1976

Graphics subsystem for a terminal with conic section capabilities.

Nelson, States Lee

Monterey, California. Naval Postgraduate School

GRAPHICS SUBSYSTEM FOR A TERMINAL WITH CONIC SECTION CAPABILITIES

States Lee Nelson

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

GRAPHICS SUBSYSTEM FOR A TERMINAL
WITH
CONIC SECTION CAPABILITIES

by

States Lee Nelson

June 1976

Thesis Advisor: G. M. Raetz

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) Graphics Subsystem for a Terminal with Conic Section Capabilities	5. TYPE OF REPORT & PERIOD COVERED Master's thesis; June 1976
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s) States Lee Nelson	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE June 1976
	13. NUMBER OF PAGES 88
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Graphics
Graphics Subsystem
Conic Section
Interactive Graphics
Graphics
Graphics Display
Software
Computer Display
Computer Software

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The aim of this thesis is to design and implement a graphic subsystem for a graphic display terminal with conic section capabilities. The display terminal with its associated peripheral hardware is connected to a PDP-11/50 computer.

The basic concepts and design principles of graphic systems are discussed. A brief description is given of the particular hardware configuration of the Conographic-12 display system as

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (cont.)

installed at the Naval Postgraduate School. Concepts incorporated in the designed implementation are explained. Recommendations are included for possible future extensions of the system's capabilities.

Graphics Subsystem for a Terminal with Conic Section Capabilities

by

States Lee Nelson
Lieutenant, United States Navy
BSMA, Illinois Institute of Technology, 1971

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
JUNE, 1976

ABSTRACT

The aim of this thesis is to design and implement a graphic subsystem for a graphic display terminal with conic section capabilities. The display terminal with its associated peripheral hardware is connected to a PDP-11/50 computer.

The basic concepts and design principles of graphic systems are discussed. A brief description is given of the particular hardware configuration of the Conographic-12 display system as installed at the Naval Postgraduate School. Concepts incorporated in the designed implementation are explained. Recommendations are included for possible future extensions of the system's capabilities.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I wish to express my deepest appreciation to my thesis advisor, Gary M. Raetz, for his effort, guidance, and patience. His knowledge and assistance have been invaluable in developing and installing the CONIE System.

I also wish to thank my son Kristofer for his many distractions.

I. INTRODUCTION

The purpose of this thesis is to design and implement a graphic subsystem for a graphic display terminal with conic section capabilities. A literature search was performed to determine the desirable features of a graphic system and plausible means of implementation. The resulting implemented design provides the applications programmer with a library package of general purpose subroutines which construct the commands for the Conographic-12 Interactive Graphic Display System.

The Conographic-12 Display System, as installed at the Naval Postgraduate School, is a display terminal interfaced with a PDP-11/50 computer. The terminal system includes an alphanumeric keyboard, special program function keyboard, joystick, and extended symbol capability. This terminal draws points, straight lines, characters, circles, arcs, ellipses, and conic curves. As a result, major reductions in the amount of data required to draw a given picture are made possible. Data compression means that pictures take less CPU time to manipulate, significantly less storage space, and substatially less time to transmit from computer memory to display device [3].

II. INTERFACE DESIGN CONCEPTS

A. UNIVERSAL GRAPHIC LANGUAGE

One of the major obstacles in defining a general graphic subsystem is that, even among the experts, there is no consensus of opinion on what its capabilities should be. Many can point out various functions which they would like to see in the system, but each person desires different functions. The one point on which most do agree is that, at this time, simplicity is more important than efficiency in a graphic language. Once a language has been conceived and can be used by a wide variety of people, then the efficiency of the language can be extended.

Wiseman says it is a mistake to say that "because we are dealing with graphical things, there is a need to invent a new language, or perhaps to embellish an existing one"[4]. What is needed is a rich language able to express and manipulate complex data structures. The task of generating pictures as a byproduct is easy; the complexity in graphics arises from the problems of data description and task coordination.

William Newman claims we already have a general purpose graphic language in FORTRAN IV. Most proposed graphics packages are based on FORTRAN IV, even though it is not a particularly good language [6]. It has to be considered

universal because everyone uses it [4]. Extrapolating from this arguement any language with input/output capabilities could be considered a general purpose graphic language if there are no particular requirements that it must meet.

Several of the experts believe that it is impossible to define a universal graphics language only because graphics is not understood well enough. For instance, we do understand arithmetic well enough so nobody questions the usefulness of a universal arithmetic language. Rosenfeld suggests that "we can inculcate graphical thinking at a stage as early as arithmetic thinking is now inculcated, and let the next generation design the universal graphic language"[4].

Wells considers set theory a universal graphic language. Unordered sets appear in many languages. The ordered structures, lists and arrays, in programming languages are just the sequences of set theory; the procedures are the functions of set theory; and lines are naturally considered to be sets of points. Reliance should be, wherever possible, on established general mathematical symbolism and conventions, rather than considering particular extensions to computer-oriented languages that have been developed with economics of machine operation as a prime consideration. Languages should first be designed for man-to-man communication, with concern for the efficiency or implementation on various machines coming later [4].

Since there are different areas of graphics with

different requirements perhaps it would be more practical to define these areas and their requirements. The different areas of graphics include: off-line input, recognition, and scene analysis; interactive input; plotting output; line drawings; and shaded pictures. Most applications will be concerned primarily with one of these graphic areas although some applications may deal with a combination of areas, such as the field of interactive design which uses a combination of interactive input and line drawings. The specific needs of each area should be researched and defined. Then a language can be designed for each area rather than attempting to design one language for all graphics areas.

B. GOALS OF A GRAPHIC SYSTEM

Graphic terminals are particularly well adapted to the many problems which do not yield exact solutions by a given algorithm. Such problems often require the user's direct intervention at different stages in the computation, man-to-machine dialogue, in order to obtain satisfactory solutions. The efficiency of such a dialogue demands a rapid, synthetic representation of results as well as a simple, flexible, and convenient means of intervention at the user's disposal. Graphic terminals allow quick visualization of diagrams, graphs, or any picture which would be far more

easily interpreted than the usual tabulated numerical results. Moreover, these terminals, being equipped with joystick, hardware zoom, and alphanumeric and function keyboards, become accessible to non-programmers [2].

Ease of use is one of the principal goals of a graphic system. Consequently, a graphic system requires certain properties. First, concepts linked with the use of graphic terminals have to be clearly set apart. These basic concepts must appear in the structure of the application program, making the system a "guide for good programming" [2]. Second, the number of instructions related to interactive graphics should be as small as possible. The goal is to make the user's manual a handy reference guide rather than an in-depth study.

Few general purpose graphic systems would be able to satisfy completely all the requirements of a particular application. Therefore, an applications programmer is frequently obliged to develop a specific system which contains the features needed to resolve his own particular problem. In the graphic subsystem for the Conographic-12 an attempt is made to provide as broad a variety of general purpose graphic tools as is needed to give the applications programmer access to all of the terminal's capabilities. The modular nature of subroutine calls facilitates modification. This includes the deletion of functions which through utilization are determined to be non-useful, as well as the

addition of new functions which are deemed desirable. For this reason, a list of subroutine calls becomes a more efficient means of implementating an interface than the "syntactic extension of an existing language" recommended by Boullier [2].

C. SOFTWARE DESIGN

The software support required to utilize graphic devices such as the Conographic-12, include machine-dependent transmission, interrupt handling routines, and processors for user-oriented languages. The design of graphic programs begins with an evaluation of other software support available to the computer system and the degree of interaction required to incorporate the graphic devices [5].

A multi-level design approach is used in the development of graphic software. There are generally considered to be four levels. Level one includes subroutines which are dependent on the hardware characteristics of the graphic device. They perform primitive operations. Software developed at this level provides the basic tools for programming at the higher levels. Level two programs combine the facilities available from the operating system with the hardware-oriented programs of level one. This combination provides the programs and user-oriented languages to be used

at level three. Software development at level three includes user-oriented subroutines which are capable of organizing buffer and file management routines, and can formulate graphical instructions and data transmission. Level four progamming is application progams which are directed toward the support of the non-programmer [5].

The lowest level of this multi-level design is illustrated in this project. It involves efforts to provide an efficient interface for the handling of signals between the computer and the graphic devices. Progressively higher levels of software components can then be developed to increase the productivity and utility of the total system.

The programming activities of this project are primarily in the category of level one software design - primitive, hardware-oriented subroutines. At the most primitive level, subroutines are used to produce the object code which is required to generate images. Examples include code to plot a point, display a character, draw a vector, etc. At this level all the parameters are device-oriented. That is, the subroutines work with the parameters required by the Conographic-12. Using other subroutines, the screen point coordinates and slope values are converted into the device-oriented curve parameters. User-oriented subroutines convert the user's coordinates to screen coordinates.

III. HARDWARE ENVIRONMENT

The Conographic-12 Interactive Graphic Display System includes raster scan CRT display, alphanumeric keyboard, program function keyboard, joystick, extended symbol capability, and hardware zoom. The coordinate system inherent in the hardware is an integer address space with the x and y coordinate values ranging from -4096 to 4095. The hardware includes scale and offset registers which provide the capability of varying the portion of address space which is displayed on the screen. When the scale factor is one and the offsets are zero, the x coordinates from zero to 2047 and the y coordinates from zero to 1536 are visible on the screen.

The storage tube terminal has the basic capabilities of displaying vectors and alphanumeric characters. Once written the display remains visible until erased. It is not necessary to continually regenerate the output data or refresh the screen. The picture is actually drawn on the target of a video memory utilizing random X-Y deflection. The target of the video memory is electronically scanned to produce the video signal which the raster scan CRT displays. The generator, which supplies the X-Y deflection signals, draws conic curves in response to digital parameters supplied from a computer [3].

The alphanumeric keyboard has the ability to generate

all two hundred fifty-six 8-bit character codes. The keyboard generates one hundred twenty-eight ASCII character codes, and there is an additional key to force the parity bit to zero or one. This extends the range of codes that may be generated to the full two hundred fifty-six. The alphanumeric keyboard can be enabled/disabled by a command from the user's program in the host computer. When enabled, striking a key generates an interrupt, passing the keycode to the computer. For the operation of various additional keys provided on the keyboard, reference the Conographic-12 System Reference Manual [3].

The program function keyboard consists of thirty-two function keys which produce codes that may be interpreted by a program in the host computer. The resulting action is totally dependent on the program.

The joystick, when enabled by a program, provides cursor positioning control directly to the user. A program may obtain this positional information by reading the graphic display terminal's registers. Other information obtainable from the registers are: offset, scale, dash pattern, dash frequency, and selected intensity.

The hardware zoom is completely independent of any program. It provides the user with the capability of dynamically scaling and positioning the displayed picture in order to view a select portion.

The extended symbol capability provides the possibility

of having specialized fonts or special sets of symbols for particular applications. A standard ASCII character font is provided. It is implemented in a 4K ROM which is addressed as a particular range of the terminal's memory. When the terminal is in alphanumeric mode and an 8-bit character code is received, execution of the instructions located at the address indicated by the sum of the contents of the font base register plus the the 8-bit character code is performed. In the standard ASCII character font ROM, this first instruction is "jump to symbol subroutine" [4]. Thus the first 256 locations in the font's address space comprise a jump table. The symbol subroutine is composed of short graphic commands, thus all capabilities of the terminal are accessable except the hardware dash pattern and selectable intensity. All addressing within a font is relative to the value in the font base register. The address space of a font is limited to 4096 locations starting with the address in the font base register [4]. Special fonts and symbol sets may be implemented in the same manner as that just described for the standard ASCII character set. The particular font to be utilized is selected by setting the font base register. The default setting of the font base register is the ASCII font.

IV. CONIE - CONOGRAPHIC INTERFACE EFFORT

The Conographic-12 Interface is a hierarchical set of subroutines which can be divided into four general areas: Direct Graphics, Virtual Graphics, Alphanumerics, and Control Routines. The routines in all these areas provide the user interface. This is where the functions as conceived by the user are translated into functions which are acceptable to the computer. To provide ease of use, every operation which is functionally different is referred to by a seperate subroutine call. Each subroutine call and its arguements are described in Appendix A of this thesis.

A. INITIALIZING AND TERMINATING

To ensure a proper starting sequence, an initializing routine is provided and must be called before using any other CONIE routine. The initial state is defined by the following four requirements: Cursor and beam are at the HOME position(lower left-hand corner); Terminal is in graphic mode; The scaling is set to unity and rotation and offset are at zero; Character size is set to two, "normal" condition. This initializing routine opens files; therefore, it must not be recalled prior to calling the terminating routine. The terminating routine closes the files.

B. DIRECT GRAPHICS

Direct graphics utilizes the coordinate system provided by the Conographic-12 hardware. Included are scaling and offsetting. The facilities for utilizing the conic section capabilities are provided. These include routines to draw curves given the points and slopes uniquely describing the particular curve desired. Curves may be uniquely defined by: two points and their associated slope (see figure 1), three points and the slope associated with one of these points (see figures 2,3 and 4), or three points where the slope at the intermediate point is equal to the slope of the chord joining the endpoints (see figure 5). Also included are routines for drawing circles, ellipses, and quarters thereof. These figures may be specified by their center and length of axes, or their center and one endpoint of each axis (see figure 6).

C. VIRTUAL GRAPHICS

To escape the size constraints of the terminal screen, the concept of virtual coordinates was implemented. The virtual coordinate system is an imaginary surface bounded only by the range of the floating point numbers of the computer system being used. The user may construct a drawing

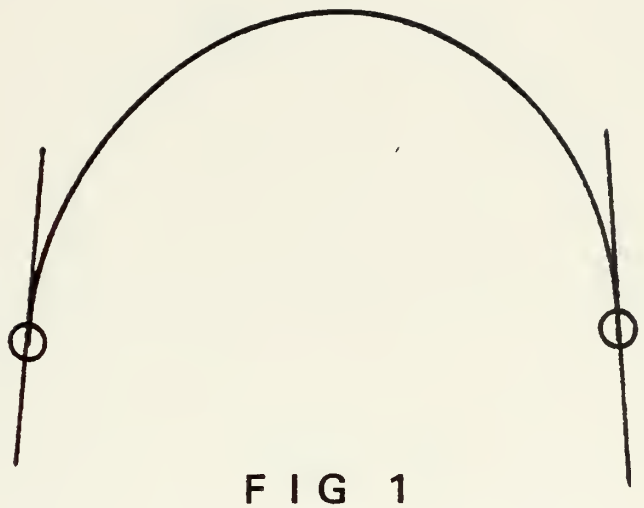

FIG 1

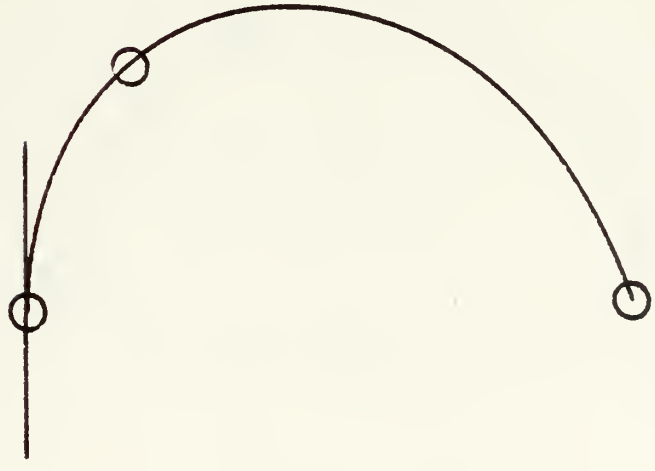

FIG 2

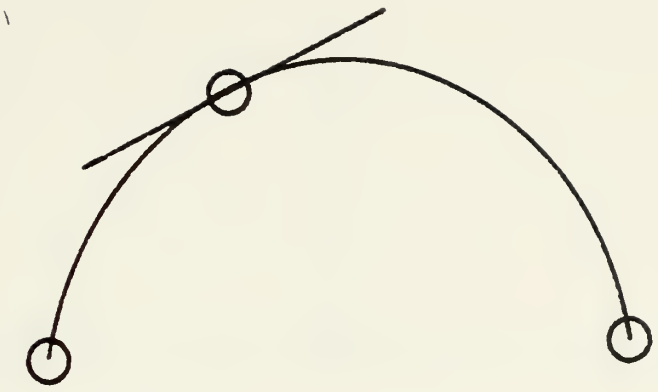

FIG 3

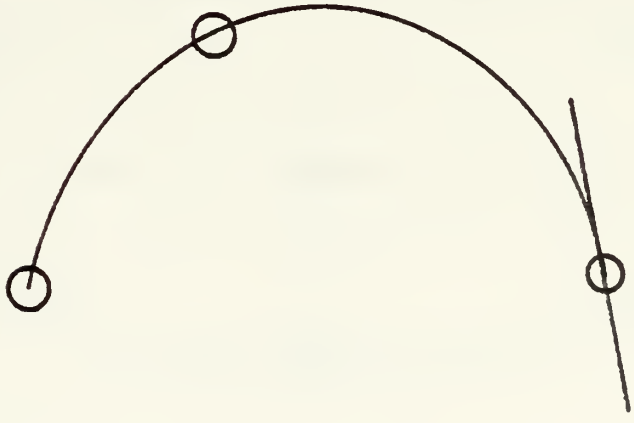

FIG 4

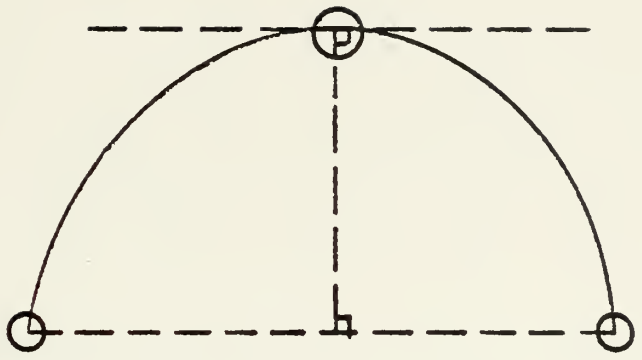

FIG 5

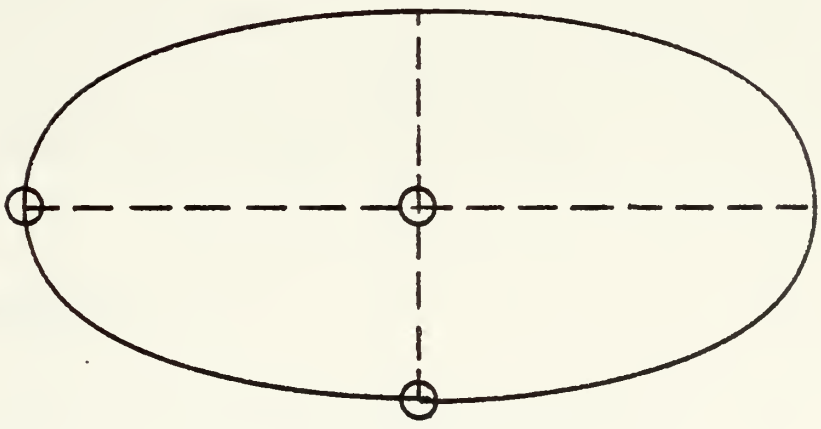

FIG 6

in any section of the virtual coordinate system that is desired. The units of the system have no physical dimensions associated with them. Such a dimension may be arbitrarily assumed at the user's convenience. Before constructing the drawing, the user specifies which portion of the virtual coordinates system to be viewed by calling a routine.

The virtual coordinate system facilities are similar to those available with direct coordinates. There is one notable exception: Virtual coordinates permit distortion - that is, the X and Y units may be of different sizes. This capability is not available with direct coordinates. The distortion capability precludes permitting lengths as parameters for routines (see figure 7). Therefore, the routines available with virtual coordinates are only a subset of those available with direct coordinates.

D. ALPHANUMERICS

Capability is provided for sending a string of characters. A character string, as defined in the "C" language [7], is any number of characters, excluding the null character. However, the null character must be included at the end of the string to indicate termination of the character string. A string of characters will normally be interpreted

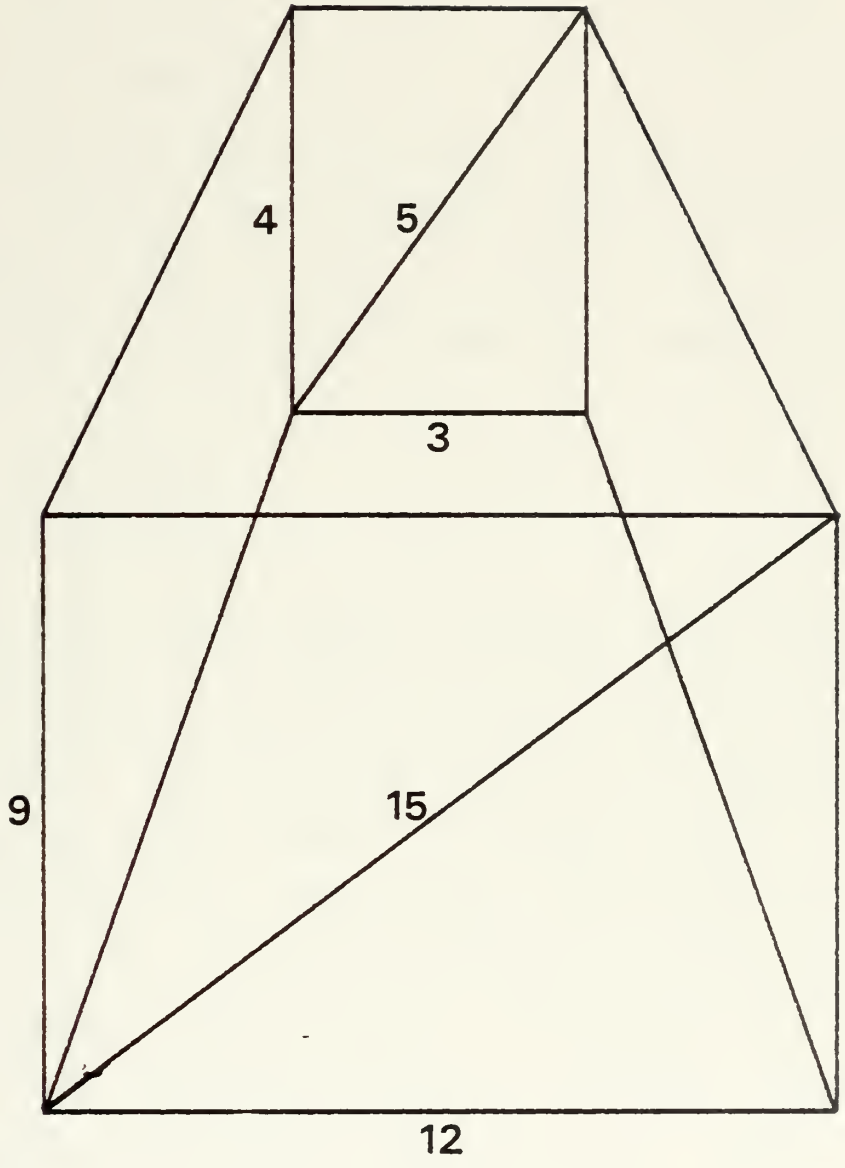

FIG 7

by the Conographic-12 as belonging to the ASCII character set, unless a different set of symbols has been defined and the font base register has been loaded to point to this other set of symbols.

There are two modes for sending a string of characters to the terminal. The difference between Template Mode and Alphanumeric Mode is only a scale factor of eight; alphanumerics are drawn scaled down by a factor of eight.

E. CONTROL ROUTINES

There are various control routines which provide for such functions as re-initialization; erasure; ringing the keyboard bell; selecting dash patterns and frequencies; and setting scale factor, angle of rotation, center of rotation, and reflection bits. Other control routines provide for the reading of the terminal's registers, keyboard characters, and the program function keys. A control routine which utilizes the recursive property of the "C" language allows large complex structures to be built up in modular fashion from small simple structures [1]. At the lowest level of complex structures, the extended symbol capabilty can be utilized to further simplify the building of structures.

V. RECOMMENDATIONS

An assembler or very powerful editor for building extended symbol sets would be a very useful extension to the present implementation. This is needed to assist the programmer in building the jump table with the proper addresses.

The addition of higher level routines to plot arrays, and to provide curve-fitting and data smoothing is recommended. The capability of plotting arrays is convenient in a variety of applications. Curve-fitting and data smoothing combine to take full advantage of the Conographic-12's capabilities so that data compression can be fully exploited.

The different graphic display terminals at the Naval Postgraduate School have uniquely individual hardware features. However, in time the capabilities provided by their software systems will achieve a high degree of overlap. At that time it will be feasible to design and implement a pre-processor extension to the "C" language [7] which will perform the translation to the particular routines for the graphic display terminal selected at compile time. A program can then be written which can be executed on any or all of the different graphic display terminals simply by recompiling [8].

VI. CONCLUSIONS

The Conographic-12 Interface Effort has been designed and implemented. As a result of the programming effort involved in the production of CONIE, the user can fully utilize the Conographic-12 and its associated peripheral hardware.

APPENDIX A - USER'S MANUAL

A. INTRODUCTION

This manual lists in alphabetical order the subroutine calls used for sending graphic commands to the Conographic-12 terminal. Prior to the execution of a program utilizing these calls, the terminal should be turned on and placed in non-local mode utilizing the switch provided on the terminal's alphanumeric keyboard.

The various graphic facilties available are curve drawing, scaling, offsetting, automatic dash pattern for drawing, reflections, variable intensity, and rotation. Also available is the ASCII character font and the capability of defining additional fonts.

To utilize the various subroutines provided, a program is written using cgstart() as the first graphic call and cgfini() as the last. The particular picture or subpictures to be displayed are best described in functions defined by the user utilizing the drawing commands such as cgline, cgmove, cgelosg, or cgcr22. There are a variety of additional drawing commands from which the user may select those best suited to his needs. Character output intermixed with drawing commands can be accomplished simply by using the cgalph command. If the convention is followed of utilizing a seperate function for each subpicture, then the positoning

of each subpicture can be easily accomplished using commands like cgdrawr. This also simplifies the defining of subpictures since each has its own set of coordinate axes. Thus the user can define independent objects for display and later accomplish the relative positioning.

An illustration of how cgdrawr may be utilized is in the segmentation of a display object into its component parts, which may have their own symmetry. A simple example would be the wheel of a vehicle. For such a case a user would write a function defining a wheel, perhaps with coordinates (0,0) for the center hub of the wheel. The vehicle body could be defined in a seperate function with the coordinates (0,0) possibly at one of the corners of the vehicle. When drawing the complete vehicle with wheels, the sequence of cgmove followed with cgdrawr would be utilized. The call to cgmove would be used to position the beam at the end of the vehicle's axle. The command cgdrawr would be called with the wheel function and the proper coordinates of the wheel's center as defined by the wheel function as parameters. This would effectively place the wheel on the end of the vehicle's axle.

The parameters of scale, rotation, and reflection for cgdrawr provide additional flexibility. With the scale parameter available, a user need not concern himself with making the units uniform for all subpictures. Instead he may utilize the most convenient units for each seperate

subpicture and then scale appropriately. The rotation parameter enables the user to define a subpicture at the angle where its coordinates are most easily expressed, and then rotate the subpicture to provide the proper display. The reflection parameter provides the capability for exact 90 degree rotations and subpicture inversions. The center for all of these transformations is the beam position when the cgdrawr command is executed.

Control of the manner in which input coordinate data is to be interpreted is provided by the subroutines cginpfm and cgvcord. The routine cginpfm permits the user to declare whether his coordinate data will be expressed in normal Cartesian coordinate format or in coordinates relative to the beam position. The routine cgvcord enables the user to specify a floating point coordinate system. The information the user provides cgvcord is used to set the parameters necessary for converting floating point coordinates into the integer coordinate system of the screen.

Additional subroutines provide control of various hardware features, some of which are as simple as ringing the keyboard bell. Others provide the input of information from the various auxiliary devices, including the alphanumeric and program function keyboards. The dash pattern and selectable intensity are controlled by the routines cgdpfis and cgdsi. The routine cgdpfis is utilized to set the pattern and intensity available. The routine cgdsi is

inserted at appropriate places in the drawing commands to turn on or off the utilization of the dash pattern or selected intensity. The ability to erase either the full screen or a select object is provided by cgerase. In order to erase a particular object, cgerase is called for the appropriate setting. The commands for drawing the object are the re-executed to accomplish the erasure, after which cgerase is recalled to reset to normal drawing.

The user's program is compiled, specifying that the Conographic-12 library be included. The routines available in this library and the parameters they require are detailed on the following pages. The naming convention followed is that all start with the letters "cg" so that the user can easily avoid collision of names. Also all routines using virtual coordinates start with "cgv" to signify that their coordinate parameters are virtual and must therefore be floating point numbers.

B. SAMPLE PROGRAM

```
// This program draws a simple sketch of a ferris wheel

// This function defines the labels to be placed on each car
char *num(i)
int i;
{
   switch(i){
   case 0: return("0");
   case 1: return("1");
   case 2: return("2");
```

```
    case 3: return("3");
    case 4: return("4");
    case 5: return("5");
    case 6: return("6");
    case 7: return("7");
    }
}

//This function draws the car and calls the label function
car(i)
int i;
{
    cgcirc(0,0,5);
    cgelpsa(0,-50,0.0,20,50);
    cgmove(-5,-80); // Position for the label
    cgalph(num(i)); // Output the label
}

int i;

// This function draws the arms of the ferris wheel
// and positions the cars at their ends
arm(a,b)
int a,b;
{
    cgelpsa(0,0,0.0,a,b);
    cgmove(a,0);          // Move to the end of the arm
    cgdrawa(1.0,0.0,0,0,0,car,i,0);
    // Set an upright orientation for the car
}

// This is the main program which initiates and
// positions the picture on the screen
main()
{
    cgstart("ferris");
    cgcrsr(0);
    cgmove(1023,800);
    for(i=0;i<8;i++)    cgdrawr(1.0,0.39269,i,0,0,arm,500,50);
    // Draw the arm in all its eight reflections
    cgfini();
}
```

NAME:

cgalph - emit character string in alphanumeric mode

SYNOPSIS:

```
cgalph(a)
char a[];
```

DESCRIPTION:

This routine places the terminal in alphanumeric mode and then sends a character string to the terminal. It returns the beam position to a point one line down from the previous position and returns the terminal to the previous graphic mode.

a - null-terminated character string to be sent to Conographic-12

SEE ALSO:

cgtplt

NAME:

cgbell - ring keyboard bell

SYNOPSIS:

cgbell()

DESCRIPTION:

This routine rings the keyboard bell.

NAME:

cgcirc - draw a circle

SYNOPSIS:

```
cgcirc(a,b,t)
int a,b,t;
```

DESCRIPTION:

This routine draws a circle when given the center point and the radius. The coordinates of the center point are shown by a,b. The radius is defined by the value of t.

a - x coordinate of center

b - y coordinate of center

t - radius of circle

SEE ALSO:

cgcirq

NAME:

cgcirq - draw circle quarters

SYNOPSIS:

```
cgcirq(a,b,dr,q,se)
int a,b,dr,q;
float *se;
```

DESCRIPTION:

This routine draws quarters of circles when given the values for the center point, a point on the circle, the direction in which to draw from this point, and the number of quarters to be drawn. The present position of the beam defines a point on the circle. The center point is given by the coordinates a,b. The direction in which to draw from the present position is defined by dr. The number of quarters to be drawn is shown by q, where one quarter is assumed and 3 draws a full circle. The address where the value of the ending slope will be returned is defined by se.

a - x coordinate of center

b - y coordinate of center

dr - direction of drawing
- 0: draw clockwise
- 1: draw counterclockwise

q - number of additional quarters
- 0: one quarter only
- 1: semi-circle
- 2: two additional quarters, total three quarters
- 3: full circle

se - address of floating point variable where ending slope can be returned

SEE ALSO:

cgcirq

NAME:

cgcrsr - turn cursor on or off

SYNOPSIS:

```
cgcrsr(o)
int o;
```

DESCRIPTION:

This routine enables/disables the joystick. When enabled, the user can directly control the positioning of the cursor by manipulating the joystick.

o - action flag
0: disable the joystick
1: enable the joystick

SEE ALSO:

cgrdcsr

NAME:

cgcr22 - draw a curve given 2 points and 2 slopes

SYNOPSIS:

```
cgcr22(x,y,sb,se)
int x,y;
float sb,se;
```

DESCRIPTION:

This routine draws a curve when given two points on the desired curve and the slopes associated with these two points. The beginning point of the curve is the present position of the beam. The coordinates x,y define the endpoint. The slope of the beginning point is defined by sb. The slope of the endpoint is given by se.

x - x coordinate of the endpoint

y - y coordinate of the endpoint

sb - slope of the curve at the beginning point

se - slope of the curve at the endpoint

DIAGNOSTICS:

The ambiguous case of parameters sb=se outputs a message to the standard output, moves to the specified endpoint, and returns a negative value.

SEE ALSO:

cgcr3b, cgcr3e, cgcr3i, cgcr3m, cgvcr22, cgvcr3b, cgvcr3e, cgvcr3i, cgvcr3m

NAME:

cgcr3b - draw a curve given 3 points and beginning slope

SYNOPSIS:

```
cgcr3b(g,h,x,y,sb,se)
int g,h,x,y;
float sb,*se;
```

DESCRIPTION:

This routine draws a curve when given three points and the slope at the beginning point. The beginning point is defined by the present position of the beam. The coordinates of the intermediate point are given by g,h. The endpoint is shown by the coordinates x,y. The slope at the beginning point is defined by sb. The address where the value of the ending slope will be returned is shown by se.

g - x coordinate of intermediate point

h - y coordinate of intermediate point

x - x coordinate of the endpoint

y - y coordinate of the endpoint

sb - slope of the curve at the beginning point

se - address of floating point variable where ending slope can be returned

DIAGNOSTICS:

If the parameters describe a curve with an inflection point, a message is output to the standard output, the beam is moved to the endpoint, and a negative value is returned.

SEE ALSO:

cgcr22, cgcr3e, cgcr3i, cgcr3m, cgvcr22, cgvcr3b, cgvcr3e, cgvcr3i, cgvcr3m

NAME:

cgcr3e - draw a curve given 3 points and ending slope

SYNOPSIS:

```
cgcr3e(g,h,x,y,se)
int g,h,x,y;
float se;
```

DESCRIPTION:

This routine draws a curve when given the coordinates of three points and the slope at the endpoint. The beginning point is defined by the present position of the beam. The intermediate point is shown by the coordinates g,h. The endpoint is defined by the coordinates x,y. The slope at the endpoint is given by se.

g - x coordinate of intermediate point

h - y coordinate of intermediate point

x - x coordinate of the endpoint

y - y coordinate of the endpoint

se - slope of the curve at the endpoint

DIAGNOSTICS:

If the parameters describe a curve with an inflection point, a message is output to the standard output, the beam is moved to the endpoint, and a negative value is returned.

SEE ALSO:

cgcr22, cgcr3b, cgcr3i, cgcr3m, cgvcr22, cgvcr3b, cgvcr3e, cgvcr3i, cgvcr3m

NAME:

cgcr3i - draw a curve given 3 points and intermediate slope

SYNOPSIS:

```
cgcr3i(g,h,x,y,si,se)
int g,h,x,y;
float si,*se;
```

DESCRIPTION:

This routine draws a curve when given three points and the slope at the intermediate point. The beginning point is defined by the present position of the beam. The intermediate point is given by the coordinates g,h. The endpoint is defined by the coordinates x,y. The slope at the intermediate point is given by si. The address where the value of the ending slope will be returned is shown by se.

g - x coordinate of intermediate point

h - y coordinate of intermediate point

x - x coordinate of the endpoint

y - y coordinate of the endpoint

si - slope of the curve at the intermediate point

se - address of floating point variable where ending slope can be returned

DIAGNOSTICS:

If the parameters describe a curve with an inflection point, a message is output to the standard output, the beam is moved to the endpoint, and a negative value is returned.

SEE ALSO:

cgcr22, cgcr3b, cgcr3e, cgcr3m, cgvcr22, cgvcr3b,

cgvcr3e, cgvcr3i, cgvcr3m

NAME:

cgcr3m - draw a curve given 3 points with the intermediate at the relative maximum

SYNOPSIS:

```
cgcr3m(g,h,x,y,se)
int g,h,x,y;
float *se;
```

DESCRIPTION:

This routine draws a curve when given three points where the slope at the intermediate point is equal to the slope of the chord joining the endpoints. The beginning point is defined by the present position of the beam. The intermediate point is shown by the coordinates g,h. The endpoint is defined by the coordinates x,y. The address where the value of the ending slope will be returned is shown by se.

g - x coordinate of intermediate point

h - y coordinate of intermediate point

x - x coordinate of the endpoint

y - y coordinate of the endpoint

se - address of floating point variable where ending slope can be returned

SEE ALSO:

cgcr22, cgcr3b, cgcr3e, cgcr3i, cgvcr22, cgvcr3b, cgvcr3e, cgvcr3i, cgvcr3m

NAME:

cgdpfis - set dash pattern and frequancy and selectable intensity

SYNOPSIS:

```
cgdpfis(p,f,i)
int p,f,i;
```

DESCRIPTION:

This routine sets dash pattern, dash frequency, and selected intensity level. Draw commands are affected only if cgdsi has been called setting dash and/or intensity select.

p - 12 bit value defining dash pattern, 1 specifies intensification and 0 specifies blanking; for example, 07777 would be a solid line, 0 would be a blank line, 05252 would be short dashes, 077 would be long dashes.

f - 4 bit value (0-15), this controls the frequency of the dash pattern thus determining the length of the dash corresponding to each bit in the pattern (note: other scale factors do not affect the dash pattern)

i - 4 bit value (0-15), sets the grey scale level of the selectable intensity, 0 is dimmest, 15 is brightest, and 12 is the grey scale level of the normal intensity

SEE ALSO:

cgdsi

NAME:

cgdrawa - set new orientation for user function

SYNOPSIS:

```
cgdrawa(s,a,r,x,y,f,ac,av)
int(*f)(),ac,av,r,x,y;
float s,a;
```

DESCRIPTION:

This routine resets the orientation then calls the user's function with the two arguements. Scaling is by means of the Conographic-12 hardware registers therefore limited to the range of 1/64 to 63 63/64. Upon return the previous orientation is restored. The user's function f is defined by the user to draw whatever subpicture he desires. The parameters ac and av are passed to the user's function so that it may have parameters. If more than two parameters are desired, or non-integer parameters are desired, the two parameters provided can be used to pass a count and a pointer to an array of parameters.

s - scale to be applied

a - angle of desired rotation expressed in radians

r - reflection desired
- 0: no reflection
- 1: reflect about x-axis
- 2: reflect about y-axis
- 3: reflect about both axes, (pi rotation)
- 4: reflect about pi/4 line
- 5: pi/2 rotation counterclockwise
- 6: pi/2 rotation clockwise
- 7: reflect about 3/4 pi line

x - x coordinate which will correspond with present beam position and be the center of reflection and rotation

y - y coordinate which will correspond with present beam position and be the center of reflection and rotation

f - user defined function

ac - first arguement for user function

av - second arguement for user function

SEE ALSO:

cgdrawr, cgvdrwa, cgvdrwr

NAME:

cgdrawr - set orientation relative to present orientation

SYNOPSIS:

```
cgdrawr(s,a,r,xc,yc,f,ac,av)
int (*f)(),ac,av,r,xc,yc;
float s,a;
```

DESCRIPTION:

This routine sets a new orientation relative to the present orientation then calls the user's function. Thus the reflections, rotations and scaling are cummulative. For best results it is recommended that most scaling factors be 1, if the hardware scaling provided here is insufficient it is recommended that virtual coordinates be utilized. Upon return the present orientation is restored. The user's function f is defined by the user to draw whatever subpicture is desired. The parameters ac and av are passed to the user's function so that it may have parameters. If more than two parameters are desired, or non-integer parameters are desired, the two parameters provided can be used to pass a count and pointer to an array of parameters.

s - scale to be applied

a - angle of desired rotation expressed in radians

r - reflection desired
- 0: no reflection
- 1: reflect about x-axis
- 2: reflect about y-axis
- 3: reflect about both axes, (pi rotation)
- 4: reflect about pi/4 line
- 5: pi/2 rotation counterclockwise
- 6: pi/2 rotation clockwise
- 7: reflect about 3/4 pi line

x - x coordinate which will correspond with present beam position and be the center of reflection and rotation

y - y coordinate which will correspond with present beam

position and be the center of reflection and rotation

f - user defined function

ac - first arguement for user function

av - second arguement for user function

SEE ALSO:

cgdrawa, cgvdrwa, cgvdrwr

NAME:

cgdsi - select dash and/or intensity

SYNOPSIS:

```
cgdsi(d,i)
int d,i;
```

DESCRIPTION:

This routine determines whether susequent drawing commands are to be dashed and/or at normal or selected intensity. This routine is used in conjunction with cgpdfis which sets the particular pattern and/or intensity which is used.

d - drawing mode selection
- 0: set solid drawing
- 1: set dashed drawing

i - intensity selection
- 0: utilize normal intensity (12)
- 1: utilize the presently set selectable intensity

SEE ALSO:

cgpdfis

NAME:

cgelpsa - draw an ellipse given center and axes lengths and orientation

SYNOPSIS:

```
cgelpsa(a,b,r,mj,mn)
int a,b,mj,mn;
float r;
```

DESCRIPTION:

This routine draws an ellipse when given the center point, lengths of the two axes, and the orientation. The center point is given by the coordinates a,b. The length of the major semi-axis is defined by maj. The length of the minor semi-axis is defined by min. For orientation, the angle between the major axis and the horizontal is given by r and is written in radians.

a - x coordinate of center

b - y coordinate of center

r - angle of major axis orientation with respect to the x-axis expressed in radians

mj - major semi-axis length

mn - minor semi-axis length

SEE ALSO:

cgelpse, cgelpsa, cgvelpa

NAME:

cgelpse - draw an ellipse given center, one axis end and other axis length

SYNOPSIS:

```
cgelpse(a,b,xe,ye,lg)
int a,b,xe,ye,lg;
```

DESCRIPTION:

This routine draws an ellipse when given the center point, one axis endpoint and the length of the other semi-axis. The center point is shown by the coordinates a,b. The coordinates of the axis endpoint are given by xe, ye. The length of the other semi-axis is given by lg.

a - x coordinate of center

b - y coordinate of center

x - x coordinate of one semi-axis endpoint

y - y coordinate of one semi-axis endpoint

lg - length of other semi-axis

SEE ALSO:

cgelpsa, cgelpsa, cgvelpa

NAME:

cgelpsq - draw ellipse quarters

SYNOPSIS:

```
cgelpsq(a,b,xe,ye,q,se)
int a,b,xe,ye,q;
float *se;
```

DESCRIPTION:

This routine draws the specified number of quarters of an ellipse defined by its center and one endpoint of each axis. The ending slope is returned in se. Present beam position is taken as the endpoint of one axis and drawing starts in the direction of the other axis endpoint.

a - x coordinate of center

b - y coordinate of center

xe - x coordinate of other axis endpoint

ye - y coordinate of other axis endpoint

q - number of additional quarters
- 0: one quarter only
- 1: semi-ellipse
- 2: two additional quarters, total three quarters
- 3: full ellipse

se - address of floating point variable where ending slope can be returned

SEE ALSO:

cgelpsa, cgelose, cgvelpq

NAME:

cgerase - erase screen

SYNOPSIS:

```
cgerase(e,w)
int e,w;
```

DESCRIPTION:

This routine erases the screen. There are two means of doing this. Full screen erase or a selective erase can be used. In a selective erase the item to be erased is redrawn after this routine is called to effect the erasure.

e - erase screen flag
- 0: do not full screen erase
- 1: full screen erase

w - writting mode flag
- 0: return in normal writting mode
- 1: return in selective erasure mode

NAME:

cgfini - close files

SYNOPSIS:

```
cgfini(fn)
char fn[];
```

DESCRIPTION:

This routine closes all files that cgstart opens.

fn - filename used previously when calling cgstart

SEE ALSO:

cgstart

NAME:

cghome - initialize

SYNOPSIS:

cghome()

DESCRIPTION:

This routine performs display initialization with cursor in lower left-hand corner of screen, character size=2, picture scale=1, and rotation angle=0. Offsets are zeroed and terminal is in graphic mode.

SEE ALSO:

cgstart

NAME:

cginpfm - set input form

SYNOPSIS:

```
cginpfm(o)
int o;
```

DESCRIPTION:

This routine sets the mode in which coordinate parameters will be expressed. They may be expressed in either Cartesian coordinates (absolute), where coordinates are relative to a fixed origin, or in relative coordinates, where the beam position is used as the origin so that adjusting one point also adjusts all subsequent points.

o - selection indicator
- 0: absolute coordinates
- 1: relative coordinates

NAME:

cgkbd - enable/disable keyboard

SYNOPSIS:

```
cgkbd(e)
int e;
```

DESCRIPTION:

This routine turns the alphanumeric keyboard on and off. It is automatically called by cgrdkey to turn the keyboard on; however, the user must utilize this call if he desires the keyboard disabledafter reading.

e - action flag
0: disable keyboard
1: enable keyboard

SEE ALSO:

cgrdkey

NAME:

cgline - draw a line

SYNOPSIS:

```
cgline(x,y)
int x,y;
```

DESCRIPTION:

This routine draws a line from the present position of the beam to the endpoint defined by the coordinates x,y.

x - x coordinate of the endpoint

y - y coordinate of the endpoint

SEE ALSO:

cgvline

NAME:

cgmove - position cursor

SYNOPSIS:

```
cgmove(x,y)
int x,y;
```

DESCRIPTION:

This routine moves the beam to the indicated position without drawing.

x - x coordinate of the endpoint

y - y coordinate of the endpoint

SEE ALSO:

cgvmove

NAME:

cgpfk - enable/disable program function keys

SYNOPSIS:

```
cgpfk(e)
int e;
```

DESCRIPTION:

This routine turns the program function keyboard on and off. It is automatically called by cgrdpfk to enable the function keys, but must be called by the user when he desires to diable the function keys.

e - action flag
0: disable program function keys
1: enable program function keys

SEE ALSO:

cgrdpfk

NAME:

cgpoint - draw a point

SYNOPSIS:

```
cgpoint(x,y)
int x,y;
```

DESCRIPTION:

This routine draws a point at the specified position.

x - x coordinate of the point

y - y coordinate of the point

SEE ALSO:

cgvpoint

NAME:

cgrdacc - read accumulator

SYNOPSIS:

```
cgrdacc(ac)
int *ac;
```

DESCRIPTION:

This routine reads the Conographic-12 accumulator.

ac - address of an integer variable where the value from the accumulator can be returned

SEE ALSO:

cgrdreg

NAME:

cgrdcsr - read cursor position

SYNOPSIS:

```
cgrdcsr(x,y)
int *x,*y;
```

DESCRIPTION:

This routine reads the Conographic-12 present cursor position. An additional side effect is that the cursor is turned off prior to reading and turned on after reading.

x - address of an integer variable where the x coordinate can be returned

y - address of an integer variable where the y coordinate can be returned

SEE ALSO:

cgcrsr, cgrdreg

NAME:

cgrddsw - read display status word

SYNOPSIS:

```
cgrddsw(ds)
int *ds;
```

DESCRIPTION:

This routine reads the Conographic-12 display status word.

ds - address of an integer variable where the value from the display status word can be returned

The bits have the following meanings (low order bit 0)

```
 0:    italics
 1:    machine check
 3:    pfk requesting attention
 7,6:  mode of the display
   00  -   alphanumeric
   01  -   template
   10  -   short graphic
   11  -   long graphic
10-8:  reflection state
11:    graphic input flag
15:    TTY switch setting
```

SEE ALSO:

cgrdreg

NAME:

cgrdfbs - read font base

SYNOPSIS:

```
cgrdfbs(fb)
int *fb;
```

DESCRIPTION:

This routine reads the address in the font base register for the memory of the Conographic-12.

fb - address of an integer variable where the address from the font base register can be returned

SEE ALSO:

cgrdmem, cgwrmem, cgrdreg

NAME:

cgrdkey - read keyboard characters

SYNOPSIS:

```
cgrdkey(k,n)
int n;
char *k;
```

DESCRIPTION:

This routine reads a specified number of characters from the Conographic-12 alphanumeric keyboard.

n - number of characters to be read

k - character array of at least size n where characters can be returned

SEE ALSO:

cgkbd, cgrdreq

NAME:

cgrdmem - read memory

SYNOPSIS:

```
cgrdmem(b,n,w)
int b,n,*w;
```

DESCRIPTION:

This routine loads the font base register with the supplied value and reads the specified number of words from memory.

b - value to be loaded into the font base register

n - number of words to be read; this is limited to the range zero to 4096

w - array of at least size n where the words from memory can be returned

SEE ALSO:

cgwrmem, cgrdfbs, cardrea

NAME:

cgrdoff - read offset

SYNOPSIS:

```
cgrdoff(x,y)
int *x,*y;
```

DESCRIPTION:

This routine reads the current Conographic-12 offset.

x - address of an integer variable where the x coordinate can be returned

y - address of an integer variable where the y coordinate can be returned

SEE ALSO:

cgrdreg

NAME:

cgrdpfk - read program function keys

SYNOPSIS:

```
cgrdpfk(k,n)
int n;
char *k;
```

DESCRIPTION:

This routine reads a specified number of keycodes from the program funtion keyboard.

n - number of keycodes to be read

k - character array of at least size n where keycodes can be returned

SEE ALSO:

capfk, cgrdreq

NAME:

cgrdreg - read all the registers

SYNOPSIS:

```
cgrdreg()
```

DESCRIPTION:

This routine reads all the Conographic-12 registers into the array cgreg. They are arranged in the following order:

```
 0:    x position
 1:    y position
 2:    x offset
 3:    y offset
 4:    object scale
 5:    picture scale
 6:    graphic scale
 7:    font base register
 8:    display status word
 9:    accumulator
```

SEE ALSO:

cgrdcsr, cgrdoff, cgrdscl, cgrdfbs, cgrddsw, cgrdacc

NAME:

cgrdscl - read scales

SYNOPSIS:

```
cgrdscl(p,o,g)
int *p,*o,*g;
```

DESCRIPTION:

This routine reads the scale registers of the Conographic-12.

p - address of an integer variable where the value from the picture scale register can be returned

o - address of an integer variable where the value from the object scale register can be returned

g - address of an integer variable where the value from the graphic scale register can be returned

SEE ALSO:

cgrdreg

NAME:

cgstart - open files and initialize

SYNOPSIS:

```
cgstart(fn)
char fn[];
```

DESCRIPTION:

This routine must be called prior to any other. If the parameter fn is present, an output file is created of the display commands that follow; otherwise the files associated with the Conographic-12 are opened. Initialization performed by this routine is identical to that done by cghome.

fn - filename of file to be created vice writing to the Conographic-12

SEE ALSO:

cghome, cgfini

NAME:

cgtplt - emit character string in template mode

SYNOPSIS:

```
cgtplt(a)
char a[];
```

DESCRIPTION:

This routine places the terminal in template mode and then sends a character string to the terminal. It returns the beam position to a point one line down from the previous position and returns the terminal to the previous graphic mode.

a - null-terminated character string to be sent to Conographic-12

SEE ALSO:

cgalph

NAME:

cgvcord - define virtual coordinates

SYNOPSIS:

```
cgvcord(lx,ly,rx,ry,d)
int d;
float lx,ly,rx,ry;
```

DESCRIPTION:

This routine defines the virtual coordinates by specifing the lower-left point of the screen and the upper-right point of the screen. Virtual coordinates are still affected by scaling and offset. Definition of the virtual coordinates with respect to the screen assumes a unity scaling and zero offset.

lx - minimum x virtual coordinate on the screen

ly - minimum y virtual coordinate on the screen

rx - maximum x virtual coordinate on the screen

ry - maximum y virtual coordinate on the screen

d - distortion flag
0: do not distort
1: permit distortion

NAME:

cgvcr22 - curve given endpoints and slopes

SYNOPSIS:

```
cgvcr22(x,y,sb,se)
float x,y,sb,se;
```

DESCRIPTION:

This routine draws a curve when given two points on the desired curve and the slopes associated with these two points. The beginning point of the curve is the present position of the beam. The coordinates x,y define the endpoint. The slope of the beginning point is defined by sb. The slope of the endpoint is given by se.

x - x virtual coordinate of the endpoint

y - y virtual coordinate of the endpoint

sb - slope of the curve at the beginning point

se - slope of the curve at the endpoint

SEE ALSO:

cgcr22, cgcr3b, cgcr3e, cgcr3i, cgcr3m, cgvcr3e, cgvcr3i, cgvcr3m

NAME:

cgvcr3b - curve given three points and beginning slope

SYNOPSIS:

```
cgvcr3b(g,h,x,y,sb,se)
float g,h,x,y,sb,*se;
```

DESCRIPTION:

This routine draws a curve when given three points and the slope at the beginning point. The beginning point is defined by the present position of the beam. The coordinates of the intermediate point are given by g,h. The endpoint is shown by the coordinates x,y. The slope at the beginning point is defined by sb. The address where the value of the ending slope will be returned is shown by se.

g - x virtual coordinate of intermediate point

h - y virtual coordinate of intermediate point

x - x virtual coordinate of the endpoint

y - y virtual coordinate of the endpoint

sb - slope of the curve at the beginning point

se - address of floating point variable where ending slope can be returned

SEE ALSO:

cgcr22, cgcr3b, cgcr3e, cgcr3i, cgcr3m, cgvcr22, cgvcr3e, cgvcr3i, cgvcr3m

NAME:

cgvcr3e - curve given three points and ending slope

SYNOPSIS:

```
cgvcr3e(g,h,x,y,se)
float g,h,x,y,se;
```

DESCRIPTION:

This routine draws a curve when given the coordinates of three points and the slope at the endpoint. The beginning point is defined by the present position of the beam. The intermediate point is shown by the coordinates g,h. The endpoint is defined by the coordinates x,y. The slope at the endpoint is given by se.

g - x virtual coordinate of intermediate point

h - y virtual coordinate of intermediate point

x - x virtual coordinate of the endpoint

y - y virtual coordinate of the endpoint

se - slope of the curve at the endpoint

SEE ALSO:

cgcr22, cgcr3b, cgcr3e, cgcr3i, cgcr3m, cgvcr22, cgvcr3b, cgvcr3i, cgvcr3m

NAME:

cgvcr3i - curve given three points and intermediate slope

SYNOPSIS:

```
cgvcr3i(g,h,x,y,si,se)
float g,h,x,y,si,*se;
```

DESCRIPTION:

This routine draws a curve when given three points and the slope at the intermediate point. The beginning point is defined by the present position of the beam. The intermediate point is given by the coordinates g,h. The endpoint is defined by the coordinates x,y. The slope at the intermediate point is given by si. The address where the value of the ending slope will be returned is shown by se.

g - x virtual coordinate of intermediate point

h - y virtual coordinate of intermediate point

x - x virtual coordinate of the endpoint

y - y virtual coordinate of the endpoint

si - slope of the curve at the intermediate point

se - address of floating point variable where ending slope can be returned

SEE ALSO:

cgcr22, cgcr3b, cgcr3e, cgcr3i, cgcr3m, cgvcr22, cgvcr3e, cgvcr3m

NAME:

cgvcr3m - curve given three points, one at the relative maximum

SYNOPSIS:

```
cgvcr3m(g,h,x,y,se)
float g,h,x,y,*se;
```

DESCRIPTION:

This routine draws a curve when given three points where the slope at the intermediate point is equal to the slope of the chord joining the endpoints. The beginning point is defined by the present position of the beam. The intermediate point is shown by the coordinates g,h. The endpoint is defined by the coordinates x,y. The address where the value of the ending slope will be returned is shown by se.

g - x virtual coordinate of intermediate point

h - y virtual coordinate of intermediate point

x - x virtual coordinate of the endpoint

y - y virtual coordinate of the endpoint

se - address of floating point variable where ending slope can be returned

SEE ALSO:

cgcr22, cgcr3b, cgcr3e, cgcr3i, cgcr3m, cgvcr22, cgvcr3e, cgvcr3i

NAME:

cgvdrwa - set orientation in virtual coordinates

SYNOPSIS:

```
cgvdrwa(s,a,r,x,y,f,ac,av)
int (*f)(),ac,av,r;
float s,a,x,y;
```

DESCRIPTION:

This routine resets the orientation then calls the user's function with the two arguements. Scaling is by means of the Conographic-12 hardware registers therefore limited to the range of 1/64 to 63 63/64. Upon return the previous orientation is restored. Thus the user is provided with a convenient means of recursively stacking the scale and offset settings.

s - scale to be applied

a - angle of desired rotation expressed in radians

r - reflection desired
- 0: no reflection
- 1: reflect about x-axis
- 2: reflect about y-axis
- 3: reflect about both axes, (pi rotation)
- 4: reflect about pi/4 line
- 5: pi/2 rotation counterclockwise
- 6: pi/2 rotation clockwise
- 7: reflect about 3/4 pi line

x - x virtual coordinate which will correspond with present beam position and be the center of reflection and rotation

y - y virtual coordinate which will correspond with present beam position and be the center of reflection and rotation

f - user defined function

ac - first arguement for user function

av - second arguement for user function

SEE ALSO:

cgvdrwr, cgdrawa, cadrawr

NAME:

cgvdrwr - set orientation relative to present orientation in virtual coordinates

SYNOPSIS:

```
cgvdrwr(s,a,r,x,y,f,ac,av)
int (*f)(),ac,av,r;"float s,a,x,y;
```

DESCRIPTION:

This routine sets a new orientation relative to the present orientation then calls the user's function. Thus the reflections, rotations and scaling are cummulative. For best results it is recommended that most scaling factors be 1, if the hardware scaling provided here is insufficient it is recommended that scaling be effected by utilizing another call to cgvcord (note: there is not automatic stacking of cgvcord settings). Upon return the present orientation is restored.

s - scale to be applied

a - angle of desired rotation expressed in radians

r - reflection desired
- 0: no reflection
- 1: reflect about x-axis
- 2: reflect about y-axis
- 3: reflect about both axes, (pi rotation)
- 4: reflect about pi/4 line
- 5: pi/2 rotation counterclockwise
- 6: pi/2 rotation clockwise
- 7: reflect about 3/4 pi line

x - x virtual coordinate which will correspond with present beam position and be the center of reflection and rotation

y - y virtual coordinate which will correspond with present beam position and be the center of reflection and rotation

f - user defined function

ac - first arguement for user function

av - second arguement for user function

SEE ALSO:

cgvdrwa, cgdrawa, cgdrawr

NAME:

cgvelpq - draw ellipse quarters in virtual coordinates

SYNOPSIS:

```
cgvelpq(a,b,xe,ye,q,se)
int q;
float a,b,xe,ye,*se;
```

DESCRIPTION:

This routine draws the specified number of quarters of an ellipse defined by its center and one endpoint of each axis. The ending slope is returned in se. Present beam position is taken as the endpoint of one axis and drawing starts in the direction of the other axis endpoint.

a - x virtual coordinate of center

b - y virtual coordinate of center

xe - x virtual coordinate of other axis endpoint

ye - y virtual coordinate of other axis endpoint

q - number of additional quarters
- 0: one quarter only
- 1: semi-ellipse
- 2: two additional quarters, total three quarters
- 3: full ellipse

se - address of floating point variable where ending slope can be returned

SEE ALSO:

cgelpsa, cgelpse, cgelpsq

NAME:

cgvline - line in virtual coordinates

SYNOPSIS:

```
cgvline(x,y)
float x,y;
```

DESCRIPTION:

In this routine the beginning point is defined as the present position of the beam. The endpoint is defined by the coordinates x,y.

x - x virtual coordinate of the endpoint

y - y virtual coordinate of the endpoint

SEE ALSO:

cgline

NAME:

cgvmove - move in virtual coordinates

SYNOPSIS:

```
cgvmove(x,y)
float x,y;
```

DESCRIPTION:

This routine moves the beam to the indicated position without drawing.

x - x virtual coordinate of the endpoint

y - y virtual coordinate of the endpoint

SEE ALSO:

cgmove

NAME:

cgwrmem - write to memory

SYNOPSIS:

```
cgwrmem(b,n,w)
int b,n,*w;
```

DESCRIPTION:

This routine writes the specified number of words into the Conographic-12 font memory starting at the supplied base address.

b - value to be loaded into the font base register

n - number of words to be written

w - array of at least size n where the words for memory are

SEE ALSO:

cgrdmem

LIST OF REFERENCES

1. Beckermeyer, Robert L., "Interactive Graphic Consoles - Environment and Software," from AFIPS Conference Proceedings, Vol. 37, 1970 FJCC.

2. Boullier, P., and others, "METAVISU: A General Purpose Graphics System," from Graphic Languages, Nake, F., and Rosenfeld A., eds., North-Holland Publishing Co., 1972.

3. Coffman, J.R., System Reference Manual for Conographic Products, Hughes Aircraft Co., 1974.

4. Herzog, Bertram, and others, "Are We Anywhere Near a Universal Graphic Language?" from Graphic Languages, Nake, F., and Rosenfeld, A., eds., North-Holland Publishing Co., 1972.

5. LaHood, J. and Wagner, F.V., "Computer Graphics - Software Design," in Computer Graphics, Gruenberger, F., ed., Thompson Book Co., 1967.

6. Meads, Jon A., "A Terminal Control System," from Graphic Languages, Nake, F., and Rosenfeld, A., eds., North-Holland Publishing Co., 1972.

7. Ritchie, Dennis, "C" Reference Manual, Bell Telephone Laboratories, New Jersey.

8. Thayer School of Engineering, Basic Multi-Use Graphics, 1976.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4.	Professor George A. Rahe, Code 52Ra Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5.	LTJG Gary M. Raetz, Code 52Rr Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
6.	LT States Lee Nelson 3808 Black Canyon Road Fort Worth, Texas 76109	1