AN ABSTRACT OF THE THESIS OF

GEORGE DAVID ROSE (Name) for the degree of MASTER OF SCIENCE (Degree)

in MATHEMATICS (Major) presented on October 15, 1971 (Date)

Title: INTERACTIVE GRAPHICS IN A TIME SHARING ENVIRONMENT

Abstract approved: Professor Harry Goheen

The storage tube computer graphics terminal provides high resolution, high speed, and comparatively low cost graphics. Oregon State University currently supports four such terminals under the OS-3 time sharing system. The terminals are used for a wide variety of applications in the physical sciences and engineering.

This thesis describes TEKPLOT and CALTEK, two FORTRAN compatible subroutine libraries tailored to the operation of the graphics terminal. TEKPLOT is used for plotting directly on the terminal. CALTEK, a companion package, can be used to automatically generate a scaled hard copy of terminal output on a Calcomp incremental plotter.

A brief description of the implementation strategy used to support the terminal within the operating system environment is also included.

Interactive Graphics in a Time Sharing Environment

by

George David Rose

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

June 1972

APPROVED:

Redacted for Privacy

Professor of Mathematics

Redacted for Privacy

Acting Chairman of Department of Mathematics

Redacted for Privacy

Dean of Graduate School

Date thesis is presented October 15, 1971

Typed by Yvonne L. Ley for George David Rose

ACKNOWLEDGEMENTS

The author wishes to thank Professor Harry Goheen for his assistance, his encouragement, and his friendship. Dr. Larry Hunter's continuing openness to new areas of inquiry together with his keen interest in computer graphics have made this project a most pleasant undertaking. Finally, I wish to thank Dr. Joel Davis for his careful review of a draft of this thesis.

TABLE OF CONTENTS

INTERACTIVE GRAPHICS IN A TIME SHARING ENVIRONMENT

INTRODUCTION

The development of the direct view, bi-stable cathode ray storage tube has ushered in a family of new computer terminals. Characterized by high resolution, high speed, and comparatively low cost graphics, such terminals are significantly affecting the field of computer graphics.

The principal advantage of the direct view terminal is also its principal weakness. Since image storage is accomplished directly on the display phosphor, the image need never be refreshed, and flicker ceases to be a problem. The disadvantage of this approach is that selective erasure of the screen is not possible.

In most cases, total erasure and redrawing is not a serious drawback. A plot that changes in time can be displayed as a sequence of frames without loss of continuity. Using time lapse photography, presentation quality films have been taken that eliminate apparent redrawing time altogether.

Oregon State University currently supports four Tektronix T-4002 direct view terminals as a part of the OS-3 time sharing system.

While the graphics terminal may be driven at data rates ranging from Teletype speed (110 Baud) to 9,600 Baud, it is our experience that speeds below 300 Baud seriously try the patience of the user. In fact, only 1,200 Baud does the drawing rate for complicated plots approach a subjectively satisfactory level.

Since one 1,200 Baud terminal has a data rate equivalent to almost 12 Teletypes, it may appear that a few graphics terminals would swamp a time sharing system. The reasons why this does not occur will be discussed.

The major portion of this paper describes a FORTRAN compatible subroutine library tailored to the graphics terminal. The library is in two parts. TEKPLOT is a set of 32 subroutines used to drive the terminal. CALTEK is a companion library that can be used to automatically make a hard copy of terminal output on a Calcomp incremental plotter. Implicit in the TEKPLOT/CALTEK fast preview device for incremental plotter output as well as conventions for accomplishing interactive graphics on the incremental plotter.

The graphics system is intended to be both effective and easy to learn. A primary goal has been to enable an average FORTRAN user to accomplish interactive computer graphics without extensive study or preparation.

The TEKPLOT/CALTEK libraries were designed and implemented by myself.

The Hardware Terminal

The Tektronix T-4002 graphics terminal allows graphics input and graphics output in addition to alphanumeric operation. Excluding the graphics capability, the terminal functions as a Teletype. Information exchange between the computer and the user is provided through a keyboard and the visual display. The keyboard includes the full USASCII character set.

The visual display is 8.25 by 6.5 inches. In alphanumeric mode, the display has 39 lines of 85 characters each. In graphics mode, the display has 1,024 by 762 addressable points.

Five distinct modes of hardware operation are selectable:

1) alphanumeric mode
2) point plot mode
3) linear interpolate mode
4) incremental plot mode
5) graphics input mode

Hardware generated alphanumeric characters may assume two sizes. Each size may be displayed in two type fonts.

In point plot mode, a single point is intensified when addressed. In linear interpolate mode, a smooth vector is drawn between consecutively addressed points. Both of these modes require a graphics location to be addressed absolutely by sending an (x,y) coordinate address in each case.

In incremental plot mode, the terminal functions as an incremental plotter. Each plotted increment is relative to the current position and in one of eight directions.

Graphics input mode allows the user to select an (x,y) coordinate position by positioning a crosshair graphics cursor. The cursor is moved by manipulating a joystick adjacent to the terminal. When the cursor is in position, striking any key on the keyboard causes both cursor coordinates and the ASCII code of the character to be sent to the computer.

Detailed information about the hardware specifications of the terminal is available from Tektronix, Inc.[1]

Implementation

The graphics terminal is one of a number of available terminal types supported by OS-3, the Oregon State Open Shop Operating System.[2] OS-3 is a time sharing operating system for the Control Data 3300 computer. The system was developed at Oregon State University and has been in use during the past four years.

OS-3 supports up to 60 on-line users together with batch and remote batch users. The system is widely used on the OSU campus and is tied to other colleges in the state through a communications network.

The operating system allows all users to have on-line access to line printer, punch, magnetic tape, high speed paper tape, and Calcomp plotter facilities. User files may be retained on-line in disk storage. Seven major languages are supported as well as a host of special purpose languages, utility programs, etc.

By coupling the graphics terminal to the OS-3 time sharing system, instead of tying it to a dedicated mini-computer, the entire spectrum of languages and I/O facilities is made available to the graphics terminal user. The remainder of this section describes the manner in which this coupling was accomplished.

OS-3 uses a PDP-8 computer as a data concentrator and multiplexer for all telecommunications I/O (see figure 1). For every active user, the system creates a program status area (PSA) in the 3300. The PSA contains all user status information necessary to keep track of a user's program. Associated with each PSA is a variable length character storage buffer.

Character output generated by a user's program is collected in the 3300 and saved in the character buffer local to his PSA. On a clock driven basis, the system gathers waiting output characters from all PSAs and tags each character with a terminal number identifier. The tagged characters are then packed into a block and output

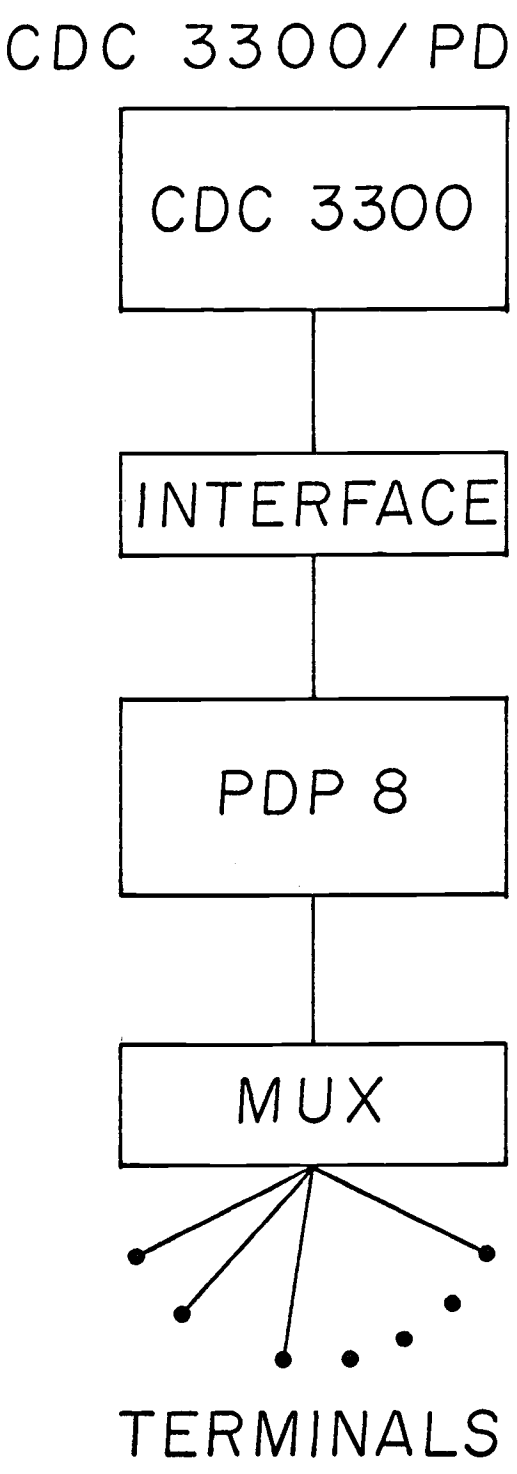


Figure 1

to the PDP-8 (see figure 2). The PDP-8 unpacks this block and, using the tag information, associates each character in sequence with its intended terminal. Terminals in use are constantly polled, and waiting output characters are transmitted when each terminal is ready to receive.

Character input to a user's program proceeds in a converse fashion. The PDP-8 collects terminal characters, tags them, and packs them into blocks. The character blocks are sent on a clock driven basis to the 3300 where they are unpacked and associated with their appropriate PSA.

It should be clear from the foregoing discussion that most of the overhead required to handle character I/O consists of gathering, tagging, and packing characters in one computer, and unpacking and scattering these characters in the other computer. Since a graphics terminal operating at 2,400 Baud can exhaust almost 24 characters during a single Teletype character time, each graphics terminal exchanges characters in its own block (figure 2). In particular, a high speed character block exchanged between the 3300 and the PDP-8 contains characters that are all intended for a single terminal. This strategy eliminates the tagging, gathering, and scattering overhead required for an ordinary Teletype block. In all

TELETYPE BLOCK

Block Type	Count
Device #	Character
Device #	Character
Device #	Character
Device #	Character

HIGH SPEED BLOCK

Block Type	
Count	Terminal #
Character	Character
Character	Character
Character	

Figure 2

other respects, a graphics terminal is handled as a Teletype by the operating system.

The Plot Packages

The TEKPLOT and CALTEK subroutine libraries are sets of FORTRAN compatible subroutines that are tailored to the operation of the graphics terminal.[3] TEKPLOT is used for plotting directly on the terminal while CALTEK plots on the Calcomp incremental plotter. Both packages have identical calls.

TEKPLOT is stored as a part of the standard FORTRAN library. Any library routine referenced by a user in his calling program will be automatically loaded and linked to his program during execution. CALTEK is stored as a separate library file. If a user wishes to make an incremental plotter copy of a plot that was viewed on the graphics terminal, he can load his binary program together with the CALTEK library. This causes the loader to satisfy its library requests first from CALTEK, and then from the standard library. In this way the user can obtain a hard copy of a plot that was created using TEKPLOT without changing his program in any way. Since the incremental plotter bed is larger than the visual display, CALTEK also allows an enlargement of the original plot, if desired.

In the event that the user wishes to load both TEKPLOT and CALTEK routines simultaneously, a variant edition of the latter, called CALTEKV, is included in the standard library. CALTEK and CALTEKV are identical with one exception: all subroutine names in CALTEKV have an appended V as the last character. By loading routines from TEKPLOT and CALTEKV, the user may work interactively at the graphics terminal and concurrently generate a Calcomp copy. This approach enables the Calcomp plotter to be used for interactive graphics.

The graphics subroutines divorce the hardware characteristics of the terminal from the logic of plotting. With these routines, the user may think of the display screen as a 'window' into two dimensional space. A window is a rectangular field of vision. Only that portion of a plot contained within the current window will be displayed by the plot package; the rest of the plot will be suppressed. The current window will automatically default to the whole screen unless otherwise defined by the user. The user defines a window by supplying window coordinates as parameters in an appropriate subroutine call. Under software control, this window can be translated to any particular area of interest, rotated any arbitrary number of degrees, and used as a magnifier.

A window may be defined by coordinate definition within a program or by user manipulation of the joystick after visual inspection of a plot.

Any plot that extends beyond the current window limits is automatically clipped at the window margin. Lines passing through the window are plotted with their true slope; plotting external to the window is suppressed entirely.

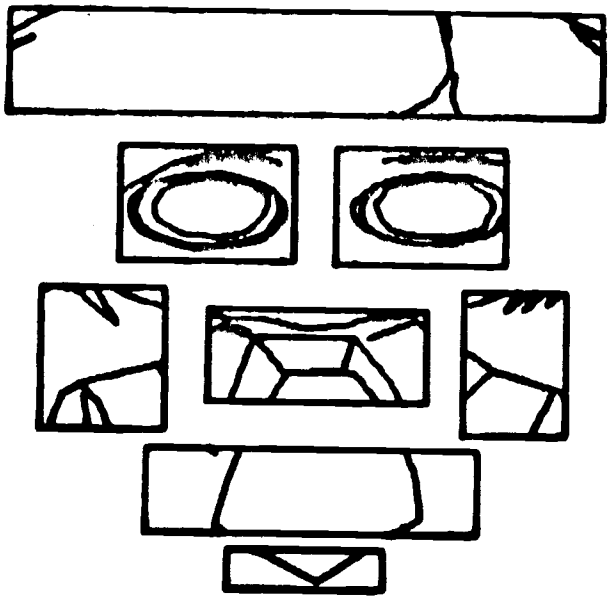
Since the window size is variable, it is possible to create write protected areas on the display screen. Windowing may also be used to display multiple plots simultaneously or bring together physically disparate portions of a plot for comparison.

Figure 3 shows an example of the windowing capability. In this example, a plot is first drawn in full. Then, using successive windows, the plot is redrawn in full several times. Each redrawing displays only that portion of the plot included within the current window. TEKPLOT and CALTEKV were used together in this example to allow a graphically input window specification.

The graphics routines also provide a mechanism for:

1) controlling the terminal environment: erasing the screen, determining character size and character font, etc.

GIRL WITH WINDOWING



GIRL

FIGURE 3

2) plotting in any of the available modes, either absolutely or relative to the current position
3) drawing axes, setting scaling, size, rotation
4) performing graphics input of either points or regional coordinates
5) drawing software generated characters of any desired size or orientation

A summary of the graphics library is given at the end of this paper.

Inherent in the graphics routines is the notion of the current graphics location. The current location is the last (x,y) point addressed by the user while in a graphics output mode. Some of the graphics routines update the current location while others leave it unchanged. For example, if the user reverts to alphanumeric mode in the middle of a plot, writes some labeling information, and then returns to one of the graphics output modes, the plot resumes from the point where it left off; no conscious initialization is required on the part of the user.

The plotting routines all display data in the currently selected mode; the mode must be set prior to calling a plotting routine. For example, a call to PLOT while in vector mode will cause a vector to be drawn from the current location to the (x,y) point specified

as a parameter to PLOT. This latter (x,y) coordinate then becomes the new current location. In a similar way, a call to PLOT while in point plot mode causes a point to be written at the (x,y) position specified in the call; this (x,y) point then becomes the new current location.

The graphics output routines scale, rotate, and translate all (x,y) information prior to output. In an inverse manner, graphically input points are translated, inversely rotated, and unscaled before they are delivered back to the user's program.

The software character generator should also be mentioned here. While the graphics terminal has an alphanumeric mode of operation, the Calcomp plotter does not. Hence, hardware generated characters written on the screen in alpha mode using TEKPLOT will not be automatically carried over to the Calcomp plotter when using CALTEK. If the user wishes to label plotted output such that alphanumeric annotation can be carried over to Calcomp output, the characters must be drawn. The software character generator is used for this purpose.

Several versions of the software character generator exist. One version causes character size and orientation to respond to global scaling and rotation factors. A second version allows local scaling and rotation factors

to be supplied as a part of the subroutine call. Dependent upon the version chosen, the current location is updated or retained upon return from the call.

The remainder of this paper is a discussion of the programming considerations involved in the implementation of TEKPLOT and CALTEK.

TEKPLOT Implementation

In this section a few salient issues pertaining to the TEKPLOT implementation strategy will be discussed. All routines were written in the assembly language of the CDC 3300. When loaded together, they occupy approximately one page of core memory (2K).

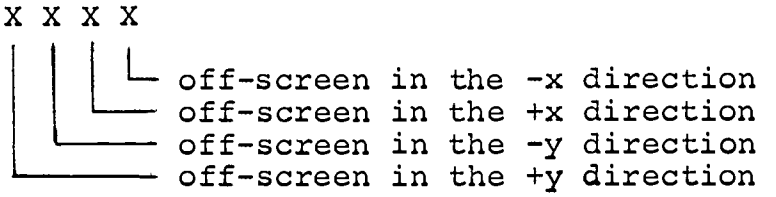
The terminal, unlike an I/O device, cannot be interrogated for status information. For this reason, the graphics routines keep track of requisite status information in a terminal status area. Such items as the current location, mode, last location, window size, scaling, rotation, etc., are all retained in the terminal status area.

A key algorithm in TEKPLOT is the one used to inhibit plotting that is external to the current window. If the user is in point plot mode and the current point is outside the window, plotting is suppressed and no further computation is required.

If the user is in vector mode and the last (x,y) or the current (x,y) are outside the window, it is necessary to determine whether the vector connecting these points passes through the window of vision. If so, then the visible portion of the vector must be drawn. Two problems exist:

1) to determine whether the vector from the last (x,y) point to the current (x,y) point intersects the window. This determination must be made with minimal computation since it is a calculation that will be frequently made.
2) to handle the z-axis properly. The T-4002 hardware controls the z-axis by blanking the first vector drawn after the output of a particular control character, viz. an ASCII GS.

The first problem is solved by forming an off-screen flag for each point plotted. This flag contains four bits whose meanings are indicated in the following diagram:



If

F = off-screen flag of the current (x,y) point

and

F' = off-screen flag of the last (x,y) point then plotting can be suppressed if the following condition holds:

$$F\&F' \neq 0 \quad (*)$$

where & denotes the logical AND operator. Figure 4 depicts this situation in greater detail.

Condition (*) is a sufficient, but not a necessary, test for suppression of plotting. If the last (x,y) and the current (x,y) coordinates are both outside the window, and if F&F' = 0, then a further test is made to determine whether the vector connecting these coordinates does, in fact, intersect the window.

The second problem can be broken down into the following cases:

	outside the window	inside the window
case 1:	last (x,y) current (x,y)	
case 2:	last (x,y)	current (x,y)
case 3:	current (x,y)	last (x,y)
case 4:		last (x,y) current (x,y)

Let L = the last (x,y) coordinate

C = the current (x,y) coordinate

and $\overline{LC}$ be the directed line segment from L to C.

OFF-SCREEN FLAGS

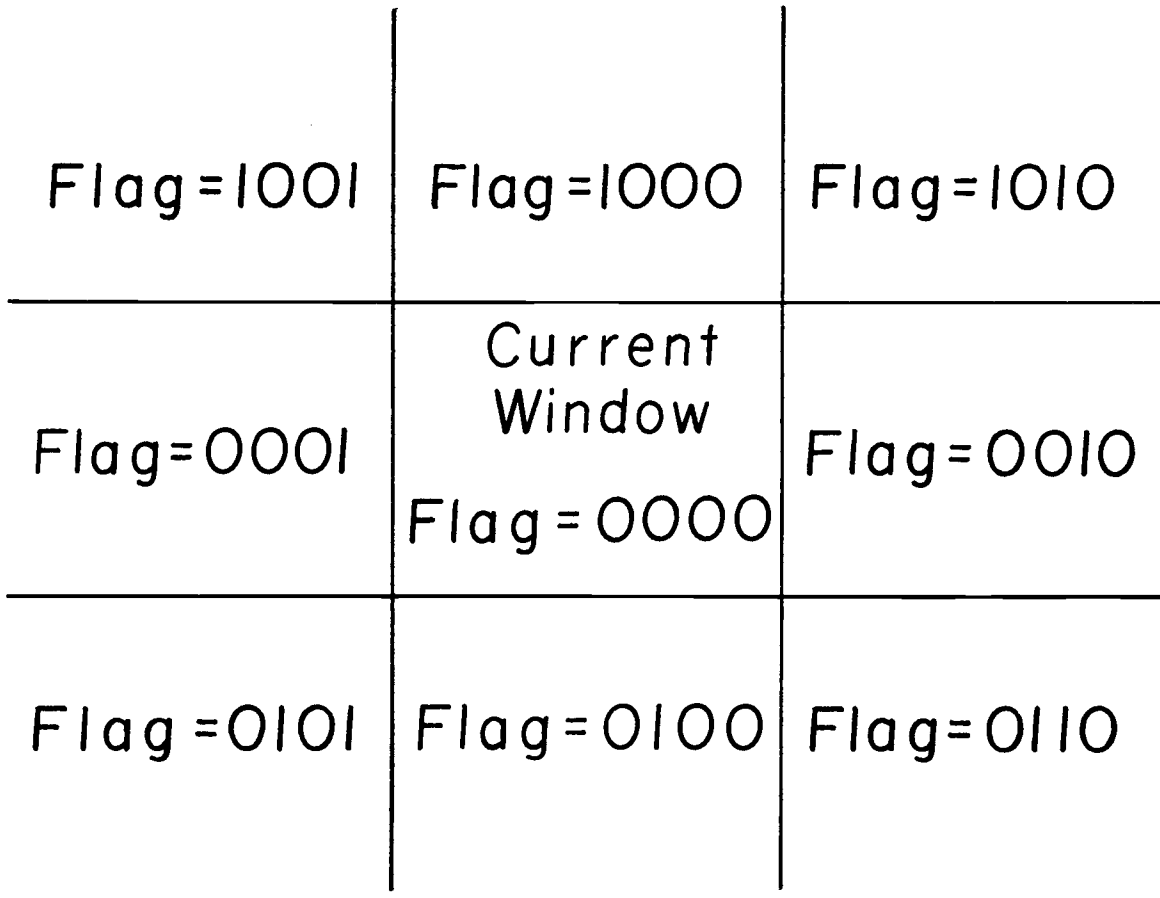


Figure 4

Then,

case 1: If condition (*) holds, then suppress plotting. If condition (*) does not hold but $\overline{LC}$ does not intersect the window, then suppress plotting. If $\overline{LC}$ does intersect the window, then output a GS to blank the next vector and plot to the point where $\overline{LC}$ first intersects the window. Then handle as case 3.

case 2: Output a GS and plot to the point where $\overline{LC}$ first intersects the window. Then handle as case 4.

case 3: If the user specified plotting with the z-axis on, then plot to the point where $\overline{LC}$ intersects the window. If the user specified z-axis off, then output a GS prior to plotting.

case 4: If the user specified z-axis, then plot to point C. If the user specified z-axis off, then output a GS prior to plotting.

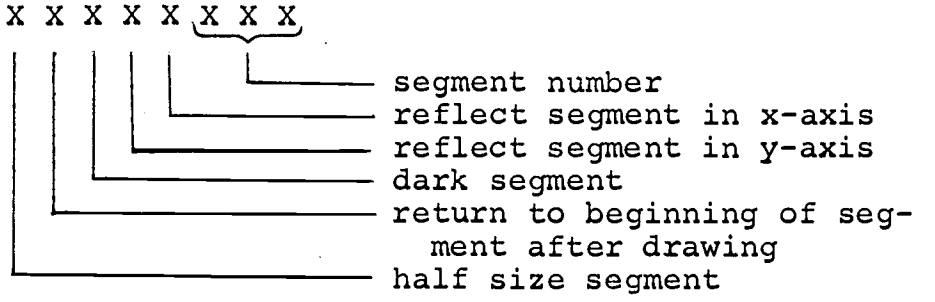
After plotting is completed, the current (x,y) coordinate and its off-screen flag are saved as the last (x,y) coordinate and flag. In addition, if the user calls the WINDOW subroutine to change the window size, it is necessary to recompute the last off-screen flag.

To conclude this section, the software character generator will be briefly discussed. Each character in

the TEKPLOT character set is defined as a variable length list of directed line segments. To draw a string of characters, the WRITEX routine calls the character generator, CHARGEN, with each character in turn. CHARGEN looks up the character definition in a table and calls the segment generator, SEGEN, with each segment. Finally, SEGEN qualifies the segment from a set of tag bit definitions and then calls the DELTA routine to draw the segment. This process is represented below:

WRITEX → CHARGEN → SEGEN → DELTA

Each segment is defined by a segment number together with five tag bits. Tag bit definitions are as follows:



A character, then, is just a set of segment definitions in the form shown above. There are only eight possible segments, these being a sufficient set to draw any of the characters in the TEKPLOT character set. The 'dark segment' bit in the segment definition is used to control the z-axis or plotter pen.

Since segment lengths and directions are not specified absolutely, a change in the scale or rotation will result in a corresponding change in character size and orientation. Character size and orientation is handled in an automatic way by the DELTA routine which treats the character segments like any other (Δx, Δy) increment.

CALTEK Implementation

The internal logic for CALTEK is similar to TEKPLOT. The CALTEK routines were written in assembly language, and the entire package is several hundred words longer than TEKPLOT.

Apparent linear interpolation on the Calcomp plotter is achieved using an algorithm to approximate a straight line by a series of incremental moves.[4] Point plotting is accomplished by positioning the pen, and then dropping and lifting it in place. (This mode does not work out very well with ball point plotter pens.)

The principal difference between the two packages is in the strategy for handling the pen. Two issues arise:

1) plotting modes on the graphics terminal are hardware selectable. Mode differences on the Calcomp plotter must be simulated by proper pen control.

2) it is easy for a user, who is only aware of his terminal output, to write programs that will generate unnecessary pen movement commands. This situation can arise, for example, by constantly reinitializing the plotting mode. While this type of overhead turns out to be relatively unimportant on the T-4002, it is costly on the Calcomp where one pen movement time is equivalent to twenty increments.

The CALTEK routines optimize pen movement by keeping a brief history of 'penup' and 'pendown' commands. If the user causes consecutive penup, pendown or pendown, penup commands to be generated, and if the current pen status coincides with the second of these two commands, then both commands are deleted from the output buffer.

This optimization strategy has the awkward effect of purging the user's buffer of points that were plotted in point plot mode (since each point is drawn by first positioning, then dropping and lifting the pen). To circumvent this occurrence, a special internal forcing command is generated to lower the pen when the user is in point plot mode.

Conclusion

Four direct view graphics terminals are currently supported under the Oregon State University time sharing system. To assess user reaction to graphics at different data communication rates, terminals are implemented at 110, 300, 1,200, and 2,400 Baud respectively. It is anticipated that several additional terminals, operating at 1,200 Baud, will be added during the coming year.

This paper has described the terminal implementation strategy as well as two software support packages tailored to terminal operation. TEKPLOT is used for plotting directly on the terminal while CALTEK, a companion package, can be used to automatically generate a scaled hard copy of terminal output on a Calcomp incremental plotter.

The terminal, together with TEKPLOT/CALTEK, is being used for graphics applications in physics, chemistry, biology, oceanography, and civil, industrial, and electrical engineering at OSU. Applications include both research and computer assisted instruction.

Summary of Routines

Utility Routines

	(TEKPLOT)	(CALTEK)
ERASE	erases the screen.	spaces the paper forward if any plotting has occurred since the last call to ERASE.
PAGE	moves the alpha cursor to the upper left corner of the screen.	positions the pen to the upper left corner of the implicit screen.
TEKPAUSE	delays the user's program until he sends a character.	has no effect.
BYENOW	returns the user to the control mode of the operating system.	empties the plotter buffer, lifts the pen and positions it out of harm's way, and returns the user to the control mode of the operating system.
PLOTLUN	has no effect.	chooses a logical unit for plotter output if the default unit is not desired.
DOUBLE	sets double character size for hardware generated characters.	has no effect.
ITALICS	sets italics font for hardware generated characters.	has no effect.
NORMAL	resets double size and/or italicized hardware character mode.	has no effect.

Mode Routines

ALPHAS	places the terminal in alphanumeric mode.
VECTORS	places the terminal in linear interpolate mode and positions to the current location.
POINTS	places the terminal in point plot mode at the current location.

Axis Routines

SCALE	sets scaling and linear translation factors.
ROTATE	performs axis transformation.
SIZE	sets terminal:plotter enlargement ratio.
AXIS	draws visible axes.
WINDOW	defines a rectangular window external to which all plotting will be suppressed.

Character I/O Routines

CHIN	performs character input.
CHOUT	performs character output.

Plotting Routines

PLOT	plots to a specified absolute (x,y) coordinate in the current mode.
DELTA	plots a (Δx,Δy) relative to the current location in the current mode.
INCPLOT	performs incremental plotting.

Graphics Input Routines

GRIN	reads a graphically input (x,y) coordinate.

GRIDSET	defines a grid for use with GRID.
GRID	reads a graphically input (row,column) coordinate relative to the current grid setting.

Software Character Generator

WRITEX	generates software characters with global scaling and rotation.
WRITEY	generates software characters with local scaling and rotation.
XLATE	translates a number to a character string suitable for input to WRITEX.

BIBLIOGRAPHY

1. Tektronix, Inc. T-4002/4802 Graphics Computer Terminal User's Manual. Beaverton, Oregon. 1970

2. Meeker, J., Crandall, R., Dayton, F., and Rose, G. OS-3: The Oregon State Open Shop Operating System. American Federation of Information Processing Societies Conference Proceedings. 34; 241-248. 1969.

3. Rose, G. TEKPLOT/CALTEK. Oregon State University Computer Center. Corvallis, Oregon. 1970. ccm-70-13.

4. Stockton, F. XYMOVE Plotting. Collected Algorithms from the Communications of the Association for Computing Machinery. Algorithm 162, 162-P-1-0.