate.

The type of application development that creates subclasses in order to focus the behavior of the data abstraction is known as programming by refinement. The large number of existing classes (over 200) also allows the user to practice programming by example, a proven technique for learning unfamiliar concepts.

Tektronix Smalltalk

Previously, Smalltalk has required special-purpose hardware in order to achieve adequate performance. For its 4400 Series, Tektronix has developed new implementation techniques that allow Smalltalk to run efficiently on a microprocessor-based system. (See Allen Wirfs-Brock's article on 4400 Series system architecture.)

Tektronix Smalltalk also includes an interface to the operating system that allows a Smalltalk process to make system calls. Among other things, this allows Smalltalk to access the RS232 port, to escape from Smalltalk by creating a shell, and to make inquiries of the operating system. The most important feature that

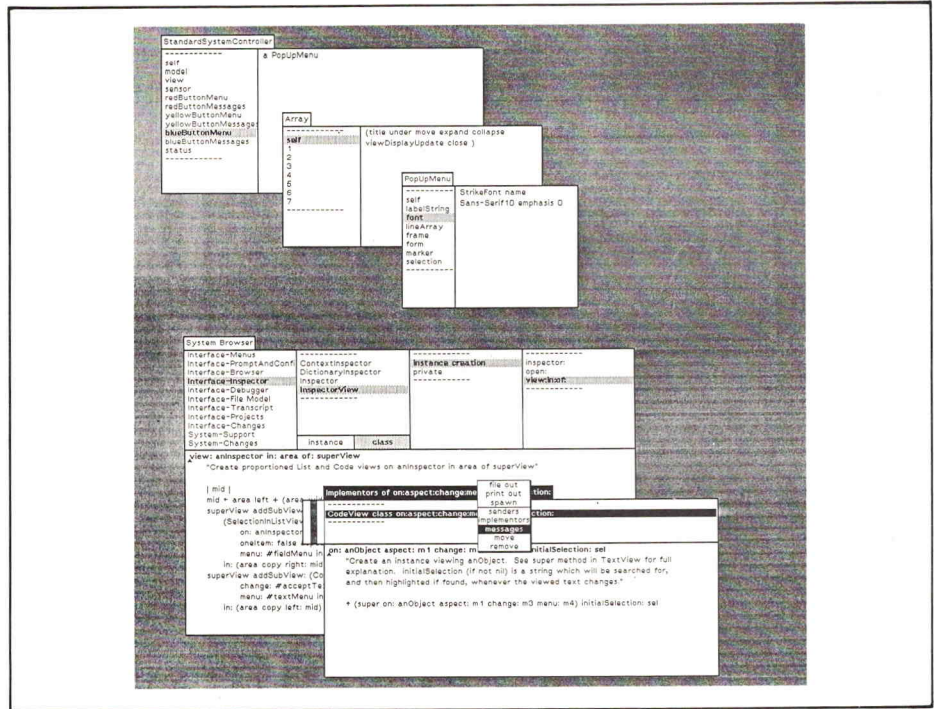


Figure 3. Browsers and inspectors facilitate interactive programming.

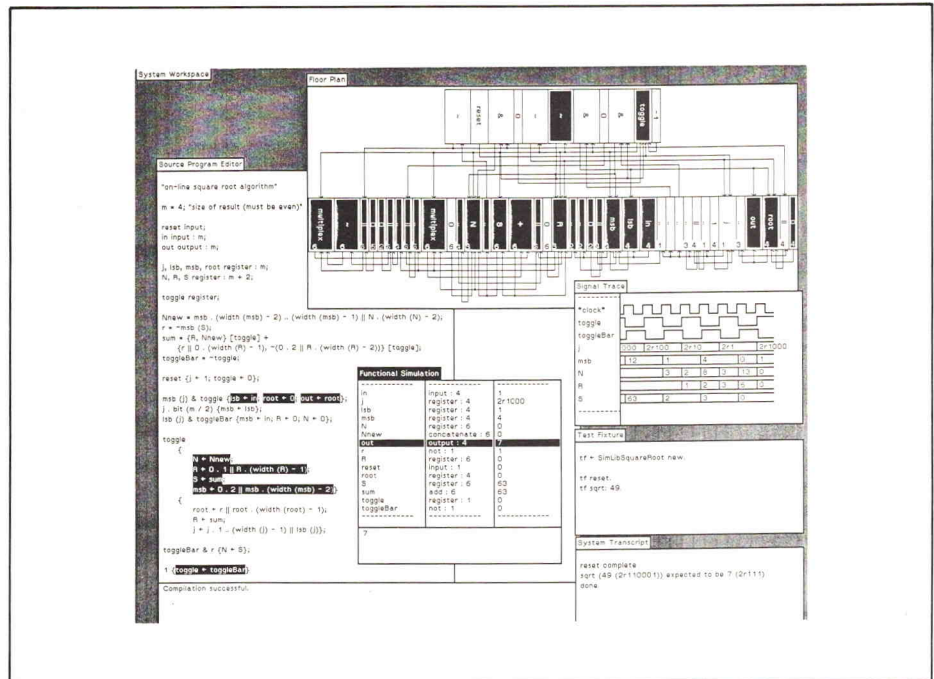


Figure 4. The Application Accelerator Illustration System, written in Smalltalk, simultaneously displays the hardware description (encoded in a hardware description language) and the visual results of a signal trace (in the "floor plan" window).

this interface allows is access to is other operating system processes. With the Tektronix interface, Smalltalk has the ability to execute fork and exec calls to create and access other OS tasks. These calls create a copy of the currently executing Smalltalk OS task and turn that copy into an executable program. Communication to these forked tasks is possible through pipes. In other words, Smalltalk can execute and talk to an applica-

tion program written in another language. Thus, Smalltalk applications may access an external database, number crunching facilities, or external communication programs.

Applications

AI research, application development and prototyping are feasible in Smalltalk since the environment and language are so easy to use. Incremental changes to an algo-

rithm are not painful because of the incremental development and debugging tools. The data hiding features of Smalltalk mean that if a programmer alters a data structure, only methods internal to the data structure need to be updated; the rest of application, including the interfaces to the modified data structure, does not have to be altered. Features such as the browser encourage the exploration that is so vital to efficient AI programming. The large environment gives the programmer access to many existing data abstractions and offers food for thought and prototyping.

Tektronix has created applications that illustrate the multitude of uses for Smalltalk. Many of these applications communicate with other programs written in Prolog, C or Lisp. A sample of these applications includes a distributed documentation system with complete version control and management, a project management application that encompasses communication and organizational aspects of a group project, an expert system for electronic troubleshooting written entirely in Smalltalk, and a similar expert system written in a combination of Smalltalk and Prolog.

Another application is an integrated Electrical Engineering CAD system called the Application Accelerator Illustration System (AAIS). It was developed over a period of three months by four people in the Tektronix Computer Research Laboratory (Ward Cunningham, Mike Miller, Steve Vegdahl and Chan Lee). This enormous output was accomplished in the typical Smalltalk style, by reusing and refining existing code from user interface and collection classes. The data abstraction and data hiding ability of Smalltalk allowed these people to work independently, yet because each area of the application didn't need to know the internal details of the other portions of the application, allowed their code to work together. The actual components of this EECAD system were built from scratch using AAIS's own hardware description language. AAIS included seven different kinds of windows, which gave it a very sophisticated user interface, particularly considering how quickly the project was accomplished (Figure 4).

Summary

Smalltalk's advanced features, including windows, inheritance and interactive programming, make it a pleasure to use. Its extensibility and flexibility mean it can handle a variety of applications without

stretching its functionality. Its tools for increasing productivity and its integrated environment qualify it as a leader among fifth-generation programming languages.

Currently, Tektronix' Artificial Intelligence Machines Group is offering classes in both beginning and advanced Small-

talk. For information on these classes, contact Jeff McKenna (503-685-2943).

*Smalltalk-80 is a trademark of Xerox Corporation. Unix is a trademark of AT&T Bell Laboratories. 

4400 Series System Architecture Supports Smalltalk Implementation

by Allen Wirfs-Brock
Artificial Intelligence Machines
Tektronix, Inc.
Wilsonville, OR

The team that developed Tek's 4400 Series Artificial Intelligence Systems faced a challenge: how to build a low-cost Smalltalk-80* system that would deliver the high performance needed for serious applications development.

To achieve high levels of performance, we chose to base the 4400 Series around Motorola's 68000 microprocessor family. It includes a large linear address space, a large register set and flexible addressing modes—all features that make it the most suitable microprocessor for executing AI-oriented languages. The lowest cost series member, the 4404, uses Motorola's 68010, operating at a clock speed of 10 MHz. The newer 4405 and 4406 systems use Motorola's 68020, which has a full 32-bit data path and operates at 16 MHz. All three systems include a floating point processor for enhanced arithmetic performance. The Smalltalk virtual machine is implemented by an assembly language program that runs under control of the operating system.

Prior to the development of the 4404, custom micro-code processors were required in order to efficiently execute Smalltalk programs. The 4400 Series Smalltalk implementation incorporated proprietary algorithms that allow a general-purpose processor such as the 68000 to execute Smalltalk as effectively as a micro-coded implementation.

Smalltalk and other AI-oriented languages require large amounts of main memory. The 4400 Series systems offer a minimum of one megabyte of dynamic RAM, with up to four megabytes maximum on the 4404 and up to six megabytes maximum of the 4406. Because many artificial intelligence applications

may require even larger amounts of memory, the system design includes hardware virtual memory support.

A bitmapped graphics display is an integral part of the Smalltalk environment. All 4400 Series systems include a monochrome frame buffer that is directly addressable as part of the 68010's or 68020's address space. On the 4404 and 4405, the display acts as a 640×480 pixel-sized viewport upon the 1024×1024 frame buffer. Hardware supports the smooth panning of the viewport over the entire frame buffer, under user control with the mouse. The 4406's 19-inch screen supports a 1376×1024 pixel-sized viewport, so the user can view the entire bitmap without panning.

Bus-oriented designs tend to increase system cost and reduce overall system performance, so the 4400 Series systems do not use a general-purpose internal bus structure. Instead, we used an integrated design that makes use of standard peripheral interfaces, including an RS-232-C interface for data communications, a parallel printer port, and the Small Computer System Interface (SCSI) for access to the 45 megabyte hard disks (a 90 megabyte hard disk is provided on the 4406). An Ethernet LAN interface is optional.

The entire standard configuration of the 4400 Series systems is able to fit within a standard Tektronix terminal package—a significant factor in allowing us to meet our second goal, which was to produce a system that could be manufactured economically and sold at relatively low cost.



Animating Programs: A Smalltalk Application

by Ralph L. London
Computer Research Laboratory
Tektronix Labs
Beaverton, OR

At Tek's Computer Research Laboratory, my colleague Robert A. Duisberg and I are using Smalltalk and the 4404 to develop techniques and programs that allow ourselves and others to animate programs—that is, to show graphically how a program works.¹ We visualize the execution of programs by creating “movies” of the program's actions. We make visible the normally invisible data structures and operations of programs.

Smalltalk provides us a highly interactive, very visual, productive programming environment in which to prototype our animation tools. We are able to use its graphics facilities to create the figures and schematic diagrams needed for animating programs. There are some system-supplied animation capabilities from which we can create the necessary movements. Fortunately, we do not need exquisite shadings for our animated objects. Object-oriented programming is quite appropriate for separating the actual code of the program being animated from both the graphics and interface code. For our needs and the examples we have done so far, Smalltalk's speed has been sufficient to provide the necessary illusions of motion and dynamic representation of programs. Indeed, at times we must provide delays so the viewer can see all that is happening. It is, of course, possible to create animations with other languages such as C, Pascal, Lisp, or even BASIC, provided high resolution graphics are available.

Several groups of people can benefit from our animation tools. Designers of algorithms can see what their algorithms are actually doing. We have discovered optimizations to algorithms by watching executions and literally seeing the improvements. Using visual representations of processes has exposed problems that textual data and standard debugging presumably would have missed. Animations can serve as a form of documentation for programs. They can aid program enhancement and change, especially

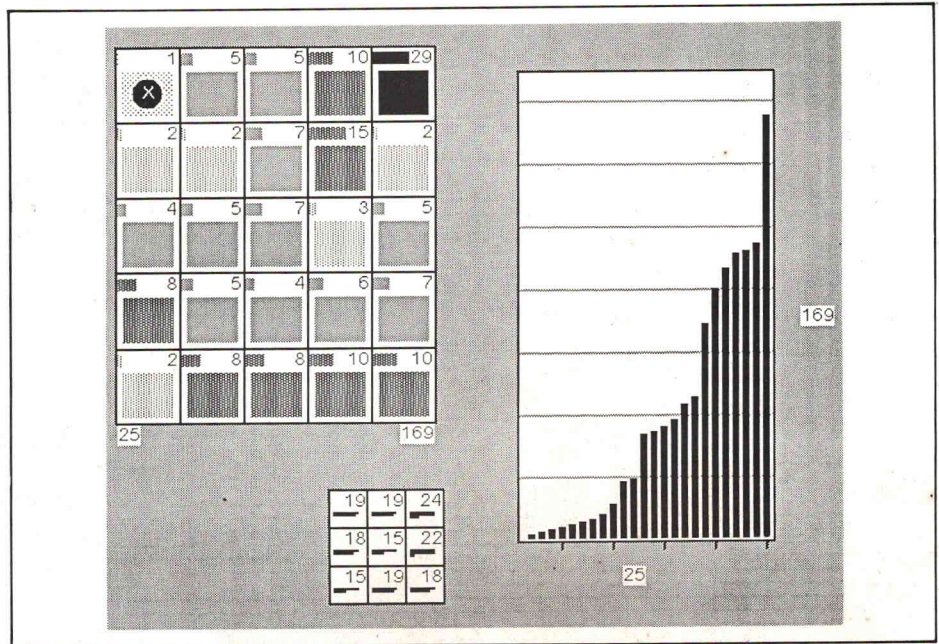



Figure 1. In Goldberg and Robson's book is a well-known programming problem to find how many steps a drunken cockroach takes to land on every tile of a rectangular grid at least once.⁴ The snapshot from our animation of the drunken cockroach problem shows the final result of one such random walk, which happened to end in the upper-left corner. The actual animation shows the idealized cockroach moving from tile to tile while also revealing interesting aspects of the relevant information that is updated and displayed in various ways.

after several months of nonuse. With Ken Dickey, of Tek's Design Automation Group, we are currently using animation as the method of choice for performance tuning an operating system before it goes into a Tek instrument. In the more general area of exploring designs, there is the need to visualize both the *structure* and *dynamic behavior* of alternatives. Animation can be used in the classroom to teach students about computer science, including introductory programming, algorithms, and data structures; other possible subjects include physics, mathematics, and neural science. In fact, such instruction is being done at Brown University using animation tools developed there.²

In order for others to animate programs and objects of interest, it is necessary that the animation tools require no more than an acceptable level of their effort. We are currently constructing an animation kit of interconnectable parts based on constraint satisfaction.³ The crucial addition to existing constraint systems is the ability to include time among the constraints to be satisfied. We expect animators to connect the parts, some perhaps designed and specified by themselves, by using a novice programming environment—one that allows certain direct manipulations

without assuming expert programming skills. We have used this prototyping tool, named *Animus*, for several interesting examples, including the operating system tuning. The speed with which an animator connects the parts and then *Animus* compiles the necessary Smalltalk code to create the animations, is indeed impressive. 

References

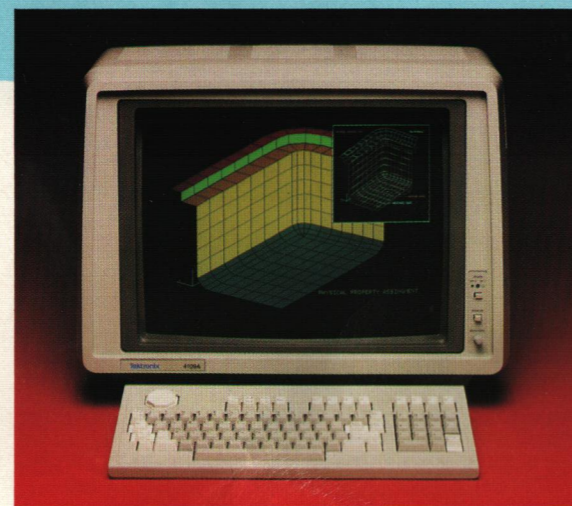
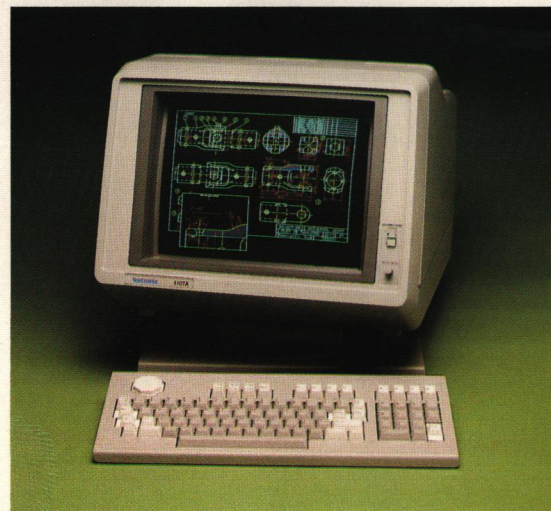
1. Ralph L. London and Robert A. Duisberg, "Animating Programs Using Smalltalk," *Computer*, 18, 8, August 1985, pp. 61-71.
2. Marc H. Brown and Robert Sedgewick, "Techniques for Algorithm Animation," *IEEE Software*, 2,1, January 1985, pp. 28-39.
3. Robert A. Duisberg, *The Design and Implementation of Animus: A Constraint-Based Animation Kit* (working title), Ph.D. thesis, University of Washington, in preparation.
4. Adele Goldberg and David Robson, *Smalltalk-80: The Language and its Implementation*, Menlo Park, CA: Addison-Wesley, 1983.

TEKTRONIX GRAPHICS TERMINALS AND WORKSTATIONS

Tektronix Graphics Terminals and Workstations

Terminal Characteristics	4104A	4105A	4106A	4107A	4109A	4111	4125	4128	4129
Screen size (diagonal)	13 inch	13 inch	13 inch	13 inch	19 inch	19 inch	19 inch	19 inch	19 inch
Display addressability	480×360	480×360	640×480	640×480	640×480	1024×768	1280×1024	1280×1024	1280 × 1024
Addressable coordinate space	4096×4096	4096×4096	4096×4096	4096×4096	4096×4096	4 billion × 4 billion	4 billion × 4 billion	4 billion × 4 billion	4 billion × 4 billion
Color palette	64	64	64	64	4096	4096	16 million	16 million	16 million
Displayable colors	4 graphics 4 dialog	8 graphics 8 dialog	16 graphics 8 dialog	16 graphics 8 dialog	16 graphics 8 dialog	16 standard	4 standard 16/64/256 opt	16 standard 64/256 opt	256 standard 4096 with dithering
Segments	no	no	limited	yes	yes	yes	yes	yes (2D & 3D)	yes (2D & 3D)
Local zoom/pan	no	no	yes	yes	yes	yes	yes	yes (2D & 3D)	yes (2D & 3D)
Upgrade to 3D	no	no	no	no	no	no	yes	3D wireframe std; upgrade to shaded surfaces	shaded surfaces standard
Local memory Standard Maximum	32 KB	32 KB	32 KB 256 KB	256 KB 1 MB	256 KB 1 MB	256 KB 1.2 MB	288 KB Ecc 800 MB Ecc	288 KB Ecc 800 MB Ecc	288 KB Ecc 800 MB Ecc
Optional coax support for IBM 3270	no	no	yes	yes	yes	yes	yes	yes	yes

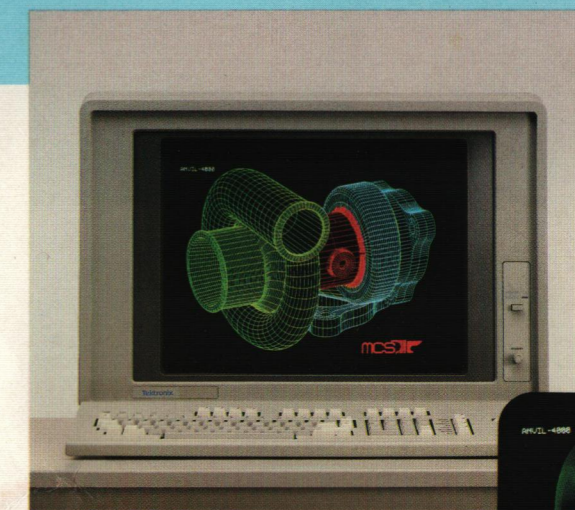
4107A
Computer Display
Terminal



4109A
Computer Display
Terminal

Display courtesy of PDA Engineering

4111
Computer Display
Terminal



4128 and 4129
3D Color Graphics
Workstations

Displays courtesy of MCS, Inc.



Tektronix, Inc.

SALES AND SERVICE OFFICES

ALABAMA

Huntsville 35805
4900 Corporate Drive
Suite H
Phone: (205) 830-9212

ARIZONA

(Phoenix)
3015 S. 48th Street, Suite 100
Tempe 85282
Phone: (602) 438-1011
Tucson Area: (602) 790-3099
Mailing Address:
P.O. Box 29540
Phoenix 85038

CALIFORNIA

(Concord)
3451 Vincent Road
Pleasant Hill 94523
Phone: (415) 932-4949
From Oakland/San Francisco: (415) 254-5353
From Sacramento: (916) 447-5072
From Fremont/Milpitas: (415) 490-7067
From Livermore: (415) 449-5176
Mailing Address:
P.O. Box 4040
Concord 94524-2040

Irvine 92714
17052 Jamboree Blvd.
Mailing Address:
P.O. Box 19523
Irvine 92713
Phone: (714) 660-8080
TELEFAX: (GP1)(714) 660-8080 X311

(Los Angeles)
21300 Erwin Street
Service Center
20920 Victory Blvd.
Woodland Hills 91367
Phone: (818) 999-1711
Mailing Address:
P.O. Box 8500
Woodland Hills 91365

San Diego 92123
5770 Ruffin Road
Phone: (619) 292-7330

Santa Clara 95054-1196
3003 Bunker Hill Lane
Phone: (408) 496-0800
TELEFAX: (GP 1) (408) 496-0800

COLORADO

(Denver)
393 Inverness Drive South
Englewood 80112
Phone: (303) 799-1000
Telex: (Infocom) 45-4455
From Colorado Springs: (303) 634-3933

CONNECTICUT

Milford 06460
40 Commerce Park Road
Phone: (203) 877-1494

FLORIDA

Fort Lauderdale 33309
2003 N.W. 62nd Street
(known as) Cypress Creek Road
Phone: (305) 771-9700
From Miami: (305) 947-6053
Also serves Puerto Rico and
U.S. Virgin Islands

Orlando 32803
3657 Maguire Blvd., Suite 100
Phone: (305) 894-3911
From the Cape Kennedy Area:
(305) 636-0343

Pensacola 32503
4700 Bayou Blvd., Bldg. 1
Phone: (904) 476-1897

GEORGIA

(Atlanta)
Technology Park/Atlanta
650 Engineering Drive
Norcross 30092
Phone: (404) 449-4770
Mailing Address:
P.O. Box 6500
Norcross 30091

HAWAII

Honolulu Service Center 96819
EMC Corporation
550 Paiea Street
Phone: (808) 836-1138 (Service)
(800) 538-8125/6 (Sales)

ILLINOIS

(Chicago)
5350 Keystone Court
Rolling Meadows 60008
Phone: (312) 259-7580
TELEFAX: (GP 1) (312) 259-7580

INDIANA

Indianapolis 46268
8751 Wesleyan Road
Phone: (317) 872-3708

KANSAS

(Kansas City)
10513 West 84th Terrace
Lenexa 66212
Phone: (913) 541-0322
Omaha, Lincoln, Wichita
ENterprise 6537

LOUISIANA

(New Orleans)
1940 I-10 Service Rd.
Concourse Place
Kenner 70065
Phone: (504) 466-4445

MARYLAND

(Baltimore)
102 Lakefront Dr.
Cockeysville 21030
Phone: (301) 628-6400

DC
700 Professional Drive
P.O. Box 6026
Gaithersburg 20877
Phone: (301) 948-7151
TELEFAX: (GP 1) (301) 948-7151 X321

MASSACHUSETTS

(Boston)
482 Bedford Street
Lexington 02173
Phone: (617) 861-6800

MICHIGAN

(Detroit)
24155 Drake Road
Farmington 48024
Phone: (313) 478-5200

MINNESOTA

St. Paul 55126
4660 Churchill Street
Phone: (612) 484-8571

MISSOURI

(St. Louis)
2318 Millpark Dive
Maryland Heights 63043
Phone: (314) 429-7707

NEW JERSEY

Woodbridge 07095
40 Gill Lane
Phone: (201) 636-8616
TELEFAX: (GP 1) (201) 636-8616 X266

NEW MEXICO

Albuquerque 87108
1258 Ortiz Drive, S.E.
Phone: (505) 265-5541
Southern N.M. Area: ENterprise 678
Southern Nevada Area: ENterprise 678
El Paso, TX ENterprise 678
TELEFAX: (GP 1) (408) 358-3421

NEW YORK

Albany 12205
26 Computer Drive West
Phone: (518) 458-7291

(Long Island)
100 Crossways Park West
Woodbury, L.I. 11797
Phone: (516) 364-9060
NYC Customers (718) 895-9215

Poughkeepsie 12601
Beechwood Office Park
385 South Road
Phone: (914) 454-7540

Rochester 14623
1210 Jefferson Road
Phone: (716) 424-5800

(Syracuse)
1 Northern Concourse
North Syracuse 13212
Phone: (315) 455-6661

NORTH CAROLINA

Raleigh 27612
3725 National Drive, Suite 104
Phone: (919) 782-5624

OHIO

(Cleveland)
7830 Freeway Circle
Middleburg Heights 44130
Phone: (216) 243-8500 (Sales)
(216) 243-8505 (Service)

Dayton 45449-2396
501 Progress Rd.
Phone: (513) 859-3681

OKLAHOMA

Oklahoma City 73108
4400 Will Rogers Parkway
Suite 220
Phone: (405) 943-8127
Oklahoma Wats Only
Phone: (800) 522-8196

OREGON

10220 S.W. Nimbus Drive
Suite K-4
Portland 97223
Phone: (503) 620-9100

Factory Service Center
Tektronix Industrial Park
Beaverton 97077
Phone: (503) 642-8600
TWX: (910) 467-8708
TLX: 15-1754

PENNSYLVANIA

(Philadelphia)
450 Sentry Parkway
Blue Bell 19422
Phone: (215) 825-6400

Pittsburgh 15221
1051 Brinton Road, Suite 300
Phone: (412) 244-9800

TENNESSEE

Knoxville 37923
9041 Executive Park Drive
Suite 411
Phone: (615) 690-6422
From Oak Ridge (615) 482-7349

TEXAS

(Dallas)
1551 Corporate Drive
Irving 75038
Mailing Address:
P.O. Box 165027
Irving 75016
Phone: (214) 258-0525
Metro: (214) 256-5534
TELEFAX: (GP 1) (214) 258-0525 X256

Houston 77099
10887 S. Wilcrest Drive
Phone: (713) 933-3000
Mailing Address:
P.O. Box 4309
Houston 77210

San Antonio 78232
14800 San Pedro Avenue
Suite 112
Phone: (512) 496-1161

Kelly 78226
Billy Mitchell Center
227 Billy Mitchell Road
Phone: (512) 432-1341

UTAH

Salt Lake City 84115
Timesquare Park
300 Mercer Way
Phone: (801) 486-1091

VIRGINIA

(Crystal City)
Hayes Building
Suite 1004
2361 S. Jefferson Davis Hwy.
Arlington 22202
Phone: (703) 920-7770

Newport News 23602
606 Denbigh Blvd., Suite 703
Phone: (804) 874-0099

WASHINGTON

(Seattle)
3709 157th Avenue NE.
Redmond, WA 98052
Phone: (206) 885-0900

CORPORATE OFFICE

Tektronix, Inc.,
P.O. Box 500
Beaverton, Oregon 97077
Telephone: (503) 627-7111

PRINCIPAL PLANT


Tektronix Industrial Park,
Beaverton, Oregon 97077

DIRECT ORDER:

For Continental United States, Alaska,
Hawaii, Virgin Islands and Puerto Rico.
Contact our National Marketing Center.
Phone: (800) 426-2200
For State of Oregon,
call collect (503) 627-9000.

ADDITIONAL LITERATURE

or Tektronix Sales Office
serving you:
P.O. Box 1700,
Beaverton, Oregon 97075
Phone: (800) 547-1512
Oregon only: (800) 452-1877
TLX: 151754,
TWX: (910) 467-8708
TEKTRONIX BEAV.

Copyright © 1986, Tektronix, Inc. All rights reserved. Printed in U.S.A. Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specification and price change privileges reserved. TEKTRONIX, TEK, PLOT 10, TEKTEST, SCOPE-MOBILE, and  are registered trademarks. For further information, contact: Tektronix, Inc., Corporate Offices, P.O. Box 500, Beaverton, OR 97077. Phone: (503) 627-7111; TLX: 151754; TWX: (910) 467-8708. Subsidiaries and distributors worldwide.

New rasterizer offers increased functionality plus support for 4695 copier and 4110/4120 Series workstations. Existing 4510s can be upgraded.

4510A and New Device Driver Increase Options for Superior Hard Copy Output

As the demand for high quality graphics hard copy—and greater user productivity—continues to rise, a rasterizer has become a key component of a color output system. Compared to an ordinary screen copy, a rasterizer provides improved image resolution and color quality. It also enhances productivity by shortening the time that the terminal is tied up to make a copy.

Users of Tek's 4106A, 4107A, 4109A and 4111 Computer Display Terminals have enjoyed easy access to Tek's 4510 Color Graphics Rasterizer. Now, the new 4510A Rasterizer, in combination with Version 8 of 4110/4120 Series Option 10, brings the benefits of rasterization to a broader spectrum of users. The 4510A now provides true plug-to-plug compatibility with Tek's 4110 and 4120 Series Color Graphics Workstations, as well as the previously supported 4100 Series terminals. In addition, the 4510A supports the full range of Tek's color graphics copiers—the low-cost 4695 copier as well as the 4691 and 4692. Table 1 summarizes 4510A compatibility.

Why a Rasterizer?

For Tek's copiers and terminals, an image consists of a pattern of dots or *pixels*—picture elements. For a terminal such as the 4107A, which has 640×480 addressable points, each screen image contains over 300,000 separate dots. For the 4120 Series workstations, with 1280×1024 addressability, each displayed screen contains over 1.3 million dots.

Inkjet copiers such as Tek's 4690 Series generally work as *direct-screen copiers*. That is, the copier reproduces, dot by dot, whatever is displayed on the screen. The hard copy output is thus produced at the *terminal's* resolution. The copiers are capable of higher dot resolution than the terminal, however. For example, the 4692 allows up to 1536×1152 dots in an A-size image, or over 1.7 millions points. A B-size 4691 image can contain 2079×1560 dots, or over 3.2 million points. The 4695 allows 1024 dots across the page and, due to roll

Table 1. Compatibility Matrix

Tek Products	4510A Support?
4100 Series Terminals	4106A, 4107A, 4109A, 4111*
411X Series Terminals	Yes—with Option 10, Version 8 or higher
4120 Series Workstations	Yes—with Option 10, Version 8 or higher. 2D support only.
4691 Copier	Yes
4692 Copier	Yes
4695 Copier	Yes

* Initial 4111 shipments will not have 4510A support, but a no-charge update will take place shortly thereafter.

paper media, any number of dots down the page. The typical image is 1024×1355, or nearly 1.4 million points.

Here's where the 4510A rasterizer comes in. To drive the copiers at their full resolution, the 4510A takes the original graphics primitives used to create the picture on the terminal (the vectors, panels and so forth), converts it into raster format to match the copier's full resolution, and sends the raster format dots to the copier. The result is a sharper hard copy image than the image displayed on the

screen—a difference, for example, of 35 percent higher resolution (or information content) on the 4692 or A-size 4691 than on the 4120 Series workstation.

The improved resolution is significant (Figure 1). Characters that may have been fuzzy on the terminal screen become crisp, and "stair-stepped" raster lines become practically smooth. Even E-size engineering drawings remain legible when printed on B-size paper on the 4691 copier.

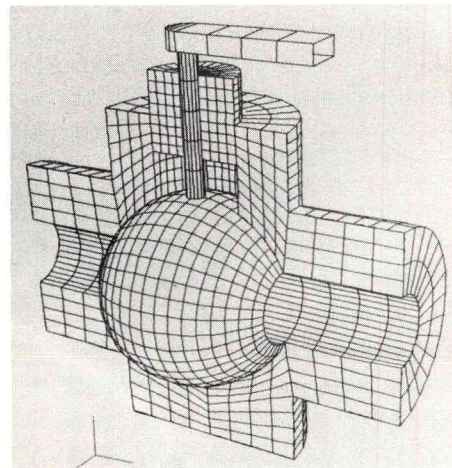
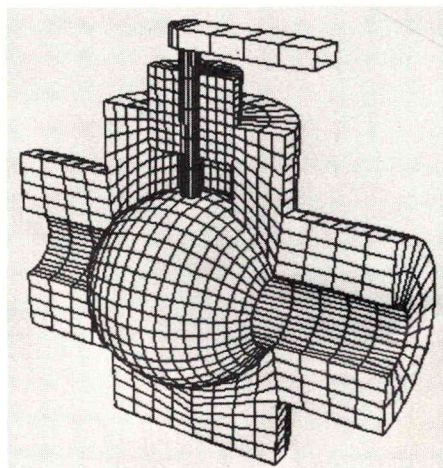


Figure 1. The 4510A drives the copier at its full speed and resolution. The rasterized output (1a) is generated at the 4692's 154 dots per inch, regardless of the terminal used. The non-rasterized hard copy in Figure 1b. is produced at the terminal's resolution, in this case the 640 × 480 of a 4107A terminal.

The color quality of the hard copy is enhanced as well. The 4510A has a palette of 132,651 colors, with 256 printable per image—much more than the few hundred colors of the copiers themselves. The enlarged color palette allows smoother shading of color transitions and a closer match of hard copy and on-screen colors.

A Flexible Resource

For maximum flexibility in using the rasterizer, the 4510A comes standard with a four-channel, multiplexed, RS-232-C interface. The rasterizer can be connected directly to a combination of up to four terminals, workstations or hosts (Figure 2). Connecting the 4510A to a host allows many users to share both the rasterizer and the copier, lowering the cost per user of both pieces of equipment.

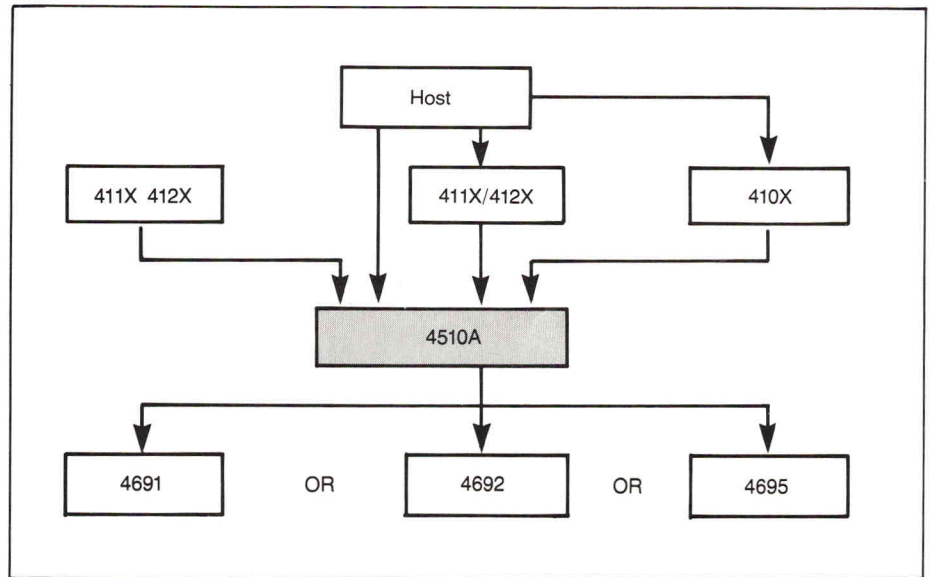


Figure 2. A four-channel, multiplexed RS-232-C interface allows up to four terminals, workstations or hosts to connect to the 4510A, thus reducing the cost per user.

Device driver support for the 4510A is included in many host software packages, including PLOT 10[®] TekniCAD and PLOT 10 IGL as well as packages from ISSCO[®], Precision Visuals, SAS Institute, Swanson Analysis Systems, UNIRAS and Zycor.

A number of new capabilities provide closer compatibility with the feature sets of the terminals, workstations and copiers. For example, the 4510A supports 32-bit coordinates, segment transforms, multiple viewports, and a new panel-based font. The 4692's repaint capability is also supported. This feature can be used to produce vivid transparencies by making multiple passes of the same image without requiring additional terminal or host involvement. The 4510A also allows continuous images to be generated on the 4695 roll media.

Productivity Gains

In addition to enhancing image quality, the 4510A also improves personal productivity by reducing the time that the terminal is dedicated to transmitting data to the copier. A typical screen copy ties up the terminal for 2–4 minutes. With the 4510A connected to a terminal, the process can be completed in anywhere from half a minute to approximately two minutes, depending on image complexity and speed of transmission. When the rasterizer is connected to the host, files can be copied with virtually no terminal tie up, resulting in the highest productivity gains.

Another productivity feature is the new multiple-image queueing facility, which allows multiple images to be sent from the active port to the 4510A. The number of files that can be stored in the 4510A de-

pends on the 4510 memory configuration and the size of the image files. Three memory configurations make it easy to select the option that matches your requirements. The 128K byte memory option can handle most presentation graphics and scientific data analysis output. The 512K byte option is suitable for more complex scientific, mechanical CAD and architectural applications. Complex CAD applications such as circuit board design and thematic or contour mapping may require the 2M byte option. For even larger memory needs, the 4510A can partition an image and handle a portion of it at a time.

To produce output quickly, the 4510A transmits data to the copier over an eight-bit parallel link. The rasterizer drives the copier at its full printing speed. In addition, a panel pixelation enhancement in the 4510A provides a ten-times speed improvement over the 4510 for the processing of simple panels.

Upgrade Path

If you have a 4510 and want to upgrade to the new capabilities, you have three options, depending on what terminal you have, whether you want 4695 support, and (if your terminal is a 4125, 4128 or 4129), whether you already have Option 10:


- **4510F50.** This kit is designed for rasterizers used with 4100 Series terminals. It upgrades the 4510 to the firmware functionality of the 4510A, but does **not** add 4695 support.
- **4510F51.** This kit is designed for rasterizers used with a 4115B or 4120

Series workstation. It includes both the 4510-to-4510A firmware capability upgrades **and** the new (version 8) Option 10 firmware. However, it does **not** add 4695 support. This kit is **not** available for a limited time to allow 4110/4120 customers to obtain compatibility with the 4510A simply, with one field kit.

- **4510F52.** The works—complete 4510A capability, including 4695 support and new Option 10 firmware.

All kits must be installed by your Tektronix service representative.

Setting the Standard for Graphics Output

A rasterizer such as Tek's new 4510A is a vital element in obtaining the highest quality color graphics output. By supporting a broader spectrum of Tek terminals and copiers than ever before, the 4510A offers superior output to users in fields ranging from technical data analysis through computer-aided design and manufacturing. The 4510A also enhances user productivity, and its four-way multiplexer means the cost per user of the higher quality output is reduced. 

Software Development on a Personal Workstation

by Eric J. Anderson
Mark T. Bell
Graphics Workstation Division
Tektronix, Inc.
Wilsonville, OR

The 6130's Distributed File System and sophisticated development tools overcome many of the problems associated with the traditional, single-computer host, and improve the software engineer's working environment and productivity.

For software development, as for many other types of product development, the environment provided by Tek's 6130 Intelligent Graphics Workstation offers a number of contrasts to the traditional shared-host computer environment. For example, a shared host is often characterized by high load averages, which climb as more engineers join a project. Compilation times increase, and functions such as file editing become tedious as response times lengthen. With the 6130 workstation, on the other hand, the user controls the load on the machine with an essentially dedicated processor.

On a shared host, disk space availability can become critical. Typically, some disk resources are devoted to non-development use, and some development groups tend to use a disproportionate amount of space. On the 6130, however, the user chooses how the disk is filled.

The shared host is generally locked away in a controlled environment to minimize failures due to environmental factors. In contrast, the 6130 workstation is designed with a degree of ruggedness that suits the standard business environment, and requires no special environmentally controlled rooms.

The shared host is typically managed by a central computer support staff. Downtime for planned maintenance tasks does not always fit engineers' schedules, and the loss of one machine halts work for many developers. The workstation, however, is a personal device, and the user can configure the machine to taste. The loss of one workstation to scheduled or unscheduled downtime affects only the user of that machine. In addition, since the user is generally the workstation's system administrator, he or she can schedule normal maintenance and system backup according to personal preferences.

For team-oriented projects, however, a shared host has traditionally had one clear advantage over most workstations: in a single-host environment, all the files for a project usually reside on the same system, which allows users to easily share files.

To address this problem, the 6130's UTek* operating system includes a *Distributed File System* that removes the barriers to file access between workstations. The DFS allows 6130 users to enjoy both the increased productivity of a workstation environment *and* the easy file sharing of a large host environment. The DFS thus makes it possible for design teams to easily handle large, complex projects in a personal workstation environment.

Distributed File System

UTek's Distributed File System is a natural extension of the UNIX* file system. The basic UNIX file system is a tree structure with directories that contain subdirectories and files. The root of the tree is a directory called "/". The root is the starting point from which all files on a system can be accessed. File names are built by listing the path from the root to the file, with the components of the path separated by the delimiter "/". For example, the "C" compiler is a file named *cc* that resides in a directory called *bin* (for binary). *Bin* is a subdirectory of the root directory. Thus, the full name of the compiler and its location in the file system is */bin/cc*.

Special files called *symbolic links* make the file system more flexible. Symbolic links are actually pointers to files that reside in another part of the file system. They make it easier for two or more users to share a file by having the file appear to reside in two parts of the file system simultaneously. For example, suppose engineers Tom and Dick are working on different parts of a project, and share a common header file called *common.h*. The file actually resides in Dick's working directory */usr/dick/common.h*. Tom could have a file */usr/tom/common.h* that is a symbolic link to the file in Dick's directory. This makes access to the file easier for Tom because the file appears to be in his working directory.

With the DFS, the network appears to the user as a large file system with a root directory called *//*. The root file system of the workstation behaves like a subdirectory of this much larger (distributed) file

system. The name of the subdirectory of *//*—that is, the workstation's file system—is the name by which the workstation is known on the network. Thus, if Dick's workstation is called *Tek1*, the DFS path of the header file is *//Tek1/usr/dick/common.h*. The advantage of this network implementation is that the network is invisible to normal programs, and symbolic links can be used to reference files on other workstations.

UTek Software Development Tools

In addition to the problem of shared files, the management of large software development projects requires that two other problems be addressed:

- There must be a protection mechanism that controls access to files, not only to ensure that a valid user/developer can access any file, but also to ensure that only one user is modifying a file at a time.
- The routine tasks of the development process must be automated. As the size of the project grows, so does the complexity of controlling the software build process. Not all team members have the knowledge needed to put together a complex system based on a large number of source files each, with its own peculiarities.

UTek provides two development tools that address these problems: a Revision Control System (RCS) and the *make* utility. Because of the Distributed File System, these tools, which were developed for use on mainframe computers, can be used without modification in the workstation environment.

Source Code Control

The team approach to software development requires that members of the team independently develop sections of code and reliably distribute new code as it is completed. On the 6130, this can be done through the Revision Control System.

RCS works somewhat like a filing system. When a file is updated, a copy is maintained so that previous versions of the file can be accessed. Once a file has been

checked in or placed under RCS control, RCS ensures that only one user at a time can check out and modify the file.

RCS control files are generally kept in a subdirectory called *RCS* of the directory in which a working copy of the file resides. For example, say our hypothetical engineer, Dick, has a directory called *projectA*, which includes a file called *part1* that is currently under active development. If Dick checks out *projectA/part1* under RCS, the file is locked so that only he is permitted to modify its contents. The file maintained by RCS will contain the current revision, as well as a list of changes from the initial revision and a log of comments by the authors of each change.

Automated Build Control

As the complexity of software systems grows and the need for portability and maintainability increases, it becomes impossible to rely on the memory (or the availability) of the original developers for the procedures needed to rebuild or modify that system. Every command that must be entered when building a system from its component parts is a potential source of error. Steps are invariably left out, dependencies forgotten, and procedures initiated out of order.

To streamline the build process and reduce the chance of error, UTeK includes the program known as *make*. *Make* provides a way to specify once, in a file called a *makefile*, all the subsystems required in building a system and the steps needed to put the subsystems together. For each subsystem, the developer specifies other subsystems or steps that must be built or performed prior to the creation of the current subsystem. At each step, *make* verifies that the dependencies exist and are up to date before executing that step.

Make saves the developer time in two ways:

- It reduces the number of commands that the developer or maintainer must execute in order to rebuild the system. Often the command *make system* is all that is necessary.
- It recognizes when parts of a system need to be remade and only remakes those that are necessary.

Make is also a general tool. It can be used with RCS to cause new versions of a file to be checked out automatically when needed. It is not restricted to the software development environment. Any system (for example, this document) can have its build procedure defined in a *makefile*.

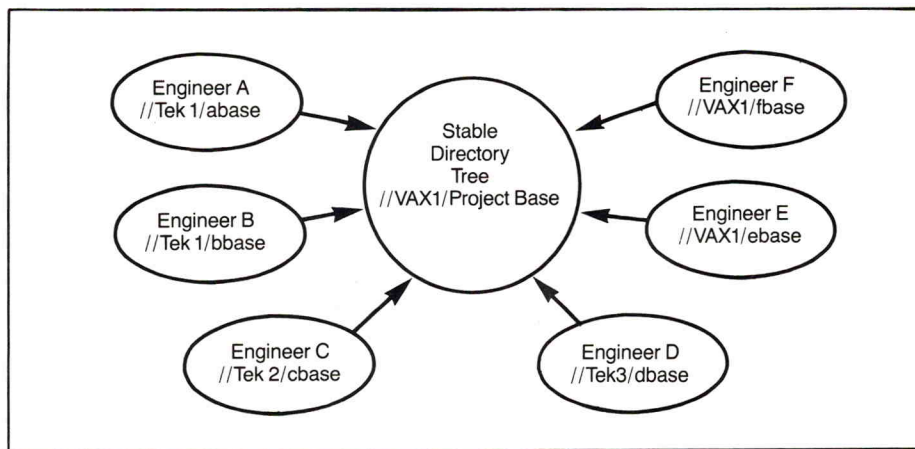


Figure 1. The Distributed File System enables symbolic links from the working trees to the RCS directories in the base directory tree on the host. This technique saves disk space and ensures that all team members have to the latest file revision.

A Development Example

Software development for the 6130 workstation began in an environment consisting of several Digital Equipment Corporation VAX* computers running Version 4.2bsd·UNIX. Later, as the hardware development was completed, some of the software development was offloaded to 6130 workstations, giving us the opportunity to become users of our own product.

UTek kernel development was one of the first projects to migrate to the workstation environment. Since the kernel is a large piece of code involving many engineers, it was important, even on the VAXes, to minimize the amount of disk space needed for its compilation. To this end, certain directory trees were designated as *stable* and were kept up-to-date with the latest versions of source code.

Working directory trees, which are copies of the stable tree, were then made for development. Symbolic links were used to replace the RCS directories in the working tree with pointers to the actual RCS directories in the stable tree.


Changes to source code migrated from the working tree to the stable tree via RCS. Each developer would check new source code in from the working tree after testing it. Since all working trees and the stable tree share the same RCS directories, the other developers would automatically get the new code the next time they rebuilt their copy of the system.

We achieved additional savings of disk space by replacing unchanged subdirectories in the working tree with symbolic links to the corresponding subdirectories in the stable tree. In this way the developer was given the appearance (and convenience) of having a complete source

tree without all the disk overhead. With the DFS, it was a simple matter to place the working tree on a workstation with symbolic links back to the stable tree on the VAX (Figure 1).

Shifting development of the kernel to the workstation environment produced several benefits. The primary one was increased developer productivity. Changes could be made and tested much more quickly. Too, development did not totally halt when unscheduled maintenance or an equipment accident disabled all the development mainframes for a day or more. In addition, the availability of disk space on the mainframes became less critical as development shifted to workstations.

In Conclusion

Software development tools such RCS and *make* have demonstrated their usefulness in controlling access and changes to large, complex software projects. With the Distributed File System, these tools can be used effectively across a multi-machine, networked environment. The 6130 multi-machine environment spreads the workload among many workstations, and minimizes the disruption to the total project effort from downtime. Personal workstations not only provide a dedicated CPU, but also a dedicated development machine tailorable to the specific needs of its user. The result: a very productive environment for conducting large-scale software development projects—or many other types of product development. 

*UTek is a trademark of Tektronix. UNIX is a trademark of AT&T Bell Laboratories. VAX is a trademark of Digital Equipment Corporation.

Solution Vendors Support Tek Graphics in IBM Host Environments

To provide customers with the best possible solutions to application requirements, the Tektronix Solution Vendor Program identifies and integrates application software that is available from a variety of independent software vendors with Tek's high-quality graphics hardware. As Tek has strengthened its compatibility with IBM host environments (see the Fall, 1985 issue of *Tekniques* for details), a number of Solution Vendors have provided driver support for Tek's CX4100 Series terminals or 4120 Series workstations with Option 5. The following table lists Solution Vendors who have announced such support. For further information on the specific hosts or terminals/workstations that are supported, check with the Solution Vendor directly, or with your Tektronix Sales Engineer. 

Software Support for Tek Terminals/Workstations in IBM Host Environments

Application	Product Name	Vendor
Data Analysis & Presentation Graphics	BIZPAK	UNIRAS™
	CUECHART®	ISSCO®
	THE DATA CONNECTION®	ISSCO
	GRAFMAKER™	Precision Visuals
	PicSure®	Precision Visuals
	SAS/GRAPH®	SAS Institute
	SPSS GRAPHICS™	SPSS
	TELL-A-GRAF®	ISSCO
Mechanical Engineering	TELLAPLAN®	ISSCO
	UNIEDIT	UNIRAS
	ANVIL-4000®	Manufacturing & Consulting Services
Structural/Architectural Engineering	ANSYS®	Swanson Analysis Systems
	PATRAN™	PDA Engineering
Mapping	GT/STRU DL	GTICES Systems Lab, Georgia Tech
	B-MAP	Zycor
	BIZMAP	UNIRAS
	CONTOURING SYSTEM	Precision Visuals
	CPS-1/G™	Radian Corp.
	GEOMAP	Geographic Systems
	GEOPAK, GEOINT, KRIGPAK, UNIMAP	UNIRAS
	QUIK™	Sierra Geophysics
	SEISPAK, GIMAGE	UNIRAS
	Z-MAP	Zycor
Application Software Development	DI-3000®	Precision Visuals
	DI-TEXTPRO®	Precision Visuals
	DISSPLA®	ISSCO
	DISSPLAY DYNAMICS	ISSCO
	DISSPLAY GKS OPTION	ISSCO
	GK-2000®	Precision Visuals
	METAFILE	Precision Visuals
	PLOT 10® IGL	Tektronix, Inc.
	PLOT 10 GDI	Tektronix, Inc.
	PLOT 10 GKS	Tektronix, Inc.
	PLOT 10 STI	Tektronix, Inc.
	PLOT 10 TCS	Tektronix, Inc.
	RASPAK, UNIGKS, G-IMAGE	UNIRAS

In Brief

Monochrome Hard Copy Output for IBM® 5080

Users of IBM's 5080 Graphics System can now use Tek's 4634 Option 13 Imaging Hard Copy Unit to print photographic-quality monochrome output from the 5080 display screen.

The 4634 uses a photographic process and fiber optic technology to produce high-resolution copies with at least twelve distinct shades of gray and continuous toning. The process used by the 4634 results in minimal terminal tie-up: once you've pressed the Hard Copy button, the 4634 needs less than nine seconds to capture the screen image. First copies are ready in 26 seconds, and successive copies of the same image are produced in only 12 seconds.


An optional four-channel multiplexer allows up to four workstations to share the 4634.

UTek™ Update

A number of enhancements have been added to Version 2.2 of UTeK, the Unix™-based operating system of the 6130 Intelligent Graphics Workstation. New features include performance improvements to the UTeK kernel and the Pascal, C and Fortran compilers; modifi-

cations to the system administration interface; and a configurable kernel that lets you install and remove device drivers without recompiling the kernel. In addition, the UTeK/A Auxiliary Utilities package now includes AT&T's System V.2 Graphics Package, a collection of numerical and graphic commands that build and edit data plots and hierarchy charts.

If you're a 6130 user and your Software Subscription Service is current, you

should have received your UTeK 2.2 update. If you need to update your SSS, contact your Tektronix field office. 

UTek is a trademark of Tektronix.

Programming Tips

Why Use Segments, or Coax Host to RS-232 Host, and Vice Versa

by Peter Keep

Terminals Division Marketing
Tektronix, Inc.
Wilsonville, OR

A lot of "homebrew" applications exist today for the 4107A and 4109A terminals. Most of these take advantage of the extensive local graphic processing power available via use of the terminal's segment operations. Normally, these programs, when written correctly, run without modification on the even higher powered 4120 Series terminals. The 4120s, however, have long held an advantage over the 4100s: the ability to save a locally created picture. The picture can be saved to the optional 4120 disk, to the connected host, or out the PPI ports. This was accomplished with both the PLOT command and the SAVE command.

With the release of the "A" Series for the 4100, this situation is changed. Now, the 4100A Series terminals include a SAVE command, and the valid destinations for the PLOT command include HO: (host) as well as P0: and P1:. And note: HO: is


either RS-232 or COAX, depending on how the terminal is equipped and set. One use of this ability is to "hack" graphic files from the RS-232 hosts to the coax hosts.

This little trick is really quite simple. Let's say you are using a CX4107, and have an IBM* host on your coax port and a VAX*, Prime, or other ASCII host on the RS-232 port. Your pictures are created with a program on the VAX, but you need them on the IBM (or vice-versa). Here's how to do it:

1. If your picture file or program does not use segments, open one up using the terminal setup key:
*SGOPEN 1
2. Send your picture to the terminal from one host using the appropriate host commands.
3. Close the segment:
*SGCLOSE
4. Change the HOSTPORT:
*HOSTPORT COAX
5. Set the EOF to a value that the receiving host will recognize as being the end-of-file.
6. Have the second host open a file; then the host tells the terminal to send all

visible segments with either the SAVE or PLOT commands. The choice will depend on the associated parameters you want sent with the picture. Further details on these parameters are available from your local Tek representative.

If you are trying to save your picture back to the program that created it, that's even easier. Since SAVE and PLOT are host commands, that host program can draw the picture, open a file, send the appropriate command, wait for the EOF, and then continue on.

This ability is available in the 4106A, 4107A, and 4109A, as well as the companion CX series. 

*IBM is a trademark of International Business Machines Corporation. VAX is a trademark of Digital Equipment Corporation.

Using 4115B/4120 Series Pseudo Devices Without a Parallel DMA Link

by Ron Bryant

GWD Engineering
Tektronix, Inc.
Wilsonville, OR

Even if your host computer does not support RS-422 communications, you may still be able to obtain significant increases in the throughput of graphics data to 4115B/4120 Series workstations equipped with pseudo devices.

A pseudo device is a software routine resident in the terminal that behaves like a mass storage device. It will only accept and produce sequential binary data. It must be copied to and from, and will only stop the transfer upon receipt of the unique End-of-file String.

There are four pseudo devices in the 4115B/4120 Series Workstations: PX:, CM:, SG:, and DS:. Each has a specific

operational function:

- PX: Can receive or transmit eight-bit pixel data or special run-length encoded format pixels.
- CM: Can receive or transmit machine RGB color map values.
- SG: and DS: Can receive or transmit segment list (retained or unretained) picture processor opcodes and data.

As with the file system, a command sequence such as: "Copy HO: to SG:" will receive segment data over the RS-232 port and send it directly to the segment list. Other valid copy modes involving the pseudo devices are:

- Copy HO: to DS:
Copy HO: to CM:
Copy HO: to PX:

This list shows the direction of data flow being from the host to the terminal. You can change the direction by reversing the order of the HO: and pseudo device name.

The following subroutines generate

4115B/4120 Series Picture Processor opcodes and display list opcodes for use in RS-232 communications with the pseudo devices in the terminals. An example of system calls for altering the tty to allow the transmission of binary data is also provided.

Configuring the Workstation for Handling Eight-Bit Data

As with the segment list, the data path invoked is from the RS-232 port directly to the pseudo device consumer firmware routine. Since these pseudo devices contain attributes resembling file-structured devices, the data source must produce binary data. This is only possible with the RS-232 port when the parity bit is combined with the seven-bit ASCII to form an eight-bit byte. You can do this by using the PARITY DATA command in Setup.

Since binary data will be transmitted from the host, which will include all control characters, some other Setup changes and precautions are required. The control characters DC1 and DC3 will not pass

through to the pseudo devices if terminal flagging is enabled. Flagging must be set to 'none' or 'DTR/CTS' (if supported by the host). If No Flagging is the only mode available, then precautions must be taken to ensure that the terminal will not be overrun with data. The problem is reduced, if not eliminated, by adjusting the communications "que size" within the workstation to a "large" value. "QUE 1000" will be large enough to allow for most graphics. If segment list data is being transferred and the content of the data will draw rectangles, or scales and rotates, then the queue size may have to be increased more.

At this point the terminal is ready to receive the binary data.

Host Considerations For Eight-Bit Data Transmission

Adjusting the host to transmit binary or eight-bit data may or may not be a problem on your particular host. Some operating systems filter the outgoing data to ensure that only seven-bit ASCII data will be sent to the terminal. Characters outside of this range may be filtered by the operating system and never even reach the terminal. It may be possible to use the host "Set" programs to alter the terminal driver routine on the host, or a special program/utility can be used to open the terminal port in the "Special Mode." This will be the most difficult part of the job. Host transmission of eight-bit binary data over normal RS-232 lines to a terminal is not a common request of system programmers. In most cases, however, the host hardware can do it; it is currently using the eighth bit for parity rather than data.

On a DEC® system running Unix*, opening the RS-232 terminal driver in the "raw" mode will allow the transmission of the eight-bit data. By setting the Flags member of the tty structure to include the "raw" mode, binary data can be transmitted to the terminal. Just a warning: working with the host once the driver has been put in this mode can have nasty side effects, because in this mode the special control characters lose their meaning—it may not be possible to interrupt the program in the middle of its execution.

Terminating the Copy Command


Once the terminal driver has been set up to accommodate the transmission of eight-bit data (only the output needs to be changed to this mode), the next task is to tailor the program to transmit the terminal device-dependent data. Pixel copies are by far the easiest. The computed array of pixels (as an unsigned character array) is simply transmitted after the copy command (Copy HO: to PX:) is issued to the terminal. Stopping or terminating the copy will involve the selection of an EOF string with a Setup command such as: 'EOFS "The End"'. The terminal will recognize this unique string from the host and will terminate the copy command.

Using the Pseudo Device Explicitly

To make effective use of color map updates on the 4115B/4120 Series, use the binary mode to transmit machine RGB values directly to the color map. The sequence is much like the pixel copy, except that it is invoked with the command "Copy HO: to CM:". The values to be sent

are the RED, GREEN and BLUE values from index 255 to index 0. For example, sending 100,100,100 will set index 255 to be 100% red, 100% green and 100% blue. This will result in index 255 being white, since the values are additive.

To use the binary path with the segment list and non-retained segment list, it is necessary to construct the terminal Picture Processor Display List in the host's memory. For the most part, the Picture Processor List has a one-byte opcode followed by one or more bytes of data. For example, an absolute Move16 is a hexadecimal '1F' followed by four bytes of data—two bytes for the X value and two bytes for the Y value.

There are 36 opcodes supported in the 4115B; they include moves and draws in 8, 16, 32 and 40-bit format. The opcodes use relative coordinates from the current beam position rather than absolute coordinates. Sample code for generating the display list and Picture Processor opcodes is available from your local Tektronix field office. 

*DEC is a registered trademark of Digital Equipment Corp. Unix is a trademark of AT&T Bell Laboratories.

Tips Welcome

Tekniques welcomes the submission of Programming Tips from readers. If you'd like to share a tip you've developed, or if you have a topic you'd like to see explored, write to:

Jan Rowell
Tekniques
Tektronix MS 63-635 P.O. Box 1000
Wilsonville, OR 97070-1000

IDG Customer Training Workshops

A week-long customer training workshop on UTEK™, the Unix™-based operating system of the 6130 Intelligent Graphics Workstation, will be offered beginning March 3 at Gaithersburg, Maryland, and May 12 at Dallas, Texas. Previous Unix experience is not required. Attendees will become proficient in the use of the file system, the Bourne and "C" shells, the "vi" screen editor, and selected UTEK utilities. Basic system administration tasks are also covered, including system security, file system maintenance, and the use of the "sysadmin" interface.

The following workshops are also offered:

Tekniques
Vol. 9 No. 2

PLOT 10 TekniCAD

Mar. 10-14	Gaithersburg, MD
Mar. 24-28	Santa Clara, CA
Apr. 14-18	Dallas
May 5-9	Boston
May 12-16	Santa Clara, CA

Local Programmability

Apr. 14-18	Boston
------------	--------


PLOT 10 IGL

Apr. 28-May 2	Dallas
---------------	--------

For further information, or to register for a workshop, you can call collect (503) 642-8953; or write to:

Customer Training, MS 54-076
Tektronix, Inc.
P.O. Box 500
Beaverton, OR 97077-0001

Classes in Beginning and Advanced Smalltalk are offered by Tek's Artificial Intelligence Machines Group. For information on Smalltalk classes, call Jeff McKenna, (503) 685-2943, or write to:

Jeff McKenna
MS 60-405
Tektronix, Inc.
P.O. Box 1000
Wilsonville, OR 97070-1000 

104

Tektronix
COMMITTED TO EXCELLENCE

TEKTRONIX, INC.
Information Display Group
Tekniques
Mail Stop 63-635
P.O. Box 1000
Wilsonville, Oregon 97070

BULK RATE
U.S. POSTAGE
PAID
TEKTRONIX, INC.



Address Correction Requested—Forwarding and Return Postage Guaranteed.