To the Graduate Council:

I am submitting herewith a thesis written by Mary Watts Jones entitled "Design of a User Interactive Digitizing Program to Meet the Needs of Casual, Naive Users." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Kenneth R. Kimble, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Vice Provost
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Masters degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Mary Watts Jones

Date 10-21-85

DESIGN OF A USER INTERACTIVE DIGITIZING PROGRAM TO MEET THE NEEDS OF CASUAL, NAIVE USERS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Mary Watts Jones

December 1985

ACKNOWLEDGMENTS

The author wishes to express gratitude to her major professor, Dr. Kenneth R. Kimble, Associate Professor of Mathematics, The University of Tennessee Space Institute, for the suggestion of the topic and for his valuable guidance. Sincere thanks are also extended to the committee members, Dr. Ronald Belz and Mr. Wilbur Armstrong, for their helpful comments.

The author would also like to extend appreciation to her husband and parents for their constant encouragement.

ABSTRACT

A user interactive digitizing program to meet the needs of the naive, casual user was developed. The program provides the user the basic services needed to manipulate graphics primitives using the software package GRAPAC II by the Japan Radio Corporation (uses Core Standard Graphics). It is intended to be used by both naive and expert users but is directed toward the naive users. Human factors, aids for novice users and various techniques for selection, positioning, and text entry are introduced. A summary of the techniques and aids used in the program is given. Recommendations for further expansion to the user interface plus an evaluation of the interface are provided.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The need for research in the area of human-computer interfaces is illustrated by the fact that an increasing number of people who are not computer professionals are using computers to aid them in their jobs. As computer graphics applications are used more and more in engineering, architectural and scientific applications, the need for human factors guidelines governing the person-computer interface becomes more and more important. As several reviews of the human factors literature in computer graphics have indicated, human factors research lags far behind advances in computer graphics technology and applications [1].[1]

Thus, the purpose of this thesis is to design a user interface digitizing program using logical input devices to meet the needs of the naive, casual user. The program is a general purpose human-computer interface program based primarily on the concept of menu selection with a database of menus and responses to these selections. It provides the user with the basic services needed to manipulate graphics objects or primitives. These primitives include drawing lines, rectangles, circles, arcs, fans, polygons, writing text from the keyboard, selection of colors, selection of line type, and the filling of any polygon, circle or fan.

---

[1] Numbers in brackets refer to similarly numbered references in the Bibliography.

The study is not intended as an exhaustive presentation of human factors guidelines dealing with the general topic of man-computer interaction, but is intended only as an example of guidelines developed for a specific area. Thus the human factors guidelines presented here as well as those lacking in the design refer mainly to the components of the system being used.

CHAPTER II

GRAPHIC SYSTEM AND SOFTWARE PACKAGE

This thesis describes an application of existing human factors guidelines to a specific graphic system and software package, GRAPAC II [2]. GRAPAC II is a software package for the NWX-225/235 raster scan intelligent graphic display terminal produced by the Japan Radio Company (JRC). The software conforms to the CORE graphic system announced by SIGGRAPH-ACM in 1979. GRAPAC II supports synchronous and asynchronous input CORE specifications and provides powerful interactive functions for CAD and other applications. Programs are coded with ANSI FORTRAN IV so they are not dependent on the host computer. Parts that must depend on the host computer are provided as independent subroutines.

GRAPAC II is a set of basic graphic processing subroutines. These subroutines are divided into six functional groups: 1) primitive output function, 2) picture and segment function, 3) attributes function, 4) viewing function, 5) input function, and 6) control function. A brief description of each function is given in the following paragraphs. The GRAPAC II functions are also summarized in Table 1.

The subroutines of the primitive output function are the most basic graphic elements: output lines, polygons, circles and characters.

The picture and segment function subroutines assign a number to a primitive logical set as a segment. Thereafter, graphics can be handled on a segment basis. A segment is used by graphic display

Table 1. GRAPAC II Functions

Function	Subtopic	Routines
Primitive output		Move, line, character, marker, symbol, polygon, circle, arc, fan, ellipse, shading
Picture segment function	Retained segment	Delete, number, modification, copy
	Temporary segment	Delete, segment picture modification
Attribute function	Primitive attributes	Color, type of line, character precision, font, character size, character up direction, character spacing, character output position, type of marker, edge style, picked
	Segment attributes	Visibility, blink, detectability, image transformation
	Picture attributes	Visiblity, detectability
Image function	GRAPAC II imaging	Window, viewport, world coordinate conversion
	Terminal imaging	Terminal window, terminal viewport, hardware zoom
Input	Pick input	cpick, awpick, setpic
	Keyboard input	gkeybd, awkeyb, setkb, echkb
	Button input	await, awbutn, awbldc, setbtn, setalb, echsg
	Stroke input	GSTROK, AWSTRK, SETSTR, SETLCP
	Locate input	RDLOC, GLOCAT, AWBLOC, ECHSG, ECHLC, SETLCP
Control	Initialization & Stopping	INITG, terms
	Error handling	SETERR, IQUERR
	Color	SELTBL, SCOLOR, SCOLA
	Screen Modes	IMMODE
	Message	MSGPUT, MSGSZ, MSGCOL, MEGPOS, MSGCLR, MSGERS

on/off, deletion, move and input echo specification and other functions. A picture is considered to be a number set of segments. Figure 1 illustrates the pictures and segment data structures.

GRAPAC II handles three kinds of attributes: primitive, segment, and picture. Primitive attributes specify the color of lines and characters, the type of lines, character precision, character size, etc. Segment attributes consist of a visibility attribute, detectability attribute, image transformation, etc. The two picture attributes consist of visibility and detectability.

The viewing function specifies which part of the plane is to be displayed on the screen. The plane is a two-dimensional virtual plane with infinite real axes (world coordinate system).

The input function accepts inputs from five devices: pick, locator, button, keyboard and stroke.

Finally, there are control functions which control GRAPAC II initialization, error processing and the GRAPAC II functions themselves.

The hardware (Figure 2) consists of the display which is an intelligent CRT graphic display in which the NWX-200 series high resolution rastor scan architecture has been reinforced by inclusion of Random Scan Architecture Refresh Memory, Picture Data Buffer Memory System and a Picture Control Processor. A Dedicated Graphic Processor is used as the internal processor.

A large capacity segment buffer is used to store display data sent from the host computer (VAX 11/785). The adaptation of a dedicated high speed graphic processor provides interactive functions and permits

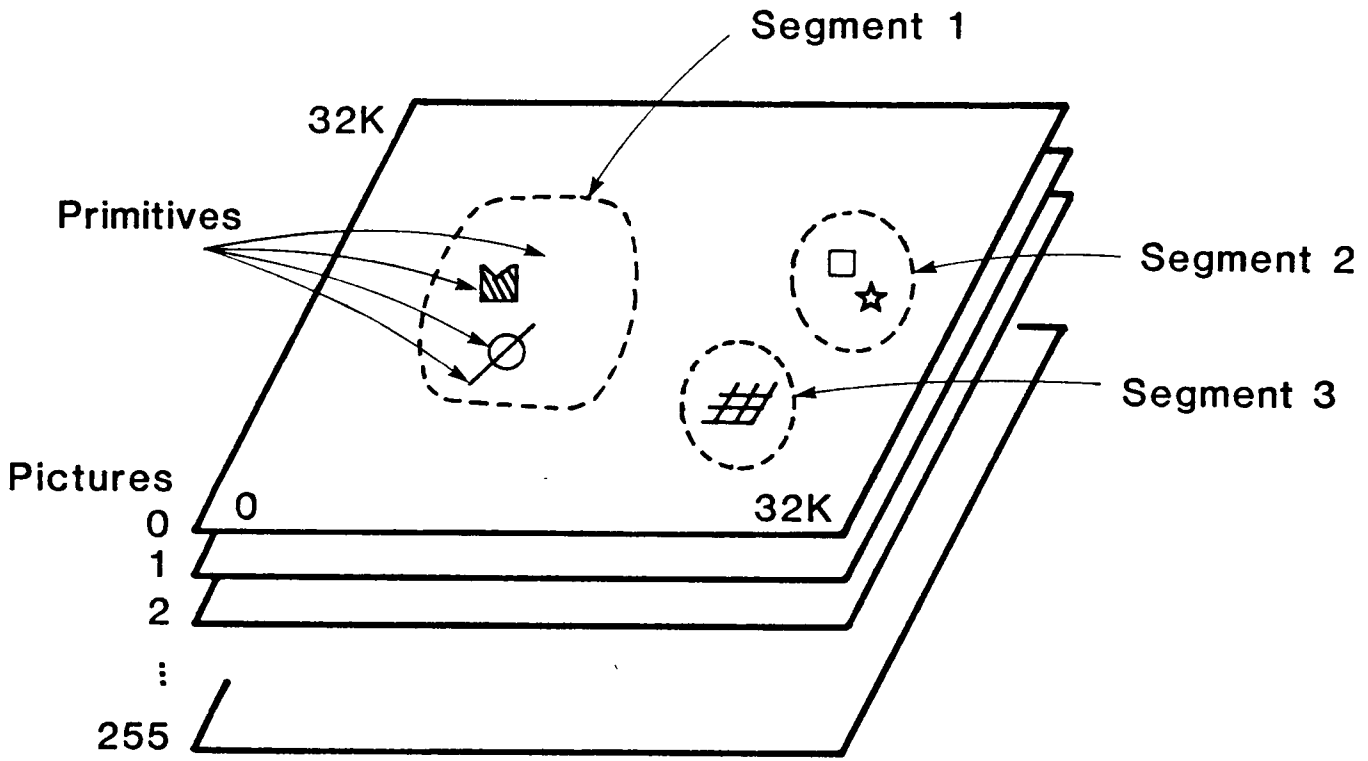


Figure 1. Pictures-Segment Data Structures

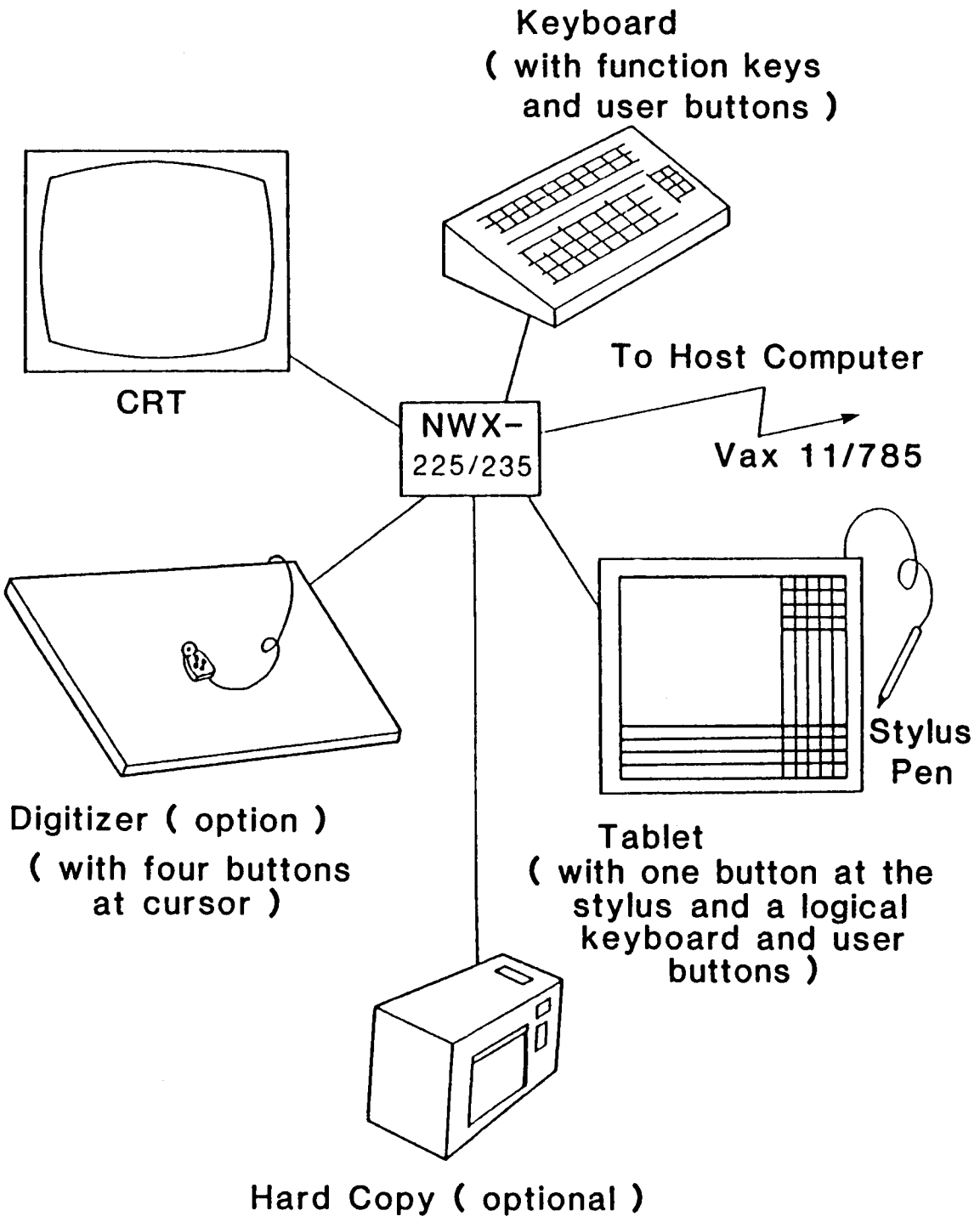


Figure 2. NWX-225/235 Hardware Configuration

use as a specialized CAD/CAM system terminal. The Graphic Processor Unit is a dedicated 32 bit microprogram-controlled microprocessor capable of speed coordinate transformation (zooming/scaling, scrolling and rotation) and high speed display processing (window-viewport transformation and clipping) [3].

CHAPTER III

LITERATURE REVIEW

Human Factors

The design of interactive graphic applications whose aim is a good interface between man and machine involves numerous factors. One of the major factors is human-machine conversation. James D. Foley and Victor Wallace [4] stress that the language in computer graphics communication is not one of spoken or even written words, but rather one of pictures, and of actions such as button pushes, lightpen indications and joystick movements which serve as words. A natural grammar offers few awkward constraints to express ideas using the fundamental devices and symbols which are available. This also permits the application and user interface to be useful with minimum user learning and training.

Another factor that needs to be considered is the psychological principles involved. The interface should avoid psychological blocks that often prevent full user involvement in an interaction. A few of these blocks are boredom, panic, frustration, confusion and discomfort.

Boredom is a consequence of improper pacing. Adequate response time is the critical concern.

Panic is the consequence of unexpected long delays, such as a conversational partner giving no reply to a statement or question over a prolonged period of time. The remedy, when the real delay cannot be reduced, is to introduce a response providing assurance and information.

Frustration results from an inability to easily convey intentions to the application and consequently an unexpected response occurs or nothing at all. It is further brought about if the unexpected response cannot be undone or if the user cannot determine what response actually took place. Frustration is caused by inflexible and unforgiving operations. One way to avoid frustration is to force the user to think twice and reconfirm his request whenever an action from which recovery would be difficult is to be executed. Deleting a picture is an example of an action which should require confirmation. A forgiving application should let the user cancel in midstream so he can proceed with what he really wants to do.

Confusion is the consequence of being overwhelmed by detail. The user may be offered many poorly defined options at once. The remedy is to increase the amount of underlying structure, to improve the perception of the structure already present or to reduce unnecessary detail. The command structure can be used to prompt the user, telling him what he can do now. The perception of structure among objects on the screen can be enhanced by using different line types, widths, intensity levels, or geometric shapes.

Discomfort comes from providing an inappropriate physical environment for the graphics workstation. This psychological block is all too often neglected when designing a system.

There are two distinct languages in the communication between the human and the graphic machine. One is the language of display by which the machine presents information regarding the state of its data and the options available for further action by the user. The second is

the language of actions using logical input devices, by which man relates his intended actions of machine-stored data with reference to objects in the displayed picture. The goals of language design, both in action and picture, should be directed toward a language format which is natural, and which does not add to the boredom, panic, frustration, and confusion of the user [5].

Studies have further demonstrated that user attitudes can dramatically affect learning and performance. Anxiety, that may be generated by the fear of failure, may reduce short-term memory capacity and inhibit performance. If users are insecure about their ability to use the program, afraid of destroying files or the computer itself or are overwhelmed by volumes of details, their anxiety will lower their performance.

Various Techniques

Eason [6] defined the naive computer user as a person who is not a computer expert, but who does use the computer to help in performing certain tasks. Naive users are, by definition, limited in their general fund of knowledge in the area of computer technology. One of the most important elements in the design of interactive user-computer interfaces is the selection of the devices and techniques by which a user performs elementary tasks. The common objective of all human factors work simply stated, is to achieve, through appropriate design, functional effectiveness of whatever physical equipment or facilities people use.

A multitude of interaction techniques, each having a specific purpose, such as specifying a command or selecting a displayed object, are

implemented with some device, such as a tablet, joystick, keyboard, lightpen, or trackball. Several techniques for selection (Figure 3), positioning (Figure 4), and text entry (Figure 5) will be discussed [6,7].

A selection technique (Figure 3) involves picking an item from a list of alternatives. Typical interaction techniques are 1) menu selection with a lightpen, 2) menu selection with a cursor controlled by a tablet or mouse, 3) type-in of name, abbreviation or number on an alphanumeric keyboard, 4) programmed function keyboard, and 5) voice input.

Menu selection is used most often for command entry. The following parameters of the menu itself should be considered.

Organization: single level vs hierarchical must be considered. If the set of alternatives is small enough to be contained in the screen space, a single-level menu can be used. Otherwise, a hierarchical menu or a single-level menu requiring several sequentially displayed screens are possibilities. In these last two possibilities aids will be needed to move through the selections. In a recent experiment, Snowberry [8] found that selection time accuracy improved when broader menus with fewer levels of selection were used.

The item order of command entries in a menu have to be taken into context. There are several reasonable ways of sequencing command entries in a menu: 1) alphabetical, 2) frequency of use, the most frequent appearing at the top of the menu, and 3) logical, placing entries of the same category together (normally used in hierarchy).

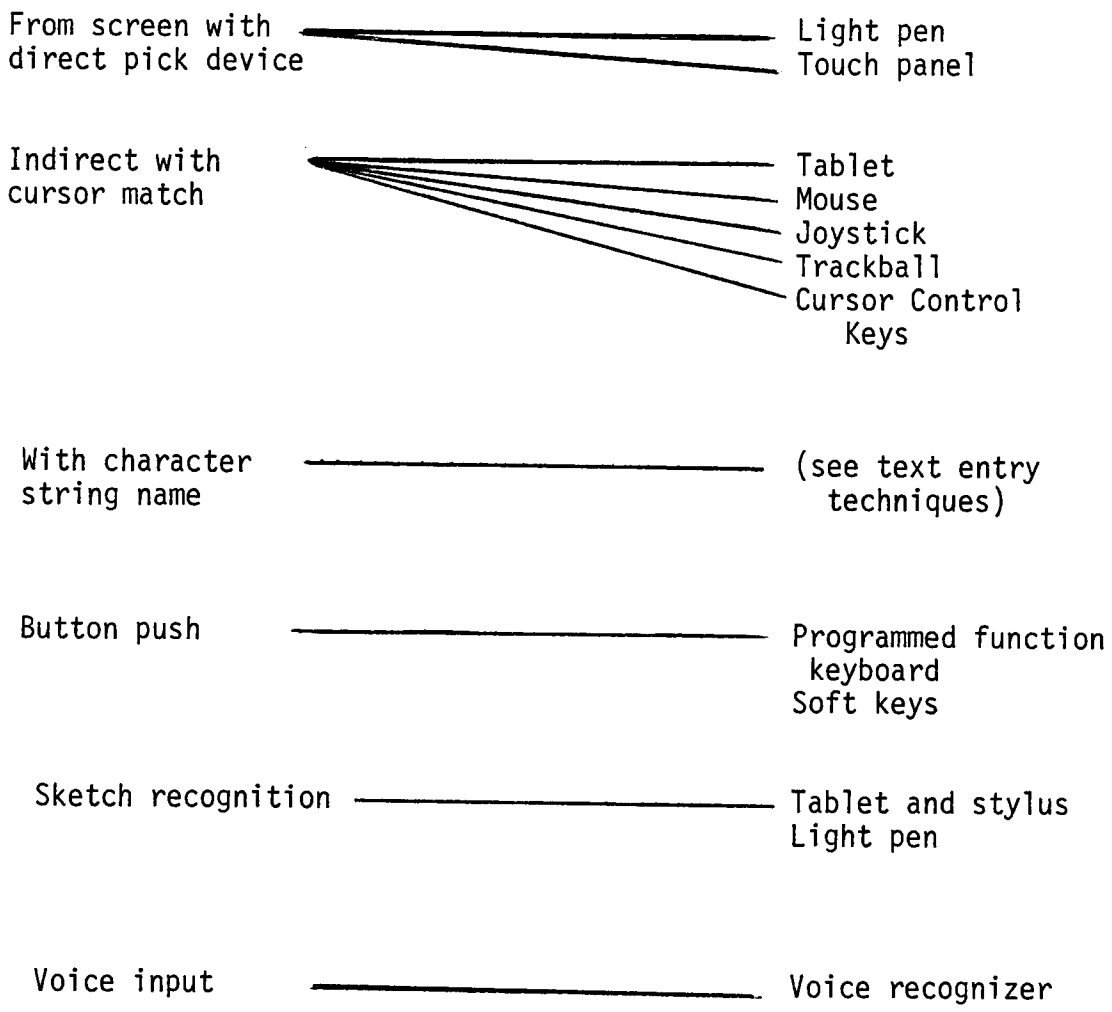


Figure 3. Selection Techniques

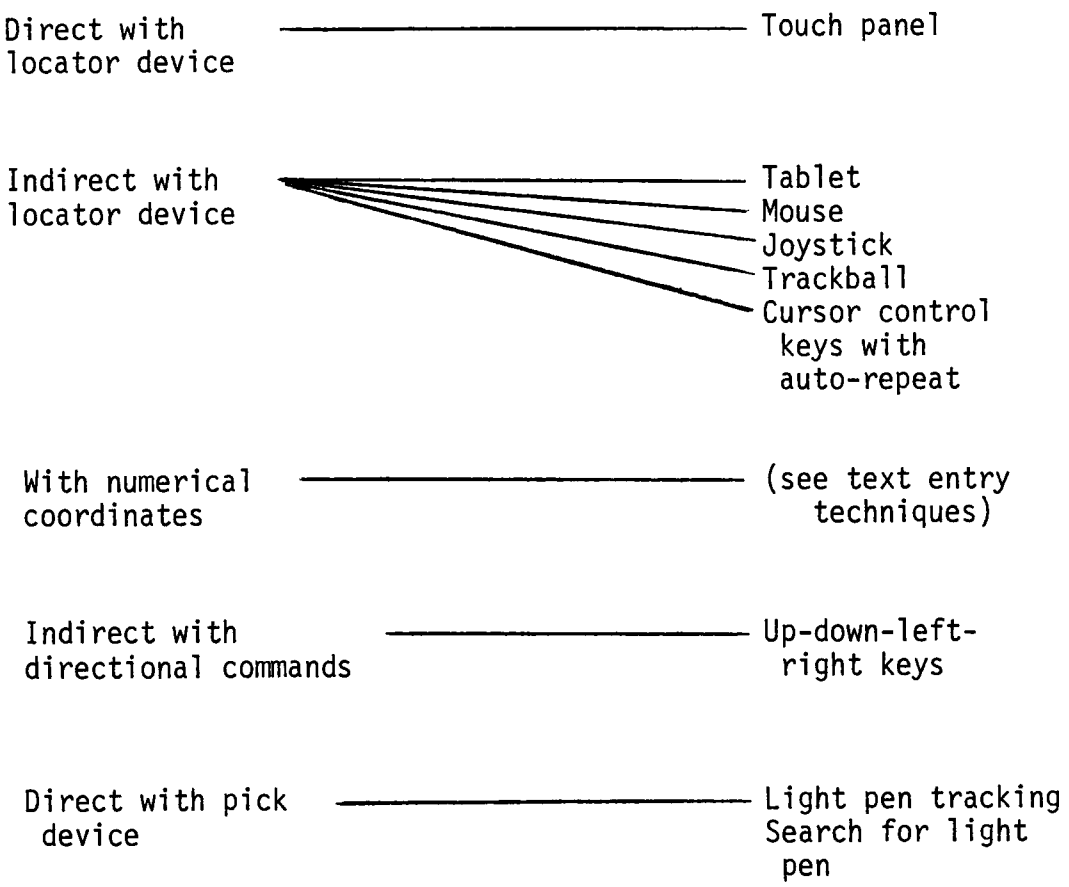


Figure 4. Positioning Techniques

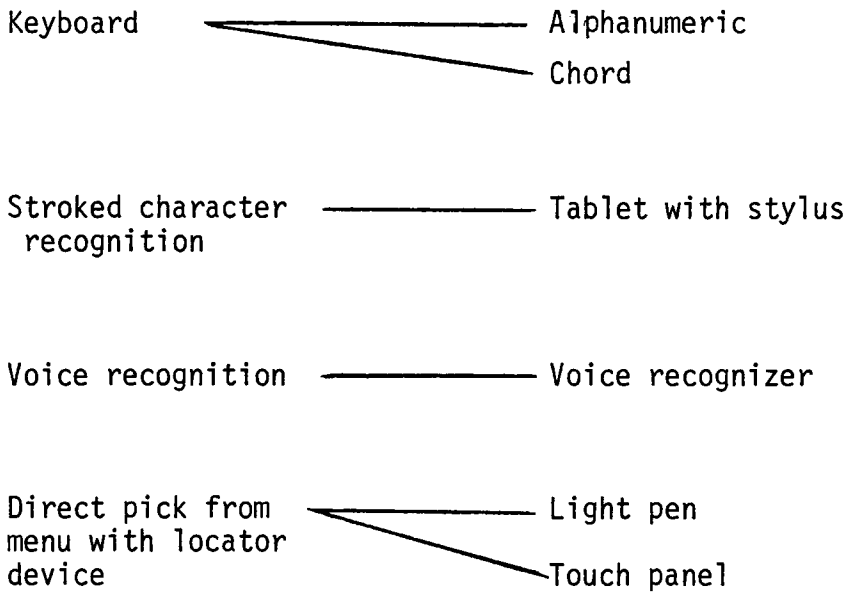


Figure 5. Text-Entry Techniques

Representation: iconic vs textual, is another consideration. Menus are often in text form, but they can also be in graphic form (graphical symbols known as icons). Iconic menus can be designed to occupy less screen space than text menus. The menus for the Lisa, Macintosh, and Star computers contain many examples of icons used to represent objects and attribute values. One disadvantage of icons is that if not very careful they can lose their meaning.

Positioning a menu on the display screen is still another parameter that has to be considered. Static menus and pop-up menus can be used. A menu that is always in the same position on the display screen is called static. A static menu can be 1) part of the same screen as the main display, 2) on an auxiliary screen next to the main display screen, 3) on the same screen as the main display, but replacing the main display when the user wants to make a selection, and 4) printed on a tablet. A menu printed on a tablet means that the user has to look away from the display screen, hence destroying visual continuity.

A pop-up menu, which appears when a selection is to be made, is feasible, if a positioning device can be considered for implementation of the selection. It requires only minimum hand movement when a user makes a selection.

Visibility, always visible vs sometimes visible, is a consideration. Pop-up menus and "pull-down" menus are in the sometimes visible category, while static menus can be either.

Organization such as horizontal, vertical or blocked menu items should be considered. Studies by Coffee [9] show no difference in search times between horizontal and vertical lists.

One of the last considerations should be display targets, whether alphanumeric or iconic. They should be large enough in order to reduce positioning time and error rate.

To make menu selections, many interaction techniques and variations are possible. This discussion limits itself to some commonly used selection techniques. Character string name type-in is often used. A menu is displayed and the desired item is typed in on an alphanumeric keyboard. Label type-in is where a label associated with a menu item is input. Direct pick by lightpen is used to pick the desired item directly from the displayed menu. Touch panel pick allows the user to touch a finger to the desired menu choice on the display screen. Simulated pick with a cursor match is used by positioning a cursor over the desired menu choice and depressing a button. A unique function key associated with the command is depressed. Voice recognition is where the user speaks the name of the selected menu and a word recognizer determines which of a set of known words was spoken. And finally the user makes sequences of movements with a continuous positioning device such as a tablet, mouse or a lightpen.

The positioning (Figure 4) task involves specifying a position in application coordinates. A few of the parameters that are involved in the interactive positioning techniques are listed below.

The coordinate systems that a user of an interactive graphics system is typically aware of are: the application coordinate system in which the computer maintains coordinates; the screen coordinate system, in which the user views an image; and the positioning device coordinate system, in which the user moves a tablet, joystick or other devices.

Cursor form and visual aids for positioning involve movement of a displayed screen cursor to a desired location. Studies have shown that users have problems in first finding the screen cursor when targets and background items are alike. The type of display in view has to be considered before the cursor type is chosen [5].

Feedback is a positioning parameter. Forms of feedback will be discussed later.

A short description of some specific positioning techniques are given: 1) a locator is a device whose position is used to indicate a location on the screen and movement is directly mapped into the movement of a cursor or a displayed object, 2) the location of a graphic object is controlled by using step keys to move the object up, down, left and right for 2-d applications, plus in or out for 3-d, 3) direct translation with a lightpen tracking is performed with a small cross which is pointed at with a lightpen, 4) a lightpen is pointed at the screen and a raster scan of the screen is used to find the pen's position, 5) to indicate a position on the display screen, the user types in the coordinates of the location via on an alphanumeric keyboard.

Text entry (Figure 5) involves expressing information in strings or blocks of characters selected from a predefined character set. An important distinction should be noted between text entry and selection. In text entry each character individually causes no action, but collectively in a strong the characters act as a single entity.

A few text entry techniques will be given. Text entry by voice recognition is where a keyboard can be implemented by means of a voice

recognition device. Each letter is spoken and separated by 0.1 to 0.2 seconds of silence. Commonly used words can also be recognized to speed up the input process. Text entry by stroke character recognition is accomplished when the user prints the text with a continuous positioning device, usually a tablet and stylus. The computer breaks the text into strokes from which letters can be recognized. Text entry by menu selection is a series of letters, syllables, or other basic units displayed as a menu. The user then inputs text by choosing letters from the menu with a selection device.

Fundamentally each of the techniques described represents a task for choosing. Selection chooses from among a set of entities and positioning chooses a place in space. Text entry chooses a sequence from among a special set of entities, the characters.

Aids for Novice Users

S. K. Tagg [11] pointed out that it is very difficult to create an interactive program that is able to meet adequately the needs of users with varying degrees of experience. He also asserted that interactive programs should be designed to help meet more adequately the needs of the many novice users, and he proposed that this be accomplished in part by creating more adaptable data structure and documentation.

In discussing the application of human factors in the area of software design for the novice, Stewart [12] asserted that at times the software interface should serve as a barrier. The most effective man-computer interaction occurs when the user is protected by the

interface from parts that are too complex for him or her to understand.

Another idea to facilitate the man-computer interface was suggested by Miller and Thomas [13]. Basically, they proposed that it would be desirable to allow the user to erase or withdraw a prior command by executing a special undo command. Such a command would benefit the novice, as the ability to withdraw a command should provide relief to novices who often experience distress from worrying about doing something wrong.

Help facilities for the notice are another area of concern. According to Maguire [14], a help facility could be designed to aid the novice in situations where he is unable to recall what is to happen next. A main concern in providing help is that of giving the user the type of information that he or she needs. The return from help must leave the program at exactly the same state as it was in when help was invoked.

Feedback is an integral part of the design. The user must know a number of things: 1) what the previous state of the program was, 2) what action was executed, 3) what happened, and 4) what alternatives are now possible. In other words, the user should be aware at all time where he is, what he has done, and whether it was successful.

Echoing is a very simple type of feedback for the user. Another form of feedback shows the program's response to a user's request. For example, if the user requests that an item be deleted from the screen, it is appropriate that the image of that item be removed from the screen. Feedback can profitably take the form of a partial result

slowly appearing on the screen. Other forms of feedback involve highlighting a menu item so the user can know actions have been accepted.

Positioning of feedback is important. There is a natural tendency to designate a fixed area of the screen for feedback messages.

Error messages should be brief, without negative tones and should be constructive. Ringing of bells and bold messages should be avoided. The suggestion has been made that human performance improves if errors are issued immediately and that the disruption of the user thought process by immediate interruption is not a serious impediment. A powerful advantage of immediate interruption is that changes can be made simply by instructing the user what to do at that point to correct the error, either by redoing the request completely or allowing a backup to the point before the error.

Other aids briefly mentioned here are: messages to the user should be displayed in upper and lower case rather than all upper case; display messages should have bright characters (yellow, white) on a dark (green, grey) background; information contained in a message should be factual, necessary, complete and readily usable; critical information should be presented at the beginning of a message; screen separation should be made apparent to the user through lines or some other demarcation; commands, messages, etc. should appear in the same area of the screen at all times; to minimize the eye-hand coordination, the tablet could be slanted to enable the user to see his hand at the lower portion of his visual field; the perception of displayed items can be enhanced by various visual aids such as color and line style.

CHAPTER IV

TECHNIQUE AND IMPLEMENTATION

General Interface

The software developed is a small example of how a user can interact with the machine to perform a specific task. The user interface interprets the user interactions directly. The interface is the link between the user and the software and the machine. (Figure 6)

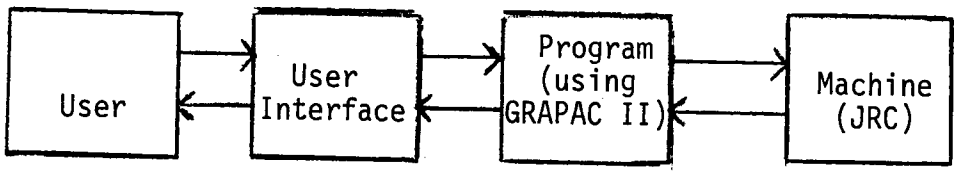


Figure 6. Interaction Between the User, Interface, Software and Machine

The user interface is based on the screen layout in Figure 7. Features of the interface include menu management, prompting, echoing, confirming, error handling, and cancelling a sequence of interactive inputs to return to the previous state.

The NWX-200 series has one of two graphics display modes, full graphics mode and separate mode. The separate mode was chosen for use in this program. An eight line area for normal size message characters is provided at the bottom of the screen as a message area separate from the graphic area.

The program generates the picture within a single window on the screen which is considered the user working area. This working area

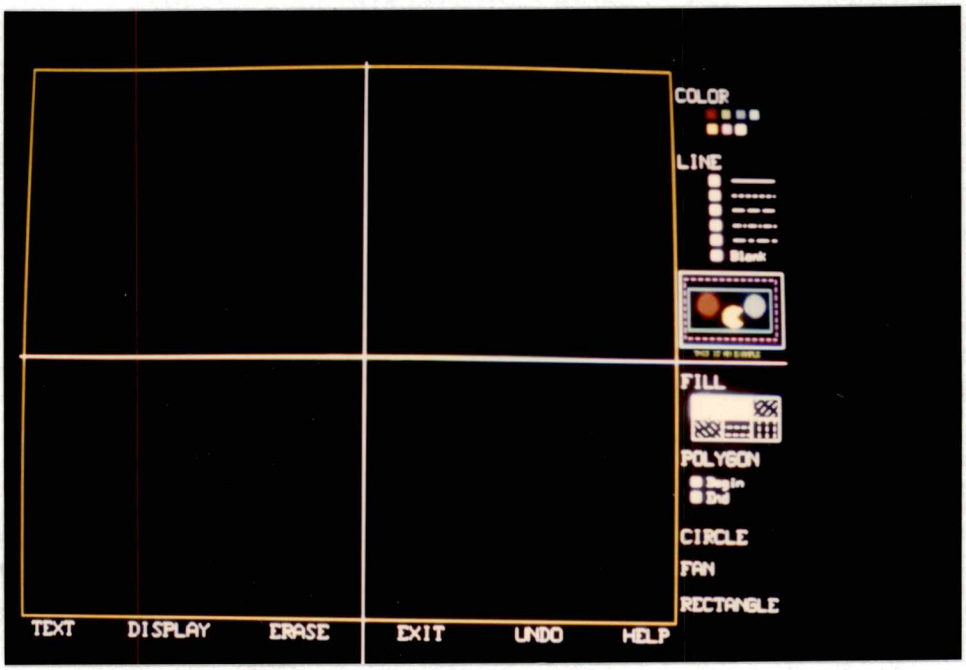


Figure 7. Screen Layout

has been outlined for user visibility. Clipping will occur whenever a primitive's bounds exceed the window. A clipping problem occurred here which will be discussed later.

The program is completely menu driven. There are no key words, no command sequences or command structures to memorize. The menus guide the user by telling him only those operations that he needs to perform at each step in the procedure.

On the right side of the user window is the main menu. This menu permits the user to create and display a picture. At the bottom of the user window is another menu which aids the user in the construction of a picture.

The menu items are organized by frequency of user. COLOR and LINE style changes will occur very frequently. The organizations of POLYGON, CIRCLE (ARC), FAN and RECTANGLE could have been in any order. The bottom row menu consists of the following commands in order from left to right: 1) TEXT; 2) DISPLAY; 3) ERASE; 4) EXIT; 5) UNDO; 6) HELP. The command, TEXT, is on the far left because reason dictates that it should be used less often and HELP is on the far right side because reason dictates it should be used most often. Details on the underlying structure of these menu items will be given later.

Each item of the menus is referenced by an X and Y coordinate (the hit area). To select a menu item the user has only to place the cursor over the desired item and press the button. Then the commands needed to complete the operation will be given to the user in the message area. Other forms of storing menu items could be used. For example, each menu item could be put into a segment. Then pick mode

could be used to select the menu item. The item selected could be blinked or highlighted. The only constraint against using this in the program is that for the user to select a menu item, he would somehow have to tell the interface that he was ready to select a menu item. The program would have to go from locate mode to pick mode. This would cause a two or three step process, whereas by using X and Y coordinates, he can stop what he is doing and place the cursor over the item he wants. This is only a one step process. He can also pick as many items as he wishes without having to enter and exit a menu.

It is notable that the user menus are referenced by file name. These can be changed without having to recompile the program.

The clipping problem that was mentioned previously, occurred when calling the routines from the software package to the window, viewport, and the separate screen mode. Originally the bottom row menu was at the top of the user working window. When creating a primitive whose outer bounds exceeded the window, clipping occurred. Then if the picture visibility was turned off and then back on, clipping occurred everywhere except at the top of the user window. The primitives' bounds extended up into the top menu. If the separate mode routine was taken out, all clipping worked fine. The only conclusion that could be made at this point without contacting the company, was that the combination of using the separate mode and defining a window other than the default (0.0, 1.0, 0.0, 1.0) caused the clipping problem.

All input except text entry defaults to the digitizing pad. The input is done through the one button on the cursor. Data for circles, arcs and fans (radius, beginning angle, ending angle and center point)

can be entered through the keyboard if the user selects to do so.

Pick input is used in filling a polygon, circle or fan, in deleting a particular primitive and selecting items from the text entry sub-menu to determine the character size and positioning of the text on the screen. Pick input is input by direct selection of a segment displayed on the screen.

The limitation in the use of the input devices is based upon the idea of simplicity. The fewer devices the user has to go between, the less likely he is to get confused, lost and encounter errors he does not understand.

All X and Y coordinates recorded are in Normalized Device Coordinates (NDC). Both X and Y are on a 0.0 - 1.0 square plane. In this particular program the viewport and window are mapped one-to-one. A further extension to this interface would be to let the user define his own window. This would involve routines to scale the data to the plane.

The recording of the data about the user's interactions is essential to the interaction techniques used in the program. With a menu-driven application based primarily on the cursor and tablet, the data consist of the recording of each user input, recording the X and Y tablet coordinates of the cursor and recording the information about each primitive and its attributes at the time of input. The final external data file created from this information should be able to be used as the source of other inputs. This will be elaborated in another chapter.

Segmentation plays an important role in the creation of the user interface. Both menus are stored in different segments. This allows for further expansion of adding new menus to the program. Each primitive (line, polygon, circle, arc, fan, and text), at the time of creation, is stored in a segment. This allows easy manipulation of the primitives such as deleting a particular primitive, filling a primitive or canceling the creation of one.

GRAPAC II allows for the opening of segments 0 to 9999. In this program segments 0 to 299 are used for menu set up and control. The starting segment for the creation of the primitives is 300.

The message area also plays an important role in the interaction between the user and the program. It is a visual aid to the user. His picture is never interrupted by a message. He always knows which area of the screen to look at for any information he might need, such as instructions or prompting, confirmation of a particular happening, a user error that has occurred and how to correct this error, and help information. The cancel and confirm options that will be mentioned in the description of a few of the menu items will appear in the message area. These options are put in messages but still can be referenced by X and Y coordinates. They have their own hit area and will always appear on the far right side of the message area.

Each type of message (instruction, confirmation, error and help) has its own color. If a user uses the program often, he will soon distinguish the different types of messages. Yellow is used for instructions and prompting messages, blue for confirmation messages, cyan (light blue) for help messages and green for error messages.

Another visual aid is the placing of small markers at most of the data points on the screen. The markers are temporary and the user is able to see exactly the digitized point. On a polygon the first point is marked. On a line, a marker is placed at the beginning of a line sequence. When a blank line is selected a new marker will again appear at the beginning of the first line in the sequence. On a circle, arc or fan, the center point, beginning and ending points are marked.

The introduction to the program is short, simple and to the point. The user is given a brief explanation of how the menus are set up, what the initial defaults are, how to get help, and what the program is actually there to do. See Figure 8.

The introduction is also referenced by file name. If the program is modified, any explanation changes to the introduction can be made without having to recompile the program.

The program initially defaults to the creating of lines (two points are all that are needed). The endpoint of one line is treated as the beginning of the next. Unless a blank line is chosen from the main menu, this will always occur. The creation of polygons, circles, arcs, fans, and text are treated as separate parts. Their data points are not connected in anyway. Once one of these is created and the user just arbitrarily selects a point in the user working area, a line will be drawn from the last endpoint of the last line to the new point.

One safety check that occurs is if the user chooses a point outside the user working area and that point is not in one of the hit areas for a menu item. This check is to see if the point was intended

This program will allow you to digitize a picture by creating lines, polygons, circles, arcs, and fans. Text entry for labeling and etc. is also available. An external data file will be created for you after the digitizing session is over to save the picture. This file is a text file and can be used to reproduce the picture on the JRC or to be used to produce the picture on another machine.

Unless indicated by menu all data points will be considered as lines and will be connected together. A BLANK line will allow you to start new.

A working space has been defined for you (to work in).

All input including DIGITIZING and MENU SELECTIONS will default to the digitizing pad unless otherwise indicated. You do have the option of input by keyboard for circle, arc and fan data. Also text entry is through the keyboard.

All X and Y coordinate values are between and including 0 and 1.

The default color is YELLOW.

The default line style is a SOLID line.

You have the option from the edit menu to delete any line polygon, circle, arc, fan or text created. (UNDO)

Any item from the main menu is picked by clicking once over the menu item.

If in the process of creating a polygon, circle, arc or fan you decide to cancel the operation, you have the option to do so.

If you need help at any time choose the HELP option and instructions will be given to choose the item for which you need help.

YOU MAY BEGIN DIGITIZING BY HITTING RETURN.

Figure 8. Introduction Menu to Program

for a menu item but missed, an error on the user's part or if the user actually wants the point recorded. The user is to confirm or cancel the recording of the point.

Another safety check that occurs is when a user is creating a polygon and he forgets to end it, then indicates he wants to create another primitive. A message occurs telling him to end the polygon first before he selects the other menu item.

Menu Management

The right hand side menu consists of COLOR, LINE style, an example, FILL types, POLYGON, CIRCLE, FAN and RECTANGLE (see Figure 7). This menu has the options that allows the user to create and display a picture on the screen. A detailed description of how the user interacts with each is given in the following paragraphs.

The color item allows the user to select a new color. The default color is yellow. Colors are selected by moving the cursor over the desired color box and clicking the button. A message appears in the message area to confirm the color change. The reason for using these seven colors is that the program is set to ECHO mode. Normally, when using GRAPAC II sixteen colors can be displayed but in this mode only eight can be displayed (black is excluded). Aaron Marcus has suggested that the use of color in computer graphics has often emphasized too many colors [10]. The limited use of colors here in this application may not be a hindrance after all.

For a user to change the color in creating a particular primitive he must change the color first before starting the primitive. This is

true for all primitives except a polygon. The color can be changed in midstream in creating this primitive.

The line menu item allows the user to select the type of line to be output to the screen. If a line, polygon, circle, fan or rectangle is created, the line type is applied to the edge. The types of lines available from GRAPAC II are solid lines, dash lines, long dash lines, dot-dash lines, and long dot-dash lines. A blank line has also been added to the line types. The line type is changed the same way the color is changed. The default line type is a solid line.

The picture between the line styles and the fill option is a small example to let the user see what this application offers. The picture consists of a red and a green circle, each filled with different fill types, a blue fan filled with the other four fill types, a cyan rectangular polygon, a magenta rectangular polygon with dashed lines, a white rectangular polygon and a text string explaining this picture is an example picture. The picture shows a few of the primitives, colors and line styles that are available to the user. If the user selects this area a message will be displayed informing him this is an example picture. For further development to this menu, the space could be used to add a new menu item.

The fill option permits the user to fill any polygon, circle or fan created. There are six fill types given (GRAPAC II will allow up to thirty). The six fill types chosen were chosen arbitrarily. Further expansion to the menu could be to allow more fill types.

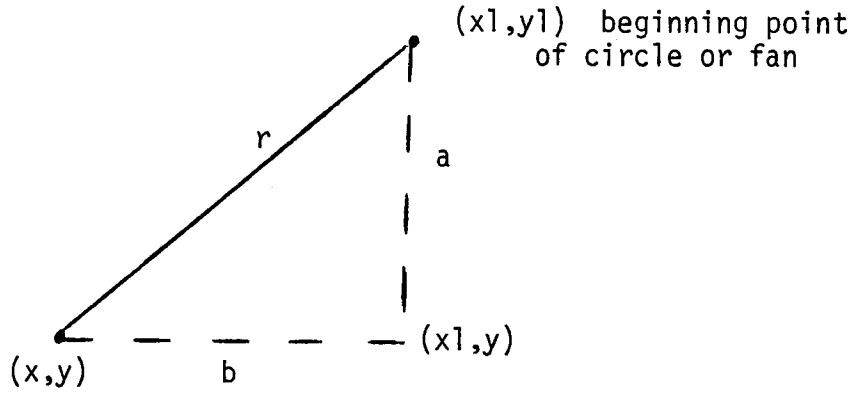
The user does not have to fill the primitive at the time it is first created. To fill a primitive the user first needs to select the

fill type. Then instructions on how to choose the primitive will be given in the message area. Once the fill option has been selected, the mode of the input device is changed from locate mode to pick mode. Pick mode allows a particular segment to be detected. The user is instructed to pick the outer edge of the primitive he wants to fill. If the user has not changed the color before selecting the fill option, the current color will be the fill color. Error checks are incorporated just in case the user happens to choose a segment that does not have a primitive in it that can be filled. For example, the user may choose a line that he has created. If the user happens to pick such a primitive that cannot be filled, an error message is displayed. He is instructed to try again if he chooses to do so. Upon leaving this routine, the program returns to locate mode.

The polygon option allows the user to create a polygon (GRAPAC II permits up to 256 vertices and as few as three vertices). The user has to first indicate the beginning of the polygon by selecting the BEGIN option of the polygon item. Upon completion of the polygon, he must then indicate the ending of the polygon by selecting the END option. He must then confirm or cancel the polygon. If the user gives less than three vertices, the option is automatically cancelled and he is informed of this cancellation and why it occurred. If the user inputs up to 240 vertices, he is warned by a long ringing of the bell that he must end the polygon soon. If the vertices reach the upper limit, then the bell rings again. The polygon will complete itself at this point. A message explaining what has happened is displayed to the user. He is then to confirm or cancel this polygon.

Ringing of bells is advised against, but in this case it seemed the appropriate thing to do. More than likely the user will be looking down at the tablet and not the screen. The long ringing bell will give him an indication that something has occurred and he will more than likely look up at the screen to see what is going on. If the user decides to cancel the creating of a polygon, he is informed that to do so he should end the polygon and then select the cancel option when it appears on the screen.

The circle item is just what it says. It allows the user to create a circle or an arc. If the beginning angle equals the ending angle (or the beginning point equals the ending point) then a complete circle is created. Once this item is selected, the user is given the option of keyboard entry for the circle data (radius, beginning angle, ending angle and the X and Y center point coordinates). The default is the digitizing pad. A help option is also given at this point. To create a circle or arc by the pad, the user is instructed (in the message area) to first indicate the center point of the circle, then indicate the beginning point of the circle and then the ending point using the cursor. Within the routine itself, the radius is calculated from the center point to the X and Y coordinate of the beginning of the circle ($C^2 = A^2 + B^2$) (see Figure 9). The angles are calculated by using common trigonometry functions (see Figure 10). After the radius and angles are calculated, all information pertaining to the circle is displayed in the message area (center point (X and Y coordinate), radius, beginning angle, and ending angle). An error check on

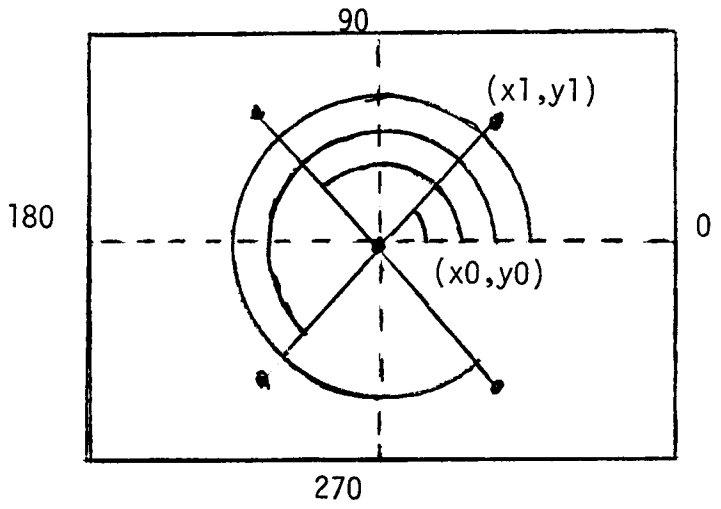


$$r^2 = a^2 + b^2$$

$$r^2 = (y1-y)^2 + (x1-x)^2$$

$$r = \sqrt{(y1-y)^2 + (x1-x)^2}$$

Figure 9. Calculation of Radius



(ATAND - finds arc tangent of angle in degrees)

Tests made to find angle based on the positioning of x1,y1.

1. If y1 > x0 and y1 > y0, then
   angle = ATAND ((y1-y0)/(x1-x0))

2. If x1 < x0 and y1 > y0, then
   angle = 180.0 - ATAND ((y1-y0)/(x0-x1))

3. If x1 < x0 and y1 < y0, then
   angle = 180.0 + ATAND ((y0-y1)/(x0-x1))

4. If x1 > x0 and y1 < y0, then
   angle = 360.0 - ATAND ((y0-y1)/(x1-x0))

5. If x0 = x1 then

   If y1 > y0, angle = 90.0

   Else angle = 280.0

6. If y0 = y1, then if x1 > x0, then angle = 0.0

   Else angle = 180.0

Figure 10. Calculation of Angles

the radius equaling zero occurs. If the radius does equal zero, the user is informed of the error and that to correct the error he will have to begin by selecting the circle menu item again.

For keyboard entry, the user is instructed to type in the radius ($r > 0$), beginning angle (in degrees), and then ending angle (in degrees). The program enables the keyboard at this point. Return is the signal for the end of each input. He then has the option to enter the center point of the circle through the keyboard ($0 <= X <= 1$, $0 <= Y <= 1$) or he may use the digitizing pad to indicate the center point. Error checks are incorporated to see that the radius does not equal zero and that the data are valid. Again if an error is detected the user is told so and will have to begin by selecting the circle menu item again. No matter which way he chooses to create this primitive, he has the option to cancel the process at any time.

The fan option is created in the same way the circle and arc are created.

The rectangle option allows the user to create a somewhat true rectangular polygon. GRAPAC II has several echo types for the locator device. One type is the square. The square echo places a square with the echo position and the locator cursor position as the two opposite vertices (see Figure 11). After selecting this item, the user is instructed to give two opposite corner points of the rectangle. Once the two corner points are selected the routine can then determine the other two corner points and draw the rectangle.

The bottom row menu is made up of HELP, UNDO, EXIT, ERASE, DISPLAY and TEXT. This menu has the options that aid in the manipulation of

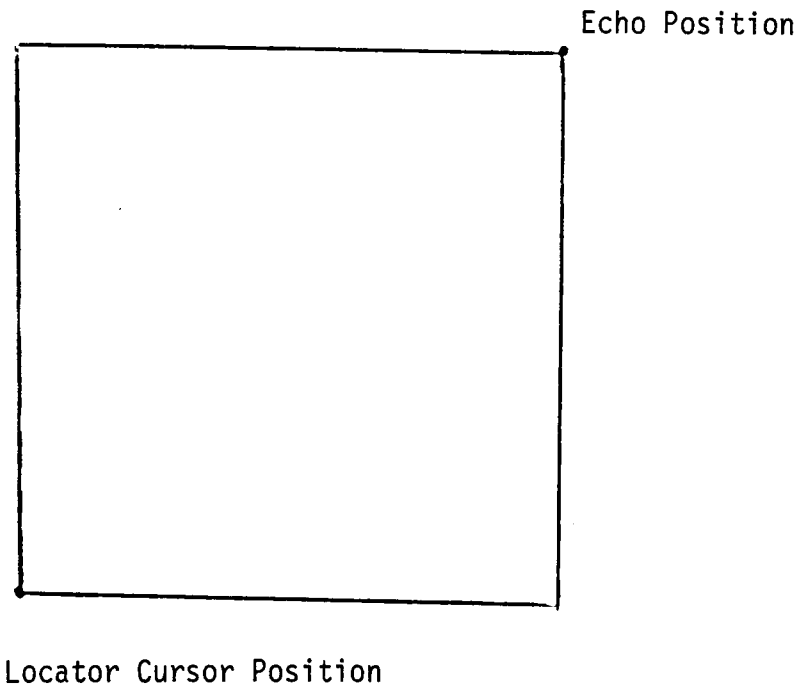


Figure 11. Rectangle

the primitives created in a particular picture. A detailed description and how the user interacts is also given of each in the following paragraphs.

The help option is an option that the user, at almost any time during a session with the program, can invoke. The help option is set up so all help information will appear in the message area. Once the user has indicated he needs help, a message will appear that says he is now to select the menu item he needs help on (whether it be in creating a polygon, changing the color, putting text on the screen, etc.). The user then goes to that particular "hit area" by placing the cursor over the item he needs help on and clicking the button to record the request. If the user selects an area that is not a "hit area", a message will tell him there is no information on that particular item. After help is completed the user will be at the point where he left off before selecting help.

The undo option lets the user delete any primitive he has created (line, polygon, circle, arc, fan, or text). Once the undo option is selected the input is changed from locator mode to pick mode. The user is instructed to pick the outer edge of the primitive he wants to delete using the cursor. This does not necessarily mean the last primitive created. Since the primitives are all in separate segments, deletion of any one primitive can occur anytime during the digitizing session. The primitive selected will blink to indicate to the user which one he has selected. The routine then returns to locator mode and instructs the user to confirm or cancel this deletion. Once confirmed, the primitive will disappear from the screen.

The exit option is the termination of a digitizing session. Once the graphics mode is exited the user is asked if a picture file is desired to save what he has just created. If yes, the user is to type in the name of the data file. At this point an external graphics picture file is created (called a metafile). Further discussion will be given later on how the data are stored internally as well as the creation of the external file. If the user does not want to save the picture, he is to confirm this decision. A second confirmation is required. Once this second NO is given all is lost. Last the user is asked if he wishes to return to digitizing, either to work on the same picture (if he saved it) or to create a new one. If not, the execution of the program is terminated.

The erase option clears the screen of everything created up to that point in the user working area. The user is to confirm or cancel the erasing process and is informed that all his work so far will be lost if he confirms. All counters are reset to zero and the segment number is reset to the initial segment (300) that is to be opened first.

The display option when selected displays the last X and Y coordinate that was recorded (either the last vertice recorded in a polygon, the center of the circle just created, the beginning or ending point of a line, etc.). This menu option was added later in the development of the menu. At first, every X and Y coordinate that was recorded was displayed in the upper left-hand corner of the message area. The displaying of each coordinate caused the whole digitizing process to be slowed down. For each X and Y recorded, a conversion from real to character had to take place plus it had to be displayed. The input of

the points could get way ahead of the displaying of the points and the displaying of the primitives on the screen. To cure this, the display option was incorporated. The user must indicate he wants to see the data point before it will be displayed.

The text option allows the user to input text strings through the keyboard to be displayed in the user working area. Once this is selected, the user has the choice to select a sub-menu which allows him to choose a particular character size and a particular positioning (vertical versus horizontal). If he chooses to go into this sub-menu, he still may choose the default values on either or both the character size and positioning. The default character sizes on the world coordinates are character height 0.01 and character width 0.03. The default positioning is horizontal (0.0, 1.0). Within the sub-menu, the user has the choice of six character sizes and six positionings. When the sub-menu is entered the input mode is changed from located mode to pick mode. The user is then instructed to choose the character size and positioning he wants. He must choose both before he can exit this sub-menu. Once the menu is exited, the input is changed to keyboard mode. If the user does not select the sub-menu, the default values are assumed. After the character size and the positioning are in order, either through defaults or the sub-menu, the input mode is changed to keyboard mode. The user is instructed to input the characters through the keyboard. Return is the signal for the end of the input. A maximum of fifty characters are allowed. Truncation of the character string will occur at that point. He then is instructed to place the cursor where he wishes the beginning of the string to be

in the user working area. The input mode is then changed back to locator mode. The user then has to confirm or cancel this operation. The current color is the text color. Further expansion to the text input option could be to allow the user the choice of different font styles.

Phases of Interaction

The following gives a small glimpse of an interactive event in this user interface. The phases consist of the following: 1) the input the user provides (initially from the digitizing pad), 2) a prompt which indicates to the user that the input has been accepted and gives instructions on what to do next, 3) an action takes place according to the input, and 4) flow control occurs and the program evaluates the input and records all information.

In addition to the above phases, more phases are included to accommodate the user's needs. These consist of 1) input checks for validity of input, whether they be error checks or safety checks, 2) two menus which can be invoked by the user at almost any time, and 3) a cancel feature which enables the user to skip the present input requested and thus the user will not be locked into an event which might have been invoked inadvertently.

The interaction cycle is now 1) the input the user provides, 2) if menu, prompt the user that the input has been accepted and if needed give instructions on what to do next, 3) action takes place according to the input, 4) check the cancel option: if INPUT = "cancel" then end cycle, 5) apply input checks and if errors then

report errors, give a backup option or end cycle, and 6) flow control occurs and the program evaluates and records all information. Figure 12 shows a flowchart of the interaction cycle.

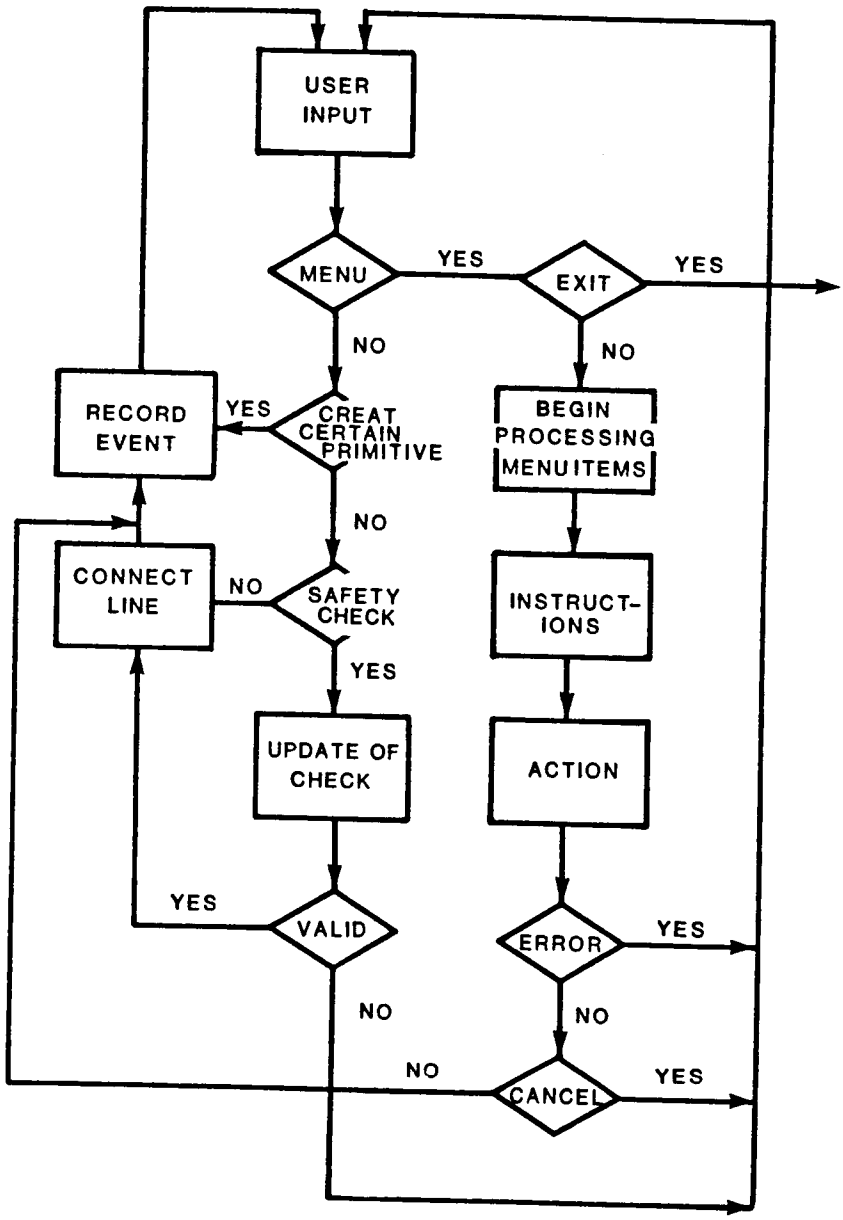


Figure 12. Flowchart of Interaction Cycle

CHAPTER V

DATA STORAGE

Internal Data Storage

The data are stored within the program until the exit option is selected. To be able to produce a graphics data file, all information pertaining to the primitives created in the application must be saved.

As mentioned earlier in the paper, each line, polygon, circle, arc, fan, and text string is stored in an individual segment. Several tables have been set up to store all the graphical information and the segments are the link to each of these tables.

The main table (ISEGARR) contains the basic information concerning the picture. The table holds the segment number, the code (whether the segment has been deleted (0) or not deleted (1)), the type of primitive created (line (1), polygon (2), circle (3), fan (4), text (5)), the number of X and Y coordinates involved in the particular primitive (n), the beginning position in the X and Y coordinate array where the data points involved are stored, the color of the primitive (red (1), green (2), blue (3), cyan (4), yellow (5), magenta (6), white (7)), and the line type of the same primitive (solid (1), dash (2), long dash (3), dot-dash (4), long dot-dash (5)).

Whenever a primitive is deleted, the segment is detected by using the pick mode. Once the segment is detected, the segment is searched for in ISEGARR and the code is flagged with a zero (initially flagged with a one). When the final graphics picture file is created, this primitive is ignored as are all other deleted primitives.

A second table (CIRFAN) contains the information needed in creating a circle or fan. The segment is, of course, the link between the main table and this table. The table holds the radius, the beginning angle and the ending angle of each circle, arc and fan created.

A third table (RCHARA) and a fourth table (CHARARR) contain the information for text string entries. RCHARA contains the segment number (link to main table), the length of the character string, the character height, the character width, vertical positioning, and horizontal positioning. CHARARR contains the text string. The position in the RCHARA coincides with the position in CHARARR.

The fifth table (IFILLAR) holds the fill information. This table contains the segment number that the fill occurred in, the interior color of the fill, the interior style number, and the kind of hatching lines.

Whenever the fill option is selected and the primitive to fill has been chosen, the fill and all attributes are appended to the segment. Once appended, the segment has to be closed again (in first creating a segment, the segment is opened and after all is completed, closed). If a primitive is chosen to be deleted, everything concerning the primitive is deleted.

A final table (STOARR) contains all the X and Y coordinates contained within the picture. All line points, vertices of polygons, center points of circles, arcs and fans, and beginning location of text strings are stored in this table.

External Graphics File

Currently, for the machine being used, the hardware needed to get a hard copy of the picture is not available. For the program to be worth using by a user, the picture created must somehow be saved, whether the user needs to put the picture back upon the JRC to make corrections or additions, or the user wishes to reproduce the picture on another machine.

GKS has designed one way to produce an external graphics file [15]. This external file is termed a graphics metafile or just metafile. GKS has three different types of encoding mechanism to be used by different applications and environments: 1) encoding within the rules of ISO 2022 to facilitate network transfer, 2) binary encoding to facilitate intra-machine reuse or to minimize processing requirements, 3) plain text encoding to facilitate transfer between highly different computer architectures. For a user using GKS, he is free to choose the meta-file encoding most appropriate to his application and environment.

In creating the external graphics file for this program, plain text encoding was chosen. The novice user can look at this file and see what was actually created. If the user wants to reproduce the picture on another machine, a program can be written for that machine to read the data and reproduce the picture.

A routine has been incorporated in this program to reproduce a picture that was created and saved earlier. Upon exiting the program, the user is asked if he wishes to save the picture and then to type in through the keyboard the name of the graphics data file to be created. Initially the program asks the user if he wants to work on an old

picture or to create a new one. If he wants to work on an old picture he types in the name of the file and the file is read and the picture is put back upon the screen. The user can still make changes to this picture, the same as if he had just created it. As the data file is read, all information is restored into the internal data tables (mentioned earlier). After the user is finished with the corrections or additions, he can once again save the whole picture.

The data are stored similarly to the GKS system. All data items have an item header containing a) the item type identification number which indicates the kind of information that is contained in the item (whether a primitive or an attribute) (see Figure 13) and b) a character string identifying visually what the item is, to improve legibility of the file and to provide an error control facility. The file is produced using character strings, integers and real numbers. Real numbers describing coordinates and length units (radius) are stored in Normalized Coordinates.

Figure 14 gives an interpretation of each item to be output to the file. Under each diagram is a description of what the field of each record type contains. The segment number is included along with each primitive mainly for the reproducing of the picture on the JRC. The user will need to understand this and may need to ignore this field when reproducing the picture on another machine. Figure 15 shows an example of the external graphics file.

The primary time that the attributes color and line style are recorded in the output file is when a color change or line style change occurs. The only other recording of color and line style is in the

Output Primitives	
Line	1
Polygon	2
Circle	3
Fan	4
Text	5
Fill	6
Output Attributes	
Color	7
Linestyle	8

Figure 13. Graphics Metafile

LINE

C	"LINE"	S	N	P

C: Code for line (1)
S: Segment number
N: Number of points (2)
P: List of points (X1,Y1,X2,Y2)

POLYGON

C	"POLYGON"	S	N	P

C: Code for polygon (2)
S: Segment number
N: Number of vertices
P: List of vertices $(Xi,yi)_{i=1,N}$

CIRCLE

C	"CIRCLE"	S	R	THS	THE	P

C: Code for circle (3)
S: Segment number
R: Radius (>0)
THS: Beginning angle (in degrees)
THE: Ending angle (in degrees)
P: Center point (x,y)

FAN

C	"FAN"	S	R	THS	THE	P

C: Code for fan (4)
S: Segment number
R: Radius (>0)
THS: Beginning angle (in degrees)
THE: Ending angle (in degrees)
P: Center point

Figure 14. Items for Output Primitives

TEXT

C	"TEXT"	S	P	CH	CW	PV	PH

C: Code for text (5)
S: Segment number
P: Beginning point of text string
CH: Character height
CW: Character width
PV: Vertical positioning
PH: Horizontal positioning

POLYGON FILL

F	"FILL"	C	S	CL	IS	IT	N	P

F: Code for polygon fill (6)
C: Code for polygon (2)
S: Segment number
CL: Fill color
IS: Interior style number
IT: Hatching line style
N: Number of vertices
P: List of vertice points $(xi,yi)_{i=1,N}$

CIRCLE/FAN FILL

F	"FILL"	C	S	CL	IS	IT	R	THS	THE	P

F: Code for circle/fan fill (7)
C: Code for circle or fan (circle = 3, fan = 4)
S: Segment number
CL: Fill color
IS: Interior style number
IT: Hatching line style
R: Radius (>0)
THS: Beginning angle (in degrees)
THE: Ending angle (in degrees)

Figure 14. Continued

COLOR

C	"COLOR"	CL

C: Code for color (8)
CL: Color code

1 - red
2 - green
3 - blue
4 - cyan
5 - yellow
6 - magenta
7 - white

LINE STYLE

C	"LINESTYLE"	IS

C: Code for line (9)
IS: Linestyle code

1 - solid
2 - dash
3 - long dash
4 - dot-dash
5 - long dot-dash

Figure 14. Concluded

```
8 COLOR 5
9 LINESTYLE 1
1 LINE 300 2 0.474197 0.534928 0.412702 0.496872
1 LINE 301 0.412702 0.496872 0.470290 0.459792
8 COLOR 6
2 POLYGON 302  7
0.703482 0.472457
0.690817 0.451003
0.725944 0.439283
0.749352 0.445143
0.740562 0.469527
0.722037 0.494919
0.703482 0.472457
6 FILL 2  302 6 1 2   7
0.703482 0.472457
0.690817 0.451003
0.725944 0.439283
0.749352 0.445143
0.740562 0.469527
0.722037 0.494919
0.703482 0.472457
8 COLOR 1
3 CIRCLE 303 0.93934  32.70  32.70 0.363933 0.703726
7 FILL 3  303 1 4 2 0.093934  32.70  32.70 0.363933 0.703726
8 COLOR 3
4 FAN  304 0.058602  60.00 304.77 0.556139 0.334880
7 FILL 4  304 4 1 1 0.058602  60.00 304.77 0.556139 0.334880
8 COLOR 4
9 LINESTYLE 2
2 POLYGON  305   5
0.586413 0.657857
0.700583 0.657857
0.700583 0.718375
0.586413 0.718375
0.586413 0.657857
8 COLOR 7
9 LINESTYLE 1
5 TEXT  306 0.119968 0.574938  27 0.010 0.030 0.000 1.000
THIS IS AN EXAMPLE OF TEXT.
5 TEXT  307 0.256600 0.457839  21 0.025 0.030 1.000 0.000
END OF TEXT EXAMPLES.
```

Figure 15. Example of External Graphics File

recording of a fill. The color and line style here are treated different than the normal color and line style since the user has the option of filling a primitive any time after the primitive is created.

CHAPTER VI

RESULTS

Conclusions

The basic guidelines that were followed in developing the interactive user interface program were 1) to provide simple, consistent interaction sequences, 2) not to overload the user with too many different options and styles for communicating with the program, 3) to give appropriate feedback to the user, and 4) to allow the user graceful recovery from mistakes.

The user interface is flexible. There is no mystery about the workings of the interface, it should be under total control of the user. The interface does have its drawbacks which will be mentioned later in this chapter.

The interface is easy to learn, novice users can use the program as a tool to help them perform their own jobs. Computer novices should be able to learn the interface and use if efficiently in under half an hour.

The program provides an environment whereby it is safe to explore and learn, providing the user reads and follows all instructions in the learning phase. There should not be any dangerous, irreversible actions onto which a user might stumble.

The database graphics file created is general in the sense that it represents both textual and graphical information. The information is set up so that the user can transfer it to another machine if he so desires.

The menu setup has the advantage in that no special vocabulary needs to be learned by the user; specifications depend on recognition. Other advantages include structuring the choice space thereby not requiring special training and minimizing the user's confusion by having the main menus readily available at all times.

As mentioned in the background information, the human factors guidelines and steps incorporated in this application are directed toward the system being used. However, the guidelines and steps used to help overcome the psychological principles and meet most of the needs of the naive user are summarized in the following paragraphs.

The two single level menus which are small enough to be contained in the screen space are the main interface between the user and the machine. The menu selections are accomplished by the cursor controlled by the tablet. Each menu is set up by frequency of use. The menu items are mainly in text form, static, always in the same place on the main display screen and always visible to the user. One menu is placed vertically along the right side of the screen, the other placed horizontally at the bottom of the user working window. Each menu item's "hit area" should be large enough to avoid errors, however, an error check is included to prevent the user from recording an input that is not correct.

The positioning technique used is the locator device whose position is used to indicate a location on the screen and movement is directly mapped into the movement of the cursor. The coordinate system used is Normalized Device Coordinates. The user is informed in the introduction that all X and Y coordinates are between and including

zero and one. The cursor is the visual aid which shows the movement from an old location to a new desired location on the screen.

The text entry technique involves the use of the keyboard as a selection device. Return is the end of input.

Aids that are incorporated in the interface to help overcome the psychological principles (boredom, panic, frustration, confusion, and discomfort) are: 1) adequate responses to all inputs given; 2) help available at almost anytime; 3) user is to confirm major happenings; 4) user can cancel or undo anything he has done; 5) errors that occur from within the program's reach can be corrected without having to start all over; 6) messages are short and brief but factual and complete; 7) screen separation made apparent to the user; 8) responses of any kind including help always appear in the same area on the screen so as not to disrupt the user's picture.

Recommendations

The program and the human interactions presented here are only a small example of what human-computer interactions are all about. The program has been left open for further development. A few suggestions for additions are given.

The adding of new menus would be fairly simple. The two menus on the screen are set up in two segments. The segments (as do all segments) have a visibility capability which can be turned on and off. To add a new menu, the new developer of the program would have to create the new menu in a file (since menus are stored in separate files) similar to the previous files. The user window would have to be

redefined to the default window temporarily. The new menu file would be read into a segment. The visibility of the menu being replaced would be turned off and the new menu placed similarly to the previous menu. The user window would then be redefined.

Further additions such as zooming, rotating and scrolling a specific segment (primitive) could be incorporated. GRAPAC II allows for this. These routines have not been tested by this developer to see if they work properly.

Other additions and expansions to the user interface were mentioned and are scatter throughout the description of the menu items.

Evaluation

After letting a few naiver computer users and experienced computer users try out and experiment with the program, the following strong and weak points about the user interface were observed. A full scale evaluation of the interface has not been made at this point because of time and the fact that the program has not been put into full use. Only then can an efficient evaluation be made.

A few of the strong points that were observed are listed and elaborated on in the following paragraphs.

The user interface was basically easy for the computer users, whether novices or not, to learn and use. Within minutes they were able to construct a small sample picture on the screen. The novice users were more leery than the experienced users but soon they learned to relax and began exploring the menus and seeing what they offered.

The help option was easy to follow. The fact that the user could get help on a specific item without having to browse through lots of other information first was very appealing.

The capability to fill a primitive at anytime during the digitizing session was approved by most of the users who experimented with this program. A user does not always decide to fill the primitive at the time he creates it. Also if the user decides he does not like the color of the fill or the fill style, he can refill the same primitive again by choosing a color (this has to be done first before the fill style is chosen or the current color will be the fill color), then the fill style and then follow the instructions given.

The cancel and confirm option given on such menu items as erasing the screen or deleting a particular primitive was well liked by most of the users. The user might accidentally hit the erase option's "hit area". No problem occurred since he was able to cancel the option. Also in deleting (UNDO) a specific primitive a problem can occur if two or more primitives are very close together on the screen. As mentioned in a previous chapter, when UNDO is chosen the mode of the input device is changed from locate to pick. The user then selects the outer edge of the primitive he wishes to delete. If two segments are very close together, the pick may detect the wrong segment. Since the primitive whose segment was detected blinks, the user is able to cancel the deletion if it is the wrong primitive. Otherwise he is to confirm the deletion.

The fact that the user could undo anything he had done (except in

creating a polygon which is a weak point and will be discussed later) was a big plus, especially during the learning phase when experimenting with the new environment. The main advantage of the UNDO is that the deletion of any primitive can occur at anytime, not just when it is created which some interfaces require. The idea of putting each primitive in a different segment made this possible.

Also from the observation the following weaknesses were noticed. These are also given and elaborated on in the following paragraphs.

One of the main weaknesses, which in a sense is a weakness on the user's part, is that the user did not always read and follow the instructions and prompts given to him. Especially a more experienced computer user, who was using the program, seemed to want to explore initially instead of reading the instructions. Eventually, he figured out how to correct his mistakes and was able to continue in a constructive fashion.

Another weakness is that the majority of the time the user is looking down at the pad instead of the screen. He may have been so involved in the digitizing process that he did not see important messages that may have appeared in the message area, especially if an error had occurred. He would more than likely miss the message that tells him how to correct the error. The ringing of the bell in the creation of a polygon whose vertice limit is about to be reached, is a way of warning the user to look up at the screen.

Another drawback is the fact that while creating a polygon there is no way to delete or correct a line of the polygon (drawn from one

vertice to another) if a mistake is made. If the UNDO option was selected that part of the polygon created up to that point would be deleted. If a major mistake is made, the user has to end the polygon, then cancel it and begin again. Currently no suggestions or ideas have been made to correct this problem.

One last drawback is the fact that once circle, fan or rectangle is selected the color cannot be changed. The user must learn to change the color before he selects one of these if he wants a different color than the current color. He does have the option to cancel the menu item, select the color and reselect the menu item again.

Areas in which the program can be of use are in remote sensing to digitize contour lines of an area (see Figures 16 and 17) and to draw and produce block diagrams to show flow control such as the example of Figure 12 (flowchart of interaction cycle) shown reproduced in Figure 18. Also cartoons (Figures 19 and 20) and emblems (Figure 21) can be drawn using this application. A few of these examples were shown directly from the screen of the JRC and others were reproduced from the external graphics file on the Hewlett-Packard plotter.

All user interfaces are going to have their strong points and their drawbacks as does this one. No user interface can completely satisfy all novice or experienced users needs but steps toward this goal are currently under development.

The role of the user, after all, is to use the program developed to successfully accomplish some goal or sets of goals. The user's primary concern is, in fact, the accomplishment of that goal or goals and not by which it is achieved.

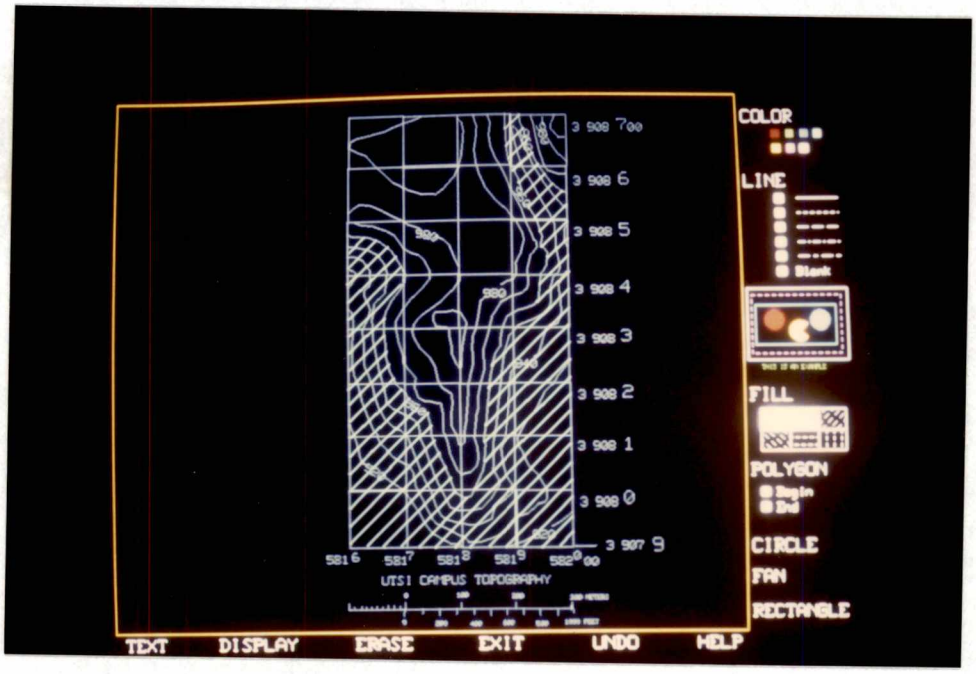


Figure 16. UTSI Campus Topography (JRC)

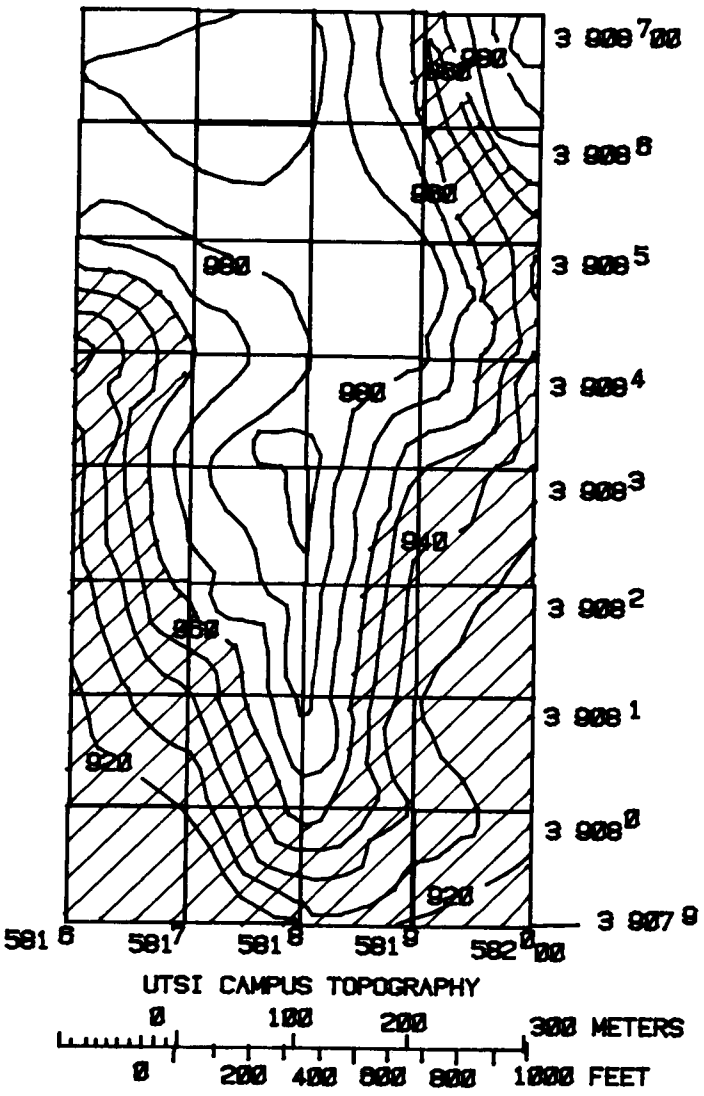


Figure 17. UTSI Campus Topography (Hewlett-Packard Plotter)

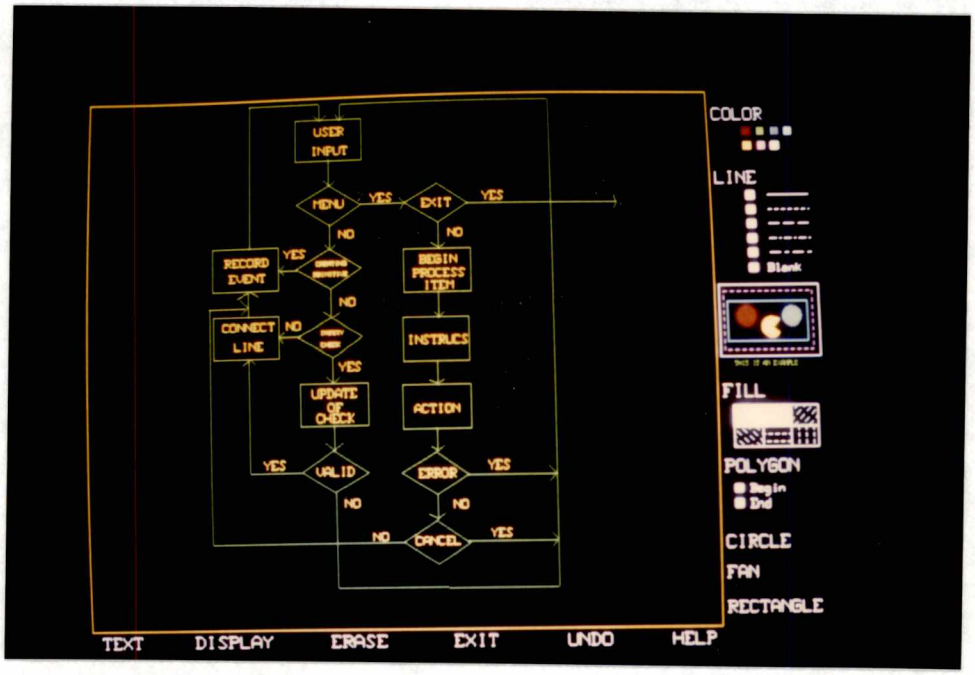


Figure 18. Flowchart of Interaction Cycle (JRC)

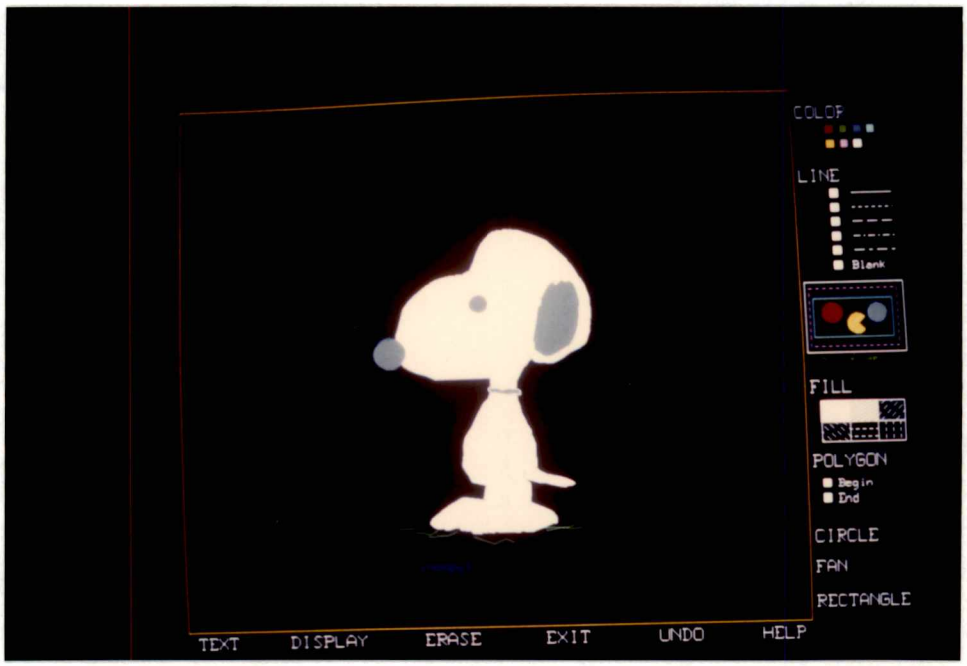


Figure 19. Snoopy (JRC)

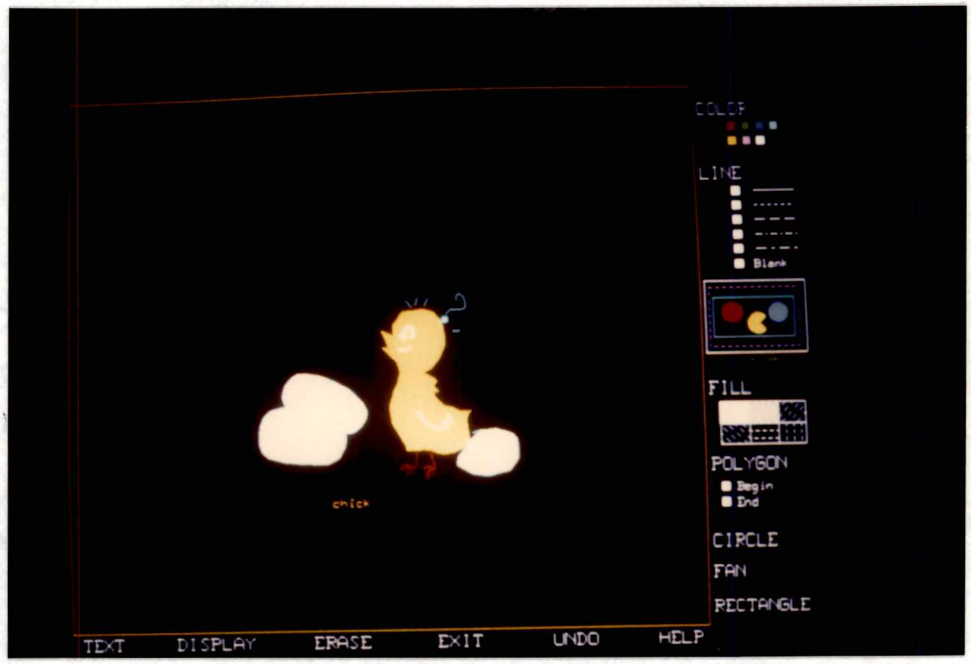


Figure 20. Chick (JRC)

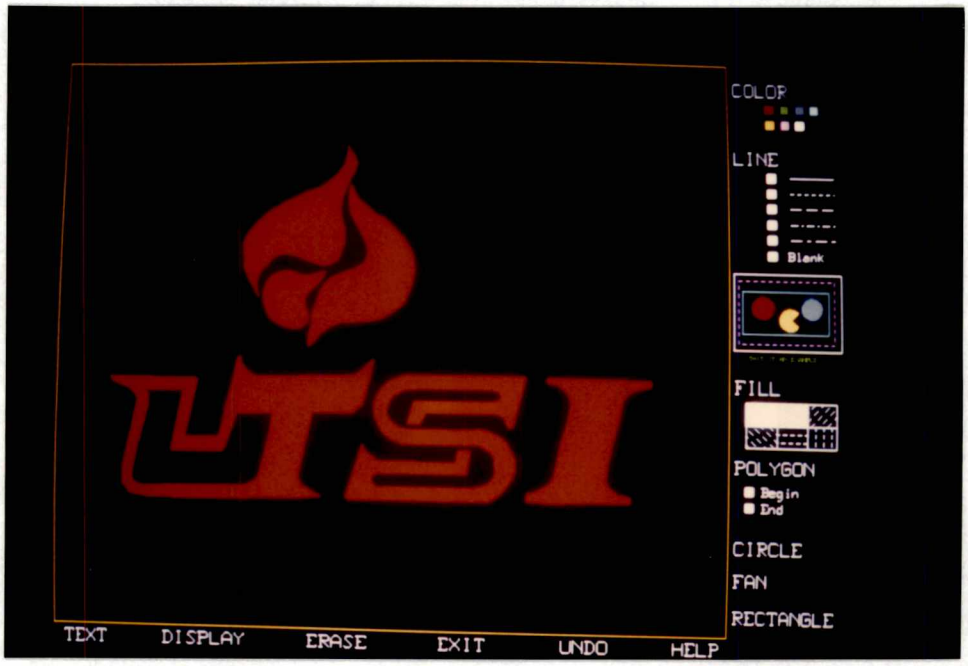


Figure 21. UTSI Emblem (JRC)

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Ramsey, H. R. and Atwood, M. E. "Human Factors in Computer Science Systems: A Review of Literature," Science Applications, Inc. Report SAI-7911-DEN, Englewood, Colorado, September 1980.

2. GRAPAC II Programming Manual. Japan: Japan Radio Tokyo, 1982.

3. JRC Hardware Manual. Japan: Japan Radio Tokyo, 1982.

4. Foley, J. D. and Van Dam, A. Fundamentals of Interactive Computer Graphics. Phillipines: Addison-Wesley Publishing Company, 1982.

5. Foley, J. D. and Wallace, V. L. "The Art of Natural Graphics Man-Machine Conversation," Proceedings of the IEEE, 62 (No. 4): 44-60, April 1974.

6. Eason, K. D. "Understand the Naiver Computer User," The Computer Journal, A (No. 1):3-7, 1976.

7. Foley, J. D., Wallace, V. L., and Chan, P. "The Human Factors of Computer Graphics Interaction Techniques," IEEE Computer Graphics and Applications, Pp. 13-48, November 1984.

8. Snowberry, et al., K. "Computer Display Menus," Erognomics, 26 (No. 7):699-712, 1983.

9. Coffee, J. L. "A Comparison of Vertical and Horizontal Arrangements of Alpha-Numerical Materials," Human Factors, 3:93-98, 1961.

10. Marcus, Aaron. "Corporate Identity for Ionic Interface," IEEE Computer Graphics and Applications, Pp. 24-32, 1984.

11. Tagg, S. K. "The User Interface of the Data Analysis Package," International Journal of Man-Machine Studies, 14:297-311, 1981.

12. Stewart, T. F. "Displays and the Software Interface," Applied Ergonomics, 7(3):137-146, 1976.

13. Millar, L. A. and Thomas, J. C., Jr. "Behavorial Issues in the Use of Interactive Systems," International Journal of Man-Machine Studies, 99:309-536, 1977.

14. Maguire, M. "An Evaluation of Published Recommendations on the Design of Man-Computer Dialogues," International Journal of Man-Machine Studies, 16:237-261, 1982.

15. Computer Graphics, Special GKS Issue. New York: Association for Computing Machinery, February 1984.

APPENDIX

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     PROGRAM DESIGNED AND IMPLEMENTED BY MARY WATTS JONES FOR PART OF
C     THESIS PROJECT ON A USER INTERFACE USING LOGICAL INPUT DEVICES TO
C     MEET THE NEEDS OF THE CASUAL, NAIVE USER.
C
C     COMPLETED ON AUGUST 27, 1985.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C THIS PROGRAM ALLOWS A USER TO DIGITIZE A PICTURE. THE PROGRAM USES GRAPAC II
C SOFTWARE AND IS LINKED TO GRAP2M/LIB. CURRENTLY THE POSSIBLE PRIMITIVES A
C USER CAN CREATE ARE LINES (2 POINTS ARE ALL THAT IS NEEDED), POLYGONS (MAX OF
C 256 VERTICES), CIRCLES (IF BEGINNING ANGLE NOT EQUAL TO ENDING ANGLE THEN ARC)
C FANS AND TEXT STRINGS. THE PRIMITIVES ARE EACH STORED IN A SEPARATE SEGMENT.
C THIS ALLOWS A PRIMITIVE TO BE FILLED (POLYGON, CIRCLE OR FAN) OR DELETED AT
C ANY TIME DURING A DIGITIZING SESSION.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C THE PROGRAM WILL TAKE THE END POINT OF ONE LINE AND USE IT AS THE BEGINNING
C POINT OF THE NEXT UNLESS A BLANK LINE IS SELECTED FROM THE MENU. THEN A NEW
C SEQUENCE OF LINES BEING CONNECTED WILL BEGIN. THE POINTS USED IN OTHER
C PRIMITIVES ARE TREATED SEPARATE FROM THE LINES.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C TWO MENUS WORK AS THE INTERFACE BETWEEN THE USER AND PROGRAM TO ALLOW HIM TO
C DIGITIZE A PICTURE. THE SOFTWARE PACKAGE ALLOWS FOR A SPLIT SCREEN. THE
C BOTTOM 8 LINES ARE RESERVED AS A MESSAGE AREA. MESSAGES (INSTRUCTIONS,
C PROMPTS, CONFIRMTION, ERRORS) ARE GIVEN IN THIS AREA. A WINDOW WILL BE DEFINED
C AND OUTLINED VISUALLY FOR THE USER TO WORK IN.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C ROUTINES IN PROGRAM LISTED IN ORDER AS THEY APPEAR.
C   CHECK_MEN      DIGITIZE        END             EXAMPLE_DISPLAY
C   MENUS          MENU_DISPLAY    PLACE           DISPLAY_XY
C   DISPLAY_RTH    COLOR_OP        LINE_OP         FILL_OP
C   BEGEND_OP      POLY_FILL       DRAW_FILL       RESTORE_PICTURE
C   GET_PT         GET_SEGMENT     GET_ANGLE       GET_N
C   GET_CHINFO     SAVE_PICTURE    STO_SEG         STO_POINT
C   STO_CF         STO_CHARINFO    PLACE_MARKER    CURSOR
```

```
C   DELETE_MARKER  PICK_MODE            LOCT_MODE       KEYB_MODE
C   MAKE_CIRFAN    FIND_ANGLE           MAKE_RECTANGLE  FIND_INTVAL
C   BEGIN_FILL     FANCIR_FILL          STO_FILL        POLYGON_FILLED
C   TEXT_SUB       INSTRUCTION_MESSAGE  ERROR_MESSAGE   CONFIRM_MESSAGE
C   CHECK_HELP     HELP_COLOR           HELP_LINTYP     HELP_EXPICT
C   HELP_FILL      HELP_POLYGON         HELP_CIRFAN     HELP_RECT
C   HELP_DISPLAY   HELP_TEXT            HELP_ERASE      HELP_EXIT
C   HELP_UNDO      CHOICE_MESSAGES      CHECK_CONFCANC  INPUT_CHOICE
C   INPUT_KEYBOARD
C
     COMMON /NAME/ PICK,KEYB,BUTN,STRK,LOCT,ON,OFF,RED,GREEN,BLUE
     1             ,CYAN,YELLOW,MAGENT,WHITE,POLYLIN,POLYGON,CRCLE
     1             ,FANN,CHARA,DELETED,GOOD,NEW,YES,NO
     INTEGER PICK,KEYB,BUTN,STRK,LOCT,ON,OFF,RED,GREEN,BLUE,CYAN
     1       ,YELLOW,MAGENT,WHITE,POLYLIN,POLYGON,CRCLE,FANN,CHARA,DELETED
     1       ,GOOD,NEW,YES,NO
C DFILE IS VARIBLE INPUT FILE NAME. FILE WILL CONTAIN A PREVIOUS PICTURE
C CREATED.
C D1FILE IS VARIBLE INPUT IFILE NAME. FILE WILL CONTAIN THE SAVED PICTURE.
C LINE IS INPUT LINE FROM INTRO.MEN.
C AGAIN IS INPUT FROM USER TO CONTINUE OR DIGITIZING OR TO END PROGRAM
C EXECUTION .
     CHARACTER*132 DFILE,D1FILE
     CHARACTER*132 LINE
     CHARACTER*3 AGAIN
     CHARACTER*3 ANSWER
     CHARACTER*80 C
     CHARACTER*50 STOCHAR(100)
     DIMENSION ISEGARR(1000,7),STOARR(5000,2),CIRFAN(500,3),ICFARR(500)
     DIMENSION IFILLAR(500,4),RCHARA(100,6)
C ISEGARR HOLDS THE INFORMATION SEGMENTS NO., CODE(IF DELETED ,GOOD OR NEW),
C TYPE(IF POLYLINE,POLYGON,CIRCLE,OR FAN), N(NUMBER OF POINTS PER POLYLINE,
C POLYGON,CIRCLE, OR FAN), IPT(INDICATES WHICH BEGINNING POINT IN STOARR, THE
C COLOR OF THAT PARTICULAR POLYLINE, POLYGON AND ETC., AND THE LINE TYPE.
C
C STOARR HOLDS ALL X,Y COORINATES.
C
C CIRFAN HOLDS ALL CIRCLE AND FAN INFORMATION (RADIUS, BEGINNING ANGLE, ENDING
C ANGLE).
C
C ICFARR HOLDS SEGMENT FOR EACH COORESPONDING CIRCLE AND FAN.
C
C IFILLAR HOLDS THE APPROPRIATE SEGMENT THAT A FILL TOOK PLACE IN, THE COLOR,
```

```
C STYLE, AND FILL TYPES.
C
C STOCHAR HOLDS THE TEXT INPUT FROM KEYBOARD.
C
C RCHARA HOLDS THE SEGMENT AND NUMBER OF CHARACTERS IN A TEXT STRING .
C
C INTRO.MEN IS FILE READ WHICH CONTAINS OPENING INSTRUCTIONS AND COMMENTS
C TO THE USER.
     OPEN(UNIT=2,NAME='INTRO.MEN',TYPE='OLD',READONLY)
     DO I=1,50
       READ(2,'(Q,A)',END=5) LLINE,LINE
       IF(LLINE.NE.0) THEN
         TYPE '(A)', LINE(1:LLINE)
       ELSE
         TYPE'(A)', ' '
       END IF
     END DO
 5   CONTINUE
     CLOSE(UNIT=2)
     ACCEPT '(A)',C
 10  CONTINUE
C NUMERICAL ASSIGNMENT OF LOGICAL DEVICES
     PICK=1
     KEYB=2
     BUTN=3
     STRK=4
     LOCT=5
C NUMERICAL ASSIGNMENT OF ON AND OFF
     ON=1
     OFF=0
C NUMERICAL ASSIGNMENT OF COLORS AVAILABLE FOR THIS PROGRAM
     RED=1
     GREEN=2
     BLUE=3
     CYAN=4
     YELLOW=5
     MAGENT=6
     WHITE=7
C NUMERICAL ASSIGNMENT OF CODES TO FLAG A DELETED OR UNDELETED SEGMENT
     DELETED=0
     GOOD=1
C NUMERICAL ASSIGNMENT OF TYPES OF PRIMITIVES IN THIS PROGRAM
     POLYLIN=1
```

```
      POLYGON=2
      CRCLE=3
      FANN=4
      CHARA=5
C NUMERICAL ASSIGNMENT OF YES AND NO.
      YES=1
      NO=0
C THE FOLLOWING VARIBLES ARE FOR STORING THE SEGMENTS AND THE DATA POINTS
C IN ARRAYS CALLED ISEGARR AND STOARR. NOPEN IS THE SEGMENT TO BE OPENED
C EACH TIME.
      ISEG=0          !COUNTS NUMBER OF SEGMENTS OPENED AND CLOSED
      IPT=0    !COUNTS NUMBER OF POINTS THAT HAS BEEN STORED.
      NOPEN=300       !BEGINNING SEGMENT TO BE OPENED.
      ICF=0           !COUNTS NUMBER OF CIRCLES AND FANS CREATED.
      IF=0      !COUNTS NUMBER OF FILLS.
      ICH=0    !COUNTS NUMBER OF TEXT ENTRIES.
C THE USER HAS THE OPTION TO WORK ON A PREVIOUS PICTURE CREATED. THE USER
C IS TO TYPE IN THE NAME OF THE FILE AND THE ROUTINE TO RECREATE THE PICTURE
C WILL BE PUT ON THE SCREEN. IF HE WANT TO WORK ON A NEW ONE HE IS TO HIT
C RETURN AND BEGIN DIGITIZING.
      TYPE'(A)',' If you wish to work on a previous picture created, type in'
      TYPE'(A)',' name of data file. if not hit return and begin digitizing.'
      ACCEPT '(A)',DFILE
C INITIALIZE PACKAGE, CLEAR BUFFERS, SET TABLET SIZE.
      CALL INITG
      CALL BUFOFF
      CALL RSTWND('ALL')
      CALL FLUSHA
      CALL TBLSIZ(3,2)
      CALL SETMEN(0.73405,1.0,0.0,0.66523)
C GOTO DIGITIZING ROUTINE AND THEN END PROGRAM EXECUTION.
      GOTO (190,200),ICOM
  190 CONTINUE
      CALL DIGITIZE(DFILE,ISEGARR,ISEG,STOARR,IPT,CIRFAN,ICFARR,ICF,IFILLAR,
     1    IF,NOPEN,STOCHAR,RCHARA,ICH)
  200 CONTINUE
      CALL END
      CALL TERMG
C USER HAS OPTION TO SAVE PICTURE WITH ALL ATTRIBUTES AND PRIMITIVES. TWO
C DATA FILES NEED TO BE NAMED. ONE TO SAVE ALL INFORMATION ABOUT THE PICTURE
C AND THE OTHER TO SAVE X,Y COORDINATES.
      TYPE'(A)', ' Do you wish to save picture created(Y or N)?'
      ACCEPT'(A)', ANSWER
```

```
      TYPE'(A)',' '
   20 CONTINUE
      IF(ANSWER.EQ.'Y' .OR. ANSWER.EQ.'y') THEN
        TYPE'(A)',' Enter name of file to save picture'
        ACCEPT'(A)',D1FILE
        OPEN(UNIT=2,NAME=D1FILE,TYPE='NEW',CARRIAGECONTROL='LIST')
        CALL SAVE_PICTURE(ISEGARR,ISEG,STOARR,CIRFAN,ICFARR,ICF,IFILLAR,IF,
     1                    STOCHAR,RCHARA,ICH)
        CLOSE(UNIT=2)
      ELSE
        TYPE'(A)',' Verify answer again'
        ACCEPT'(A)',ANSWER
        IF(ANSWER.EQ.'Y' .OR. ANSWER.EQ.'y') GOTO 20
      END IF
C USER HAS OPTION TO DO ANOTHER DESIGN WITHOUT HAVING TO RERUN THE PROGRAM.
      TYPE '(A)',' DO YOU WISH TO DIGITIZE AGAIN (Y OR N)?'
      ACCEPT '(A)', AGAIN
      IF(AGAIN.EQ.'Y' .OR.  AGAIN.EQ.'y') GOTO 10
      STOP
      END
C*******************************************************************************
      SUBROUTINE DIGITIZE(DFILE,ISEGARR,ISEG,STOARR,IPT,CIRFAN,ICFARR,ICF,
     1                    IFILLAR,IF,NOPEN,STOCHAR,RCHARA,ICH)
C*******************************************************************************
C ROUTINE TO DIGITIZE AND STORE ALL INFORMATION ABOUT THE PICTURE IN THE TABLES
C ISEGARR, STOARR, ICFARR, CIRFAN, IFILLAR, RCHARA, AND STOCHAR.
C ALSO CHECKS MENU AND CALLS APPROPRIATE ROUTINE TO DO MENU REQUEST.
      INCLUDE 'PARA.'
      CHARACTER*132 DFILE
      CHARACTER*80 STOCHAR(100)
      DIMENSION ISEGARR(1000,7),STOARR(5000,2),ICFARR(500),CIRFAN(500,3)
      DIMENSION IFILLAR(500,4),RCHARA(100,6)
      CALL TERMOD('GRAP','SEPA')  !PUTS SCREEN TO HAVE MESSAGE AREA.
C SET MODE TO ECHO, SET COLOR TABLE, PICTURE 5 WILL BE USED TO CREATE PICTURE
C IN.
      CALL IMMODE('ECHO')
      CALL COLSTD
      CALL PICTR(5)
      CALL CHPRE('STROKE')
      CALL CHJUST('LEFT','CENTER')
      CALL GTEXT(7)
C THE MENUS ROUTINE ARE THE BOTTOM ROW MENUS AND MAIN.MEN CONTAINS THE RIGHT
C HAND SIDE MENU.
```

```
      CALL MENUS
      OPEN(UNIT=2,NAME='MAIN.MEN',TYPE='OLD',READONLY)
      CALL MENU_DISPLAY(2)
      CLOSE(UNIT=2)
C DISPLAY EXAMPLE PICTURE IN MENU AREA.
      CALL EXAMPLE_DISPLAY
      CALL GLINE(0)
      CALL CHSIZE(0.07,0.07)
      DO I=1,5
        CALL OPENS(I+50)
        CALL SMARK(I)
        CALL CLOSES
      END DO
      IEC=1
      ISG=1
C SET LOCATE MODE AND BLINKING COLOR FOR SEGMENTS.
      CALL PCVIS(5,ON)
      CALL ECHO(LOCT,1,0)
      CALL ECHO(LOCT,1,1)
      CALL ECHSG(LOCT,1,52)
      CALL GLINE(RED)
C SET BLINKING COLOR OF ANY SEGMENT TO RED.
      CALL ECHCOL(RED)
      CALL ENBDEV(BUTN,1)
      CALL ASSDEV(BUTN,1,LOCT,1)
      CALL ENBDEV(LOCT,1)
C KTIME- TIME ALLOTED FOR WAITING ON USER INPUT.
      KTIME=32767
C OUTLINES THE USER WORK AREA WINDOW.
      CALL GLINE(YELLOW)
      CALL MAKE_RECTANGLE(0.00485,0.93,0.84890,0.21)
C SETS VIEWPORT AND WINDOW TO USER WORKING WINDOW SO CLIPPING WILL OCCUR.
      CALL TVPORT(0.00485,0.84890,0.21,0.93)
      CALL TWIND(0.00485,0.84890,0.21,0.93)
C IF USER HAS INDICATED HE WANTS TO WORK ON A PERVIOUS DATA FILE, THEN READ
C IN DATA FILE AND CALL ROUTINE TO PUT PICTURE BACK UPON THE SCREEN.
      IF(DFILE.NE.' ') THEN
        OPEN(UNIT=2,NAME=DFILE,TYPE='OLD',READONLY)
        CALL RESTORE_PICTURE(2,ISEGARR,ISEG,IPT,STOARR,ICFARR,CIRFAN,ICF,
     1          IFILLAR,IF,NOPEN,ICH,STOCHAR,RCHARA)
        CLOSE(UNIT=2)
        NOPEN=NOPEN+1
      END IF
```

```
C SETS DEFAULT CURSOR.
     CALL ECHO(LOCT,1,IEC)
     CALL ECHLC(1,0.5,0.5)
C FLAGS AND COUNTERS INITIALIZED.
     IFIRST = 0      !FLAG TO INDICATE FIRST PT SO LINE WILL NOT BE DRAWN
     IPFLAG=0        !FLAG TO INDICATE BEGINNING AND END OF POLYGON.
     ICFLAG=0        !FLAG TO INDICATE CIRCLE WAS CREATED.
     IFFLAG=0        !FLAG TO INDICATE FAN WAS CREATED.
     ITFLAG=0        !FLAG TO INDICATE TEXT WAS ENTERED.
     N=0        !NO OF POINTS FOR EACH POLYGON,CIRCLE,POLYLINE,FAN
     ICOLOR=5        !INDICATES BEGINNING COLOR IS YELLOW.
     ILINE=1              !INDICATES BEGINNING LINE STYLE IS SOLID.
     IMARK=0         !USED TO OPEN AND CLOSE SEGMENTS FOR MARKING POINTS.
     KFLAG=0           !INDICATES IF MENU ITEM HAS BEEN PICKED.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C THE FOLLOWING LETS USER ENTER DATA POINTS BY CREATING EITHER A POLYLINE,
C POLYGON, CIRCLE, FAN OR RECTANGLE UNTIL EXIT IS INITIATED. THE USER HAS THE
C CHOICE TO FILL ANY POLYGON, CIRCLE, FAN OR RECTANGLE. HE MAY CHOOSE A
C SPECFIC COLOR OR LINE TYPE. HE MAY ALSO CHOOSE A FILL TYPE. HE HAS THE CHOICE
C TO UNDO ANYTHING HE HAS CREATED. HE MAY ALSO CLEAR THE WORKING AREA AT ANY
C TIME BY CHOOSING ERASE. THIS CLEARS ALL DATA THAT HAS BEEN CREATED THUS FAR
C AND THE USER CAN BEGIN NEW
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 60  CONTINUE
C RECEIVE USER INPUT FROM TABLET.
     CALL AWAIT(KTIME,ID,IN)
     CALL GLOCAT(IL,XL,YL)
C THE FOLLOWING CLEARS OUT THE MESSAGE AREA WITH EACH PASS THROUGH THIS LOOP.
     IF(IPFLAG.NE.1) THEN
       CALL MSGCLR
     ELSE
       CALL MSGERS(8,1)
     END IF
C THE FOLLOWING DELETES THE MARKER THAT SHOWED THE BEGINNING AND ENDING OF A
C CIRCLE OR FAN.
     IF(ICFLAG.EQ.1 .OR. IFFLAG.EQ.1) THEN
       IF(IPFLAG.EQ.0) CALL DELETE_MARKER(IMARK)
       ICFLAG=0
       IFFLAG=0
     END IF
C THE FOLLOWING CALL CHECKS TO SEE IF A MENU ITEM HAS BEEN SELECTED.
     CALL CHECK_MENU(XL,YL,XO,YO,KTIME,IFIRST,IPFLAG,ICFLAG,IFFLAG,ITFLAG,N,
     1  ICOLOR,ILINE,ISEG,IPT,NOPEN,ICF,IMARK,ISEGARR,STOARR,ICFARR,CIRFAN,
```

```
     1  KFLAG,IFILLAR,IF,IEC,STOCHAR,RCHARA,ICH,X1,Y1,JFLAG)
     IF (KFLAG.EQ.2) GOTO 900          !EXIT HAS BEEN PICKED FROM MENU.
     IF (KFLAG.EQ.1) GOTO 60           !MENU ITEM WAS PICKED FROM MENU.
C THE FOLLOWING CHECKS TO SEE IF THE POINT OUTSIDE THE USER WINDOW WAS MEANT
C OR IF IT WAS AN ERROR ON THE USER'S PART. HE IS TO CONFIRM OR CANCEL THIS.
     IF(XL.GT.0.84890) THEN
       CALL INSTRUCTION_MESSAGE(ILINE,1)
       CALL CHOICE_MESSAGES(ILINE,1)
       CALL CHECK_CONFCANC(ICONFIRM)
       IF(ICONFIRM.EQ.NO) GOTO 60
     END IF

     IF(IPFLAG.NE.1) THEN              !POLYLINE IS BEING CREATED.
       IF(IFIRST.EQ.0) THEN
         CALL PLACE_MARKER(IMARK,XL,YL)
         GOTO 70
       END IF
       CALL OPENS(NOPEN)
       N=2
     ELSE                !POLYGON IS BEGIN CREATED.
       N=N+1
       IF(N.EQ.1) THEN                    !PLACE MARKER AT BEGINNING OF
         CALL PLACE_MARKER(IMARK,XL,YL)   !POLYGON AND OPEN SEGMENT FOR
         CALL STO_POINT(XL,YL,IPT,STOARR) !POLYGON TO BE CREATED IN.
         CALL OPENS(NOPEN)
         IF(IFIRST.EQ.0) GOTO 70
       END IF
C THE FOLLOWING RINGS A LONG BELL TO WAR THE USER HE GETTING NEAR THE POSSIBLE
C NUMBER OF VERTICES ALLOWED IN CREATING A POLYGON. THIS BELL RINGS AT N=240.
C THE BELL ALSO RINGS WHEN THE USER HAS REACHED THE LIMIT. AT THIS POINT HE
C IS TO CONFIRM OR CANCEL THE POLYGON CREATED THUS FAR.
       IF(N.EQ.240 .OR. N.EQ.255) THEN
         DO I=1,20
           CALL BELL(ON)
         END DO
         CALL ERROR_MESSAGE(ILINE,7)
       END IF
       IF(N.EQ.255) THEN                    !USER HAS REACHED VERTICE LIMIT.
         CALL INSTRUCTION_MESSAGE(ILINE,21) !INSTRUCTIONS GIVEN FOR HIM TO
         CALL INSTRUCTION_MESSAGE(ILINE,13) !CONFIRM OR CANCEL THE POLYGON.
         CALL CHOICE_MESSAGES(ILINE,1)
         CALL CHECK_CONFCANC(MFLAG)
         IF(MFLAG.EQ.YES) THEN              !IF CONFIRMED THEN LAST POINTS
```

```
          CALL MOVE(XO,YO)                    !ARE STORED AND POLYGON IS
          CALL LINE(XL,YL)                    !COMPLETED.
          IPT1=IPT-N+2
          CALL LINE(STOARR(IPT1,1),STOARR(IPT1,2))
          CALL CLOSES
          CALL STO_POINT(XL,YL,IPT,STOARR)
          CALL STO_POINT(STOARR(IPT1,1),STOARR(IPT1,2),IPT,STOARR)
          N=N+1
          CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,ITFLAG,N,
     1             IPT1,ICOLOR,ILINE)
          CALL CONFIRM_MESSAGE(ILINE,2)
          NOPEN=NOPEN+1
        ELSE
          CALL CLOSES
          CALL DELETE(NOPEN)
        END IF
        IF(JFLAG.EQ.1) THEN    !JFLAG SET MEANS TO RESET XO,YO TO LAST
          XO=X1                !COORIDATE OF LINE SEQUENCE.
          YO=Y1          !IF NOT SET IFIRST IS SET TO ZERO AND
          JFLAG=0       !A NEW SEQUENCE OF CONNECTING LINES WILL
        ELSE                   !BEGIN.
          IFIRST=0
        END IF
        IPFLAG=0               !RESET POLYGON FLAG TO ZERO.
        N=0                    !RESET COUNTER OF COORINATES TO ZERO.
        GOTO 60
      END IF
      IF(N.EQ.25) CALL REWRT
    END IF
    CALL MOVE(XO,YO)
    CALL LINE(XL,YL)
    IF(IPFLAG.EQ.0) THEN       !A LINE IS BEING CREATED. STORE INFO
      CALL STO_POINT(XO,YO,IPT,STOARR)  !IN ISEGARR.
      CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,ITFLAG,N,IPT,
    1              ICOLOR,ILINE)
      N=0                              !RESET COUNTER OF COORINATES TO ZERO.
      CALL CLOSES
      NOPEN=NOPEN + 1              !INCREMENT TO NEXT SEGMENT TO BE OPENED.
    END IF
    CALL STO_POINT(XL,YL,IPT,STOARR)
 70 CONTINUE
C SAVE LAST COORDINATE RECORDED.
    XO=XL
```

```
      YO=YL
C FLAG ROUTINE THAT SEQUENCE OF LINES ARE BEING CREATED.
      IFIRST=1
      GOTO 60
C
 900 CONTINUE
C DISASSOCIATE THE BUTTON AND TABLET.
      CALL DASSDV(BUTN,1,LOCT,1)
C DISABLE THE LOCATE DEVICE AND BUTTON DEVICE.
      CALL DISDEV(BUTN,1)
      CALL DISDEV(LOCT,1)
      CALL FLUSHA
C TURN PICTURE VISIBILITY OFF.
      CALL PCVIS(5,OFF)
C CLEAR SCREEN.
      CALL IMCLR('ALL')
      RETURN
      END
C*************************************************************************************
      SUBROUTINE END
C*************************************************************************************
C DELETES ALL PICTURES AND EVENT RECORDS.
      INCLUDE 'PARA.'
      CALL FLUSHA
      CALL DELPCA
      CALL MSGCOL(2)
      RETURN
      END
C*************************************************************************************
      SUBROUTINE EXAMPLE_DISPLAY
C*************************************************************************************
      INCLUDE 'PARA.'
      CALL GLINE(MAGENT)
      CALL LINTYP(2)
      CALL MOVE(.860563,.652389)
      CALL LINE(.986188,.652389)
      CALL LINE(.986188,.574322)
      CALL LINE(.860563,.574322)
      CALL LINE(.860563,.652389)
      CALL LINTYP(1)
      CALL GLINE(CYAN)
      CALL MOVE(.865299,.639693)
      CALL LINE(.978375,.639693)
```

```
      CALL LINE(.978375,.586042)
      CALL LINE(.865229,.586042)
      CALL LINE(.865229,.639693)
      CALL GLINE(WHITE)
      CALL MOVE(.854999,.661178)
      CALL LINE(.995954,.661178)
      CALL LINE(.995954,.565533)
      CALL LINE(.854999,.565533)
      CALL LINE(.854999,.661178)
      CALL GFILL(RED,RED)
      CALL MOVE(0.89276,.619215)
      CALL FILLC(0.013642)
      CALL GFILL(BLUE,BLUE)
      CALL MOVE(0.955944,0.619215)
      CALL FILLC(0.013642)
      CALL GFILL(YELLOW,YELLOW)
      CALL MOVE(.927887,.604566)
      CALL FILLF(0.013642,42.71,319.76)
      CALL GTEXT(GREEN)
      CALL MOVE(.917887,.556744)
      CALL CHSIZE(.005,0.03)
      CALL TEXT('THIS IS AN EXAMPLE',18)
      RETURN
      END
C*************************************************************************************
      SUBROUTINE CHECK_MENU(XL,YL,XO,YO,KTIME,IFIRST,IPFLAG,ICFLAG,IFFLAG,
     1  ITFLAG,N,ICOLOR,ILINE,ISEG,IPT,NOPEN,ICF,IMARK,ISEGARR,STOARR,
     1  ICFARR,CIRFAN,KFLAG,IFILLAR,IF,IEC,STOCHAR,RCHARA,ICH,X1,Y1,JFLAG)
C*************************************************************************************
C
C THE FOLLOWING ROUTINE CHECKS TO SEE IF A MENU ITEMS HAVE BEEN CHOSEN BY
C MAKING TESTS TO SEE IF THE X,Y COORDINATE COORESPONDS TO ONE OF THE MENUS
C HIT AREAS. IF ONE HAS BEEN CHOSEN THEN THE APPROPRIATE ACTION IS TAKEN AND
C THEN THE ROUTINE IS EXITED.
C
C ERROR CHECKS AND MESSAGES ARE GIVEN TO THE USER.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C PARAMETERS: XL,YL- LAST POINT SELECTED BY USER. XO,YO- PREVIOUS POINT
C             SELECTED. KTIME- AWAIT TIME FOR INPUT. IFIRST- INDICATES A NEW
C             DRAWING WILL BEGIN. IPFLAG- FLAG TO INDICATE POLYGON IS BEING
C             CREATED. ICFLAG- FLAG TO INDICATE CIRCLE WAS CREATED. IFFLAG-
C             FLAG TO INDICATE FAN WAS CREATED. ITFLAG- FLAG TO INDICATE TEXT
```

```
C              WAS INPUT. N- NUMBER OF DATA FOR A PARTICULAR PRIMITIVE. ICOLOR-
C              CURRENT COLOR. ILINE- CURRENT LINE STYLE. ISEG- CURRENT POSITION
C              IN ISEGARR FOR INFORMATION ABOUT EACH PRIMITIVE TO BE STORED.
C              IPT- CURRENT POSITION IN STOARR ARRAY THAT X,Y WILL BE STORED IN.
C              NOPEN- CURRENT SEGMENT NUMBER. ICF- CURRENT POSITION IN CIRFAN AND
C              ICFARR FOR NEXT CIRCLE OR FAN INFORMATION TO BE STORED. IMARK-
C              NUMBER OF TEMPORARY MARKERS ON THE SCREEN. ISEGARR- ARRAY THAT
C              CONTAINS THE MAIN INFORMATION ABOUT THE PRIMITIVES CREATED BY
C              USER. STOARR- ARRAY THAT CONTAINS ALL X,Y RECORDED. ICRARR-
C              ARRAY THAT CONTAINS SEGMENT NUMBERS OF CIRCLES AND FANS. CIRFAN-
C              ARRAY THAT CONTAINS RADIUS, BEGINNING ANGLE, ENDING ANGLE OF ALL
C              CIRCLES AND FANS. KFLAG- FLAG TO INDICATE IF MENU ITEM WAS
C              SELECTED. IFILLAR- ARRAY THAT HOLDS ALL FILL INFORMATION. IF-
C           POSITION IN IFILLAR FOR NEXT FILL OCCURRENCE. IEC- CURRENT ECHO
C              TYPE. STOCHAR- ARRAY THAT CONTAINS ALL TEXT STRINGS INPUT.
C              RCHARA- ARRAY THAT CONTAINS ALL INFORMATION FOR TEXT STRINGS.
C              ICH- CURRENT POSITION IN STOCHAR AND RCHARA FOR NEXT TEXT ENTRY.
C              X1,Y1 SAVES THE LAST POSITION BEFORE A POLYGON IS CREATED. JFLAG-
C              FLAGS THAT X1 AND Y1 ARE TO BE USED AFTER POLYGON IS CREATED TO
C              CONTINUE LINE SEQUENCE CONNECTING.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C STOCHAR- TEXT INPUT FROM KEYBOARD.
C XY- X,Y INPUT FROM KEYBOARD.
      INCLUDE 'PARA.'
      CHARACTER*80 STOCHAR(100)
      CHARACTER*8 XY
      BYTE TEST(4)
      DIMENSION ISEGARR(1000,7),STOARR(5000,2),ICFARR(500),CIRFAN(500,3)
      DIMENSION IFILLAR(500,4),RCHARA(100,6)
C STRING IS INTERGER VALUE OF INPUT FROM KEYBOARD.
      INTEGER STRING(20)
      EQUIVALENCE(TEST,STRING(2))
C KFLAG IS FLAG TO SEE IF MENU ITEM WAS CHOSEN. IF EXIT WAS CHOSEN KFLAG IS
C SET TO 2. IF ANY OTHER WAS CHOSEN KFLAG IS SET TO 1. ELSE IT REMAINS 0.
      KFLAG=0
C THE FOLLOWING TAKES XL,YL COORDINATES AND FINDS INTEGER VALUES OF THEM TO
C CHECK MENU ITEMS WITH.
      IX1=INT(10.0*XL)
      IY1=INT(10.0*YL)
      IX=INT(100.0*XL)
      IY=INT(100.0*YL)
```

```
C
C CHECKS TO SEE IF HELP ROUTINE IS TO ACTIVATED.
     IF((IX1.EQ.7 .OR. IX1.EQ.8) .AND. (IY.GE.18 .AND.IY.LE.20)) THEN
         CALL CHECK_HELP(KTIME,ILINE)
         GOTO 60
     END IF
C
C CHECKS TO SEE IF LAST X,Y RECORDED IS TO BE DISPLAYED. IF SO DISPLAY ROUTINE
C IS CALLED.
     IF(IX.GE.15 .AND. IX.LE.21 .AND. IY.GE.19 .AND. IY.LE.21) THEN
       CALL DISPLAY_XY(STOARR(IPT,1),STOARR(IPT,2))
       GOTO 60
     END IF
C
C CHECKS TO SEE IF TEXT IS TO BE ENTERED THROUGH THE KEYBOARD.
C THIS CAN BE USED FOR SUCH THINGS AS LABELING A PICTURE.
C A SUB-MENU HS BEEN CREATED THAT ALLOWS TH USER TO CHOOSE A PARTICULAR
C CHARACTER SIZE AND A PARTICULAR POSITIONING.
C CH- CHARACTER HEIGHT.
C CW- CHARACTER WIDTH.
C P1- VERTICAL POSITIONING.
C P2- HORIZONTAL POSITIONING.
C INSTRUCTIONS ARE GIVEN ON HOW TO CHOOSE SUBMENU FOR CHARACTER SIZE AND
C   POSITIONING. DEFAULTS ARE AVAILABLE.
C IF USER BEGINS ENTERING TEXT ON KEYBOARD THEN DEFAULTS ARE CHOSEN.
C IF SUBMENU CHOSEN MODE CHANGES TO PICK MODE AND USER IS TO SELECT FROM
C   BOTH THE SIZE AND POSITIONING CHOSES.
C   THEN HE IS TO ENTER TEXT THROUGH KEYBOARD.
C   HAS OPTION TO CANCEL OPERATION.
C AFTER STRING HAS BEEN ENTERED (RETURN INDICATES END OF INPUT) THEN
C   MODE CHANGES TO LOCATE
C   USER IS TO USE CURSOR AND INDICATE WHERE CENTER OF STRING IS TO BE PLACED.
C USER IS TO CONFIRM OR CANCEL POSITIONING OF STRING ON THE SCREEN.
     IF(IX.GE.2 .AND. IX.LE.7 .AND. IY.GE.19 .AND. IY.LE.20) THEN
      IF(IPFLAG.NE.1) THEN
       CALL INSTRUCTION_MESSAGE(ILINE,14)
       CALL CHOICE_MESSAGES(ILINE,4)
       CALL ECHO(KEYB,1,1)
       CALL ENBDEV(KEYB,1)
       CALL AWAIT(KTIME,ID,IN)
       IF(ID.NE.KEYB) THEN
         CALL GLOCAT(IL,X,Y)
         CALL DELETE_MARKER(IMARK)
```

```
      CALL TEXT_SUB(CH,CW,P1,P2,KTIME,ILINE,NOPEN)
      CALL INSTRUCTION_MESSAGE(ILINE,19)
      CALL AWAIT(KTIME,ID,IN)
    ELSE
      CH=0.01
      CW=0.03
      P1=0.0
      P2=1.0
      CALL MSGERS(2,60)
      CALL MSGERS(3,60)
      CALL MSGERS(4,60)
    END IF
    CALL GKEYBD(STRING(2),L)
    CALL LOCT_MODE
    CALL CHOICE_MESSAGES(ILINE,3)
    CALL AWAIT(32767,ID,IN)
    CALL GLOCAT(IL,XL,YL)
    CALL FIND_INTVAL(XL,YL,IERROR,ILINE)
    IF(IERROR.EQ.1) GOTO 60
    CALL OPENS(NOPEN)
    CALL CHSIZE(CH,CW)
    CALL CHUP(P1,P2)
    CALL CHJUST('CENTER','TOP')
    CALL MOVE(XL,YL)
    CALL LINTYP(1)
    CALL GTEXT(ICOLOR)
    CALL TEXT(TEST,L)
    CALL CLOSES
    CALL INSTRUCTION_MESSAGE(ILINE,18)
    CALL CHOICE_MESSAGES(ILINE,1)
    CALL CHECK_CONFCANC(MFLAG)
    IF(MFLAG.EQ.1) THEN
      ITFLAG=1
      CALL STO_POINT(XL,YL,IPT,STOARR)
      CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,ITFLAG,
     1     1,IPT,ICOLOR,ILINE)
      ICH=ICH+1
      DO J=1,L
        STOCHAR(ICH)(J:J)=CHAR(TEST(J))
      END DO
      CALL STO_CHARINFO(RCHARA,ICH,NOPEN,L,CH,CW,P1,P2)
      NOPEN=NOPEN+1
    ELSE
```

```
          CALL DELETE(NOPEN)
         END IF
         CALL LINTYP(ILINE)
         ITFLAG=0
        ELSE
          CALL ERROR_MESSAGE(ILINE,9)
        END IF
        GOTO 60
       END IF
C
C CHECKS TO SEE IF ERASE IS TO BE ACTIVATED. THIS WILL CAUSE EVERYTHING THE
C USER HAS CREATED TO BE ERASED. ALL SEGMENTS ARE DELETED.
C USER IS TO CONFIRM OR CANCEL THIS OPERATION.
C ALL COUNTERS ARE RESET AND THE INITIAL SEGMENT IS SET TO 300.
     IF(IX1.EQ.3 .AND. (IY.GE.18 .AND. IY.LE.19)) THEN
       CALL INSTRUCTION_MESSAGE(ILINE,2)
       CALL CHOICE_MESSAGES(ILINE,1)
       CALL CHECK_CONFCANC(MFLAG)
       IF(MFLAG.EQ.1) THEN
         IF(IPFLAG.EQ.1) THEN
           IPFLAG=0
           IF(N.GT.0) THEN
             CALL CLOSES
             CALL DELETE(NOPEN)
           END IF
         END IF
         IF(ISEG.NE.0) THEN
           CALL DELPC(5)
           IFIRST=0
           ISEG=0
           NOPEN=300
           IPT=0
           ICF=0
           IF=0
           CALL PICTR(5)
         END IF
       END IF
       GOTO 60
     END IF
C
C CHECKS TO SEE IF EXIT IS TO ACTIVATED. THIS CAUSES THE DIGITIZING SESSION
C TO END. NOW IS WHEN ALL DATA FILES WILL BE CREATED.
     IF(IX1.EQ.5 .AND. (IY.EQ.19 .OR. IY.EQ.20)) GOTO 900
```

```
C
C CHECKS TO SEE IF THE UNDO OPTION IS TO BE ACTIVATED. THIS ALLOWS THE
C USER TO DELETE ANY POLYGON OR, POLYLINE, CIRCLE, OR FAN HE HAS CREATED.
C THE APPRORIATE SEGMENT IS DELETED.
C THE MODE CHANGES TO PICK MODE.
C THE USER INSTRUCTED TO PICK PRIMITIVE HE WANTS TO DELETE.
C THE SEGMENT CHOSEN BLINKS.
C THE USER IS TO CONFIRM OR CANCEL THIS DELETEION.
C THE SCREEN IS REWROTE AT THIS TIME.
C THE MAIN TABLE (ISEGARR) IS FLAGGED IN THE SECOND COLUMN THAT SEGMENT HAS BEEN
C    DELETED IF CONFIRMED.
C IF IPFLAG = 1 THEN USER IS STILL IN PROCESS OF CREATING A POLYGON. A ERROR
C   MESSAGE IS GIVEN TO HIM TO WARN HIM TO END POLYGON BEFORE SELECTING OPTION.
     IF(IX1.EQ.6 .AND. (IY.EQ.19 .OR. IY.EQ.20)) THEN
      IF(IPFLAG.NE.1) THEN
       CALL PICK_MODE
       CALL INSTRUCTION_MESSAGE(ILINE,9)
       CALL AWAIT(32767,ID,IN)
       CALL GPICK(IP,IS,IPID)
       CALL LOCT_MODE
       CALL INSTRUCTION_MESSAGE(ILINE,3)
       CALL CHOICE_MESSAGES(ILINE,1)
       CALL CHECK_CONFCANC(MFLAG)
       IF(MFLAG.EQ.YES) THEN
         CALL DELETE(IS)
         CALL DELETE_MARKER(IMARK)
         I=1
         CALL REWRT
         DO WHILE (ISEGARR(I,1).NE.IS)
           I=I+1
         END DO
         IF(ISEGARR(I,1).EQ.IS) THEN
           ISEGARR(I,2)=0
           IF(I.LE.1000 .AND. (ISEGARR((I+1),1).LT.ISEGARR(I,1))) THEN
             IF(IFIRST.NE.0 .AND. ISEGARR(I,3).EQ.POLYLIN) THEN
               XO=STOARR(IPT-1,1)
               YO=STOARR(IPT-1,2)
             END IF
           END IF
         END IF
       END IF
       ELSE
        CALL ERROR_MESSAGE(ILINE,9)
```

```
        END IF
        GOTO 60
      END IF
C
C THE FOLLOWING CHECKS MENU OPTION FOR COLOR CHANGE
C MESSAGE IS DISPLAYED TO TELL USER COLOR HE HAS SELECTED.
     IF(IX.EQ.90) THEN
       IF(IY.EQ.87) THEN
         CALL GLINE(RED)
         ICOLOR=RED
         CALL CONFIRM_MESSAGE(ILINE,9)
         GOTO 60
       ELSE IF(IY.EQ.85) THEN
         CALL GLINE(YELLOW)
         ICOLOR=YELLOW
         CALL CONFIRM_MESSAGE(ILINE,13)
         GOTO 60
       END IF
     ELSE IF(IX.EQ.92) THEN
       IF(IY.EQ.87) THEN
         CALL GLINE(GREEN)
         ICOLOR=GREEN
         CALL CONFIRM_MESSAGE(ILINE,10)
         GOTO 60
       ELSE IF(IY.EQ.85) THEN
         CALL GLINE(MAGENT)
         ICOLOR=MAGENT
         CALL CONFIRM_MESSAGE(ILINE,14)
         GOTO 60
       END IF
     ELSE IF(IX.EQ.94) THEN
       IF(IY.EQ.87) THEN
         CALL GLINE(BLUE)
         ICOLOR=BLUE
         CALL CONFIRM_MESSAGE(ILINE,11)
         GOTO 60
       ELSE IF(IY.EQ.85) THEN
         CALL GLINE(WHITE)
         ICOLOR=WHITE
         CALL CONFIRM_MESSAGE(ILINE,15)
         GOTO 60
       END IF
     END IF
```

```
      IF(IX.EQ.96 .AND. IY.EQ.87) THEN
        CALL GLINE(CYAN)
        ICOLOR=CYAN
        CALL CONFIRM_MESSAGE(ILINE,12)
        GOTO 60
      END IF
C
C THE FOLLOWING CHECKS MENU OPTION FOR LINE TYPE CHANGE.
C     ILINE = 1 IS SOLID LINE
C     ILINE = 2 IS DASHED LINE
C     ILINE = 3 IS LONG DASH LINE
C     ILINE = 4 IS DOT-DASH LINE
C     ILINE = 5 IS LONG DOT-DASH LINE
C     IF LAST OPTION IN LINE STYLE CHOSEN THEN A BLANK LINE IS UNDERSTOOD
C      AND USER CAN BEGIN A NEW LINE SEQUENCE.
      IF(IX.GE.88) THEN
        IF(IY.EQ.78) THEN
          CALL LINTYP(1)
          ILINE=1
          GOTO 60
        ELSE IF(IY.EQ.76) THEN
          CALL LINTYP(2)
          ILINE=2
          GOTO 60
        ELSE IF(IY.EQ.74) THEN
          CALL LINTYP(3)
          ILINE=3
          GOTO 60
        ELSE IF(IY.EQ.72) THEN
          CALL LINTYP(4)
          ILINE=4
          GOTO 60
        ELSE IF(IY.EQ.70) THEN
          CALL LINTYP(5)
          ILINE=5
          GOTO 60
        ELSE IF(IY.EQ.68) THEN
          IFIRST=0
          CALL CONFIRM_MESSAGE(ILINE,1)
          CALL DELETE_MARKER(IMARK)
          IMARK=0
          GOTO 60
        END IF
```

```
      END IF
C
C THE FOLLOWING CHECKS TO SEE IF THE USER HAS CHOSEN THE PICTURE IN
C THE MENU ITEM LIST. IF SO A MESSAGE IS DISPLAYED TO HIM TO TELL HIM HE HAS
C SELECTED THIS PICTURE AND THAT IT IS ONLY A PICTURE.
      IF(IX.GE.85 .AND. IY.GE.56 .AND. IY.LE.66) THEN
        CALL INSTRUCTION_MESSAGE(ILINE,22)
        GOTO 60
      END IF
C
C THE FOLLOWING CHECKS TO SEE IF A POLYGON, CIRCLE OR FAN IS TO BE FILLED.
C THE FILL TYPE "HIT AREA" WAS CHOSEN BY USER. THE FILL STYLE IS SET
C   ACCORDINGLY AND THE BEGIN_FILL ROUTINE IS CALLED.
C IF IPFLAG IS 1 THEN ERROR MESSAGE IS DISPLAYED TELLING USER TO END POLYGON
C  FIRST BEFORE SELECTING THIS OPTION.
      IF(IY.GE.47 .AND. IY.LE.50) THEN
        IF(IX.GE.87 .AND. IX.LE.91) THEN
         IF(IPFLAG.EQ.0) THEN
          CALL FILTYP(1,1)
          CALL BEGIN_FILL(ISEGARR,ISEG,STOARR,IPT,ICFARR,CIRFAN,ICF,ICOLOR
     1                    ,ILINE,IFILLAR,IF,1,1)
         ELSE
          CALL ERROR_MESSAGE(ILINE,9)
         END IF
         GOTO 60
        ELSE IF(IX.GE.91 .AND. IX.LE.95) THEN
         IF(IPFLAG.EQ.0) THEN
          CALL FILTYP(1,2)
          CALL BEGIN_FILL(ISEGARR,ISEG,STOARR,IPT,ICFARR,CIRFAN,ICF,ICOLOR
     1                    ,ILINE,IFILLAR,IF,1,2)
         ELSE
          CALL ERROR_MESSAGE(ILINE,9)
         END IF
         GOTO 60
        ELSE IF(IX.GE.95 .AND. IX.LE.99) THEN
         IF(IPFLAG.EQ.0) THEN
          CALL FILTYP(2,2)
          CALL BEGIN_FILL(ISEGARR,ISEG,STOARR,IPT,ICFARR,CIRFAN,ICF,ICOLOR
     1                    ,ILINE,IFILLAR,IF,2,2)
         ELSE
          CALL ERROR_MESSAGE(ILINE,9)
         END IF
         GOTO 60
```

```
        END IF
      ELSE IF (IY.GE.44 .AND. IY.LE.47) THEN
        IF(IX.GE.87 .AND. IX.LE.91) THEN
         IF(IPFLAG.EQ.0) THEN
          CALL FILTYP(3,2)
          CALL BEGIN_FILL(ISEGARR,ISEG,STOARR,IPT,ICFARR,CIRFAN,ICF,ICOLOR
     1                   ,ILINE,IFILLAR,IF,3,2)
         ELSE
          CALL ERROR_MESSAGE(ILINE,9)
         END IF
         GOTO 60
        ELSE IF(IX.GE.91 .AND. IX.LE.95) THEN
         IF(IPFLAG.EQ.0) THEN
          CALL FILTYP(4,2)
          CALL BEGIN_FILL(ISEGARR,ISEG,STOARR,IPT,ICFARR,CIRFAN,ICF,ICOLOR,
     1                   ILINE,IFILLAR,IF,4,2)
         ELSE
          CALL ERROR_MESSAGE(ILINE,9)
         END IF
         GOTO 60
        ELSE IF(IX.GE.95 .AND. IX.LE.99) THEN
         IF(IPFLAG.EQ.0) THEN
          CALL FILTYP(5,2)
          CALL BEGIN_FILL(ISEGARR,ISEG,STOARR,IPT,ICFARR,CIRFAN,ICF,ICOLOR
     1                   ,ILINE,IFILLAR,IF,5,2)
         ELSE
          CALL ERROR_MESSAGE(ILINE,9)
         END IF
         GOTO 60
        END IF
      END IF
C
C THE FOLLOWING CHECKS TO SEE IF POLYGON IS TO BE CREATED.
C THE USER MUST SELECT THE BEGIN OPTION FIRST TO GET THINGS STARTED.
C AFTER POLYGON IS COMPLETED HE IS TO SELECT THE END OPTION.
C THEN HE IS TO COMFIRM OR CANCEL THE POLYGON HE HAS CREATED.
C VARIBLES:
C    IPFLAG IS SET TO 1 TO FLAG POLYGON IS BEING CREATED.
C      IF IPFLAG IS ALREADY 1 WHEN BEGIN IS SELECTED USER IS TOLD HE HAS ALREADY
C      SELECTED BEGIN.
C    X1 AND Y1 SAVE THE LAST X,Y COORDINATES.
C    JFLAG IS SET TO 1 IF IFIRST (INDICATES NEW DRAWING OR AT BEGINNING OR
C      PROGRAM) IS ONE.
```

```
C    ISTORE IS SET TO POSITION IN STOARR WHERE FIRST POINT OF POLYGON IS STORED
C
C IF USER SELECTS END AND HAS NOT SELECTED BEGIN (IPFLAG=0) THEN ERROR MESSAGE.
C IF N (NO. OF POINTS IN POLYGON IS < 4) THEN SEGMENT IS DELETED AND USER
C IS GIVEN A MESSAGES TELLING HIM HE MUST HAVE AT LEAST 3 VERTICES.
C IF USER CONFIRMS POLYGON (MFLAG=YES) THEN ALL INFORMATION IS SAVED ELSE
C SEGMENT IS DELETED.
     IF(IX.GE.86 .AND. IX.LE.93) THEN
       IF(IY.EQ.38 .OR. IY.EQ.39) THEN
         IF(IPFLAG.EQ.1) THEN
           CALL ERROR_MESSAGE(ILINE,1)
           GOTO 60
         END IF
         CALL INSTRUCTION_MESSAGE(ILINE,4)
         IPFLAG=1
         IF(IFIRST.EQ.1) THEN
           X1=XO
           Y1=YO
           JFLAG=1
         END IF
         ISTORE=IPT+1
         IFIRST=0
         GOTO 60
       ELSE IF(IY.EQ.37 .OR. IY.EQ.36) THEN
         IF(IPFLAG.EQ.1) THEN
          IF(N.NE.0) THEN
           IF(STOARR(ISTORE,1).NE.XO.AND.STOARR(ISTORE,2).NE.YO) THEN
             CALL MOVE(XO,YO)
             CALL LINE(STOARR(ISTORE,1),STOARR(ISTORE,2))
             CALL STO_POINT(STOARR(ISTORE,1),STOARR(ISTORE,2),IPT,
    1                       STOARR)
             N=N+1
           END IF
           IF(N.GE.4) THEN
             CALL CONFIRM_MESSAGE(ILINE,2)
             CALL INSTRUCTION_MESSAGE(ILINE,13)
             CALL CHOICE_MESSAGES(ILINE,1)
             CALL CHECK_CONFCANC(MFLAG)
             IF(MFLAG.EQ.YES) THEN
               CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,
    1           ITFLAG,N,ISTORE,ICOLOR,ILINE)
               CALL CLOSES
               NOPEN = NOPEN + 1
```

```
            ELSE
             CALL CLOSES
             CALL DELETE(NOPEN)
            END IF
           ELSE
            CALL ERROR_MESSAGE(ILINE,6)
           CALL CLOSES
           CALL DELETE(NOPEN)
          END IF
          N=0
          IPFLAG=0
          IF(JFLAG.EQ.1) THEN
            XO=X1
            YO=Y1
            JFLAG=0
          ELSE
           IFIRST=0
          END IF
          CALL DELETE_MARKER(IMARK)
          GOTO 60
         ELSE
          CALL ERROR_MESSAGE(ILINE,6)
           IPFLAG=0
          GOTO 60
         END IF
        ELSE
          CALL ERROR_MESSAGE(ILINE,2)
          GOTO 60
        END IF
       END IF
      END IF

C THE FOLLOWING CHECKS TO SEE IF A CIRCLE OR FAN IS TO BE CREATED. ERROR
C CHECKS ARE INCORPORATED INTO THE ROUTINE TO CHECK FOR ERRORS IN X,Y
C INPUT, RADIUS = 0, X,Y GREATER THEN 1 AND ETC.
C THE USER HAS OPTION TO ENTER DATA THROUGH THE KEYBOARD OR THE TABLET.
C A HELP OPTION IS ALSO GIVEN AT THIS TIME.
C
C MFLAG IS FLAG TO TELL USER CHOICE FROM KEYBOARD HELP TABLET OPTION.
C  IF MFLAG = 0 USER HAS SELECTED DEFAULT AND HAS AVOIDED THIS CHOICE.
C    MAKE_CIRFAN ROUTINE IS CALLED AT THIS TIME.
C  IF MFLAG = 3 USER HAS ACTUALLY SELECTED THE TABLET OPTION.
C    MAKE_CIRFAN ROUTINE IS CALLED AT THIS TIME.
```

```
C  IF MFLAG = 2 HELP IS CALLED.
C    HELP_CIRFAN ROUTINE IS CALLED AT THIS TIME.
C  IF MFLAG = 1 THE KEYBOARD HAS BEEN SELECTED.
C    INPUT_KEYBOARD ROUTINE IS CALLED AT THIS TIME.
C USER CAN CANCEL THIS OPTION IS HE WANTS TO.
C
C IF USER HAS ENTERED RADIUS, BEGINNING AND ENDING ANGLE THROUGH KEYBOARD
C HE CAN ENTER CENTER OF CIRCLE/FAN THROUGH KEYBOARD OR BY USER CURSOR.
     IF(IX.GT.86) THEN
       IERROR=0
       IF(IY.EQ.31 .OR. IY.EQ.32) THEN   !CHECK FOR CIRCLE
         ICFLAG=1
       ELSE
         IF(IY.EQ.27 .OR. IY.EQ.28) IFFLAG=1  !CHECK FOR FAN
       END IF
       IF(ICFLAG.EQ.1 .OR. IFFLAG.EQ.1) THEN
        IF(IPFLAG.EQ.0) THEN
         CALL INSTRUCTION_MESSAGE(ILINE,17)
         CALL CHOICE_MESSAGES(ILINE,2)
         CALL INPUT_CHOICE(MFLAG,ILINE,KTIME,XL,YL)
         IF(MFLAG.EQ.0 .OR. MFLAG.EQ.3) THEN
           CALL MAKE_CIRFAN(KTIME,ICFLAG,R,THS,THE,X1,Y1,IMARK,IERROR,
    1                       ILINE,MFLAG,XL,YL)
         ELSE IF(MFLAG.EQ.1) THEN
           CALL INPUT_KEYBOARD(ILINE,KTIME,IERROR,R,THS,THE)
           IF(IERROR.EQ.0) THEN
             CALL INSTRUCTION_MESSAGE(ILINE,15)
             CALL AWAIT(KTIME,ID,IN)
             IF(ID.EQ.KEYB) THEN
              CALL GKEYBD(STRING(2),L)
              DO J=1,L
                XY(J:J)=CHAR(TEST(J))
                IF((XY(J:J).LT.'0' .OR. XY(J:J).GT.'9')
    1              .AND. XY(J:J).NE.'.') IERROR=1
             END DO
             IF(IERROR.NE.1) THEN
               DECODE(L,100,XY) X1
               IF(X1.GT.1.0) IERROR=1
             END IF
             IF(IERROR.NE.1) THEN
               CALL AWAIT(KTIME,ID,IN)
                CALL GKEYBD(STRING(2),L)
                DO J=1,L
```

```
                XY(J:J)=CHAR(TEST(J))
                IF((XY(J:J).LT.'0' .OR. XY(J:J).GT.'9')
     1              .AND. XY(J:J).NE.'.') IERROR=1
              END DO
              IF(IERROR.NE.1) THEN
                DECODE(L,100,XY) Y1
                IF(Y1.GT.1.0) IERROR=1
              END IF
            END IF
             IF(IERROR.EQ.1) CALL ERROR_MESSAGE(ILINE,8)
 100          FORMAT(F8.6)
            ELSE
              CALL GLOCAT(IL,X1,Y1)
            END IF
          END IF
        ELSE
          GOTO 60
        END IF
        CALL LOCT_MODE
        IF(IERROR.EQ.0) THEN
          CALL OPENS(NOPEN)
          CALL MOVE(X1,Y1)
          IF(ICFLAG.EQ.1) CALL CIRCLE(R,THS,THE)
          IF(IFFLAG.EQ.1) CALL FAN(R,THS,THE)
          N=1
          CALL STO_POINT(X1,Y1,IPT,STOARR)
          CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,
     1         ITFLAG,N,IPT,ICOLOR,ILINE)
          CALL CLOSES
          CALL STO_CF(NOPEN,R,THS,THE,ICF,ICFARR,CIRFAN)
          NOPEN=NOPEN+1
          IF(ICFLAG.EQ.1) CALL CONFIRM_MESSAGE(ILINE,3)
          IF(IFFLAG.EQ.1) CALL CONFIRM_MESSAGE(ILINE,4)
          N=0
        END IF
        ELSE
          CALL ERROR_MESSAGE(ILINE,9)
        END IF
        GOTO 60
      END IF
    END IF
C
C THE FOLLOWING CHECKS TO SEE IF A RECTANGULAR POLYGON IS TO BE CREATED.
```

```
C THE USER IS TO GIVE TWO OPPOSITE CORNERS OF THE RECTANGLE.
C MODE IS CHANGED TO BOX ECHO. WHEN USER GIVES ONE CORNER A BOX WILL
C  APPEAR ON SCREEN TO INDICATE WHAT BOX WILL LOOK LIKE AFTER OTHER
C  CORNER POINT IS CHOSEN.
C ALL INFORMATION IS STORED ACCORDINGLY IN MAIN TABLE (ISEGARR).
     IF((IX1.EQ.8 .OR. IX1.EQ.9) .AND. (IY.GE.22 .AND. IY.LE.24)) THEN
      IF(IPFLAG.EQ.0) THEN
        CALL INSTRUCTION_MESSAGE(ILINE,5)
        CALL OPENS(NOPEN)
        CALL ECHO(LOCT,1,5)
        CALL AWAIT(KTIME,ID,IN)
        CALL GLOCAT(IL,X,Y)
        CALL ECHLC(1,X,Y)
        XX=X
        YY=Y
        ISTORE=IPT+1
        CALL AWAIT(KTIME,ID,IN)
        CALL GLOCAT(IL,X,Y)
        CALL MAKE_RECTANGLE(XX,YY,X,Y)
        CALL STO_POINT(XX,YY,IPT,STOARR)
        CALL STO_POINT(X,YY,IPT,STOARR)
        CALL STO_POINT(X,Y,IPT,STOARR)
        CALL STO_POINT(XX,Y,IPT,STOARR)
        CALL STO_POINT(XX,YY,IPT,STOARR)
        N=5
        IPFLAG=1
        CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,ITFLAG,N
    1               ,ISTORE,ICOLOR,ILINE)
        N=0
        IPFLAG=0
        CALL CLOSES
        NOPEN=NOPEN+1
        CALL ECHO(LOCT,1,IEC)
      ELSE
        CALL ERROR_MESSAGE(ILINE,9)
      END IF
      GOTO 60
     END IF
     GOTO 70
900  CONTINUE
     KFLAG=2
     GOTO 70
60   CONTINUE
```

```
      KFLAG=1
 70   CONTINUE
      RETURN
      END
C*******************************************************************************
      SUBROUTINE MENUS
C*******************************************************************************
C THIS CONTAINS THE BOTTOM ROW MENU OPTIONS AND DISPLAYS THEM ON THE SCREEN.
      CALL OPENS(191)
      CALL CHSIZE(0.015,0.03)
      CALL MOVE(0.018,0.195)
      CALL TEXT('TEXT',4)
      CALL MOVE(0.15,0.195)
      CALL TEXT('DISPLAY',7)
      CALL MOVE(0.33,0.195)
      CALL TEXT('ERASE',5)
      CALL MOVE(0.49,0.195)
      CALL TEXT('EXIT',4)
      CALL MOVE(0.64,0.195)
      CALL TEXT('UNDO',4)
      CALL MOVE(0.78,0.195)
      CALL TEXT('HELP',4)
      CALL CLOSES
      RETURN
      END
C*******************************************************************************
      SUBROUTINE MENU_DISPLAY(IFILE)
C*******************************************************************************
C READS IN RIGHT HAND SIDE MENU FROM FILE MAIN.MEN AND POSITIONS THE ITEMS.
C EACH ITEM IN MAIN.MEN BEGINS WITH A # SIGN TO DISTINGUISH IT FROM COMMENTS.
      BYTE MENU(15,15)
      DO I=1,15
        DO J=1,15
          MENU(I,J)=' '
        END DO
      END DO
      CALL OPENS(196)
      CALL CHSIZE(0.015,0.03)
      CALL CHJUST('CENTER','CENTER')
      N=1
 10   CONTINUE
      READ(IFILE,'(Q,<I>A1)',END=20) I,(MENU(J,N),J=1,I)
      IF(MENU(1,N).EQ.'#') N=N+1
```

```
      GOTO 10
   20 CONTINUE
      J=0
      CALL PLACE(MENU,0.915,0.90,J)
      CALL COLOR_OP
      CALL PLACE(MENU,0.915,0.81,J)
      CALL LINE_OP
      CALL PLACE(MENU,0.915,0.52,J)
      CALL FILL_OP
      CALL PLACE(MENU,0.915,0.42,J)
      CALL CHSIZE(0.010,0.3)
      CALL PLACE(MENU,0.93,0.39,J)
      CALL PLACE(MENU,0.93,0.37,J)
      CALL BEGEND_OP
      CALL CHSIZE(0.015,0.03)
      CALL PLACE(MENU,0.915,0.32,J)
      CALL PLACE(MENU,0.915,0.28,J)
      CALL PLACE(MENU,0.915,0.235,J)
      CALL CLOSES
      RETURN
      END
C*************************************************************************
      SUBROUTINE PLACE(MENU,X,Y,J)
C*************************************************************************
C THIS PLACES THE APPROPRIATE MENU ITEM IN THE CORRESPONDING X,Y POSITION.
      BYTE MENU(15,15)
      J=J+1
      CALL MOVE(X,Y)
      CALL TEXT(MENU(2,J),10)
      RETURN
      END
C*************************************************************************
      SUBROUTINE DISPLAY_XY(XL,YL)
C*************************************************************************
C ROUTINE TO DISPLAY X,Y POINTS SELECTED IN USER WORK WINDOW. THE ROUTINE
C CONVERTS THE COORDINATES FROM REAL TO BYTE STRING SO THEY CAN BE DISPLAYED
C IN A MESSAGE.
      INCLUDE 'PARA.'
C PX IS X COORDINATE CONVERTED TO BYTE FORMAT.
C PY IS Y COORDINATE CONVERTED TO BYTE FORMAT.
      BYTE PX(8),PY(8)
C CONVERSION TAKES PLACE.
      ENCODE(10,100,PX) XL
```

```
      ENCODE(10,100,PY) YL
  100 FORMAT (F8.6)
C DISPLAY PX AND PY IN THE MESSAGE AREA.
      CALL MSGCLR
      CALL MSGCOL(GREEN)
      CALL MSGPOS(2,1)
      CALL MSGPUT(PX,8)
      CALL MSGPOS(2,12)
      CALL MSGPUT(PY,8)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE DISPLAY_RTH(R,THS,THE)
C*******************************************************************************
C ROUTINE TO DISPLAY RADIUS, BEGINNING AND ENDING ANGLES OF A CIRCLE OR FAN AS
C A MESSAGE. THE ROUTINE CONVERTS THE DATA FROM A REAL NUMBER TO A BYTE STRING
C SO IT CAN BE DISPLAYED IN THE MESSAGE AREA.
      INCLUDE 'PARA.'
C RAD IS RADIUS CONVERTED TO BYTE FORMAT.
C BE IS BEGINNING ANGLE CONVERTED TO BYTE FORMAT.
C EE IS ENDING ANGLE CONVERTED TO BYTE FORMAT.
      BYTE RAD(8), BE(5),EE(5)
C CONVERSION TAKES PLACE.
      ENCODE(8,100,RAD) R
      ENCODE(5,110,BE) THS
      ENCODE(5,110,EE) THE
  100 FORMAT(F8.6)
  110 FORMAT(F5.1)
C DISPLAYS THE INFORMAION IN THE MESSAGE AREA.
      CALL MSGPOS(2,21)
      CALL MSGPUT('CENTER POINT',12)
      CALL MSGPOS(3,1)
      CALL MSGPUT(RAD,8)
      CALL MSGPOS(3,21)
      CALL MSGPUT('RADIUS',6)
      CALL MSGPOS(4,1)
      CALL MSGPUT(BE,5)
      CALL MSGPOS(4,21)
      CALL MSGPUT('BEGINNING ANGLE',15)
      CALL MSGPOS(5,1)
      CALL MSGPUT(EE,5)
      CALL MSGPOS(5,21)
      CALL MSGPUT('ENDING ANGLE',12)
```

```
      RETURN
      END
C*******************************************************************************
      SUBROUTINE COLOR_OP
C*******************************************************************************
C ROUTINE TO DISPLAY THE COLOR OPTIONS ON THE MENU.
      DIMENSION X1(4),X2(4),X3(4),X4(4),Y1(4),Y2(4)
      DATA X1/0.9,0.91,0.91,0.9/
     1    X2/0.92,0.93,0.93,0.92/
     1    X3/0.94,0.95,0.95,0.94/
     1    X4/0.96,0.97,0.97,0.96/
     1    Y1/0.88,0.88,0.87,0.87/
     1    Y2/0.86,0.86,0.85,0.85/
      K=1
      CALL DRAW_FILL(X1,Y1,K)              !RED(1)
      CALL DRAW_FILL(X2,Y1,K)              !GREEN(2)
      CALL DRAW_FILL(X3,Y1,K)              !BLUE(3)
      CALL DRAW_FILL(X4,Y1,K)              !CYAN(4)
      CALL DRAW_FILL(X1,Y2,K)              !YELLOW(5)
      CALL DRAW_FILL(X2,Y2,K)              !MAGENT(6)
      CALL DRAW_FILL(X3,Y2,K)              !WHITE(7)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE LINE_OP
C*******************************************************************************
C DISPLAYS THE LINE CHOICES ON THE MENU.
      DIMENSION X1(4),Y1(4),Y2(4),Y3(4),Y4(4),Y5(4),Y6(4)
      DATA X1/0.90,0.91,0.91,0.90/
     1    ,Y1/0.79,0.79,0.78,0.78/
     1    ,Y2/0.77,0.77,0.76,0.76/
     1    ,Y3/0.75,0.75,0.74,0.74/
     1    ,Y4/0.73,0.73,0.72,0.72/
     1    ,Y5/0.71,0.71,0.70,0.70/
     1    ,Y6/0.69,0.69,0.68,0.68/
      K=7
      CALL DRAW_FILL(X1,Y1,K)
      CALL MOVE(0.93,0.785)    !SOLID LINE.
      CALL LINE(0.99,0.785)
      K=7
      CALL DRAW_FILL(X1,Y2,K)
      CALL LINTYP(2)                  !DASH LINE.
      CALL MOVE(0.93,0.765)
```

```
      CALL LINE(0.99,0.765)
      K=7
      CALL DRAW_FILL(X1,Y3,K)
      CALL LINTYP(3)                !LONG DASH LINE.
      CALL MOVE(0.93,0.745)
      CALL LINE(0.99,0.745)
      K=7
      CALL DRAW_FILL(X1,Y4,K)
      CALL LINTYP(4)                !DOT-DASH LINE.
      CALL MOVE(0.93,0.725)
      CALL LINE(0.99,0.725)
      K=7
      CALL DRAW_FILL(X1,Y5,K)
      CALL LINTYP(5)                !LONG DOT-DASH LINE.
      CALL MOVE(0.93,0.705)
      CALL LINE(0.99,0.705)
      K=7
      CALL LINTYP(1)                !RETURN STARTING LINE TO SOLID.
      CALL DRAW_FILL(X1,Y6,K)               !BLANK LINE
      CALL CHSIZE(0.010,0.3)
      CALL MOVE(0.95,0.685)
      CALL TEXT('Blank',5)
      CALL CHSIZE(0.015,0.3)
      RETURN
      END
C*************************************************************************
      SUBROUTINE FILL_OP
C*************************************************************************
C DISPLAYS THE TYPES OF FILL THAT CAN BE USED TO FILL A POLYGON,CIRCLE OR FAN.
      DIMENSION X1(4),X2(4),X3(4),Y1(4),Y2(4)
      DATA X1/0.87,0.91,0.91,0.87/
     1   ,X2/0.91,0.95,0.95,0.91/
     1   ,X3/0.96,0.99,0.99,0.95/
     1   ,Y1/0.50,0.50,0.472,0.472/
     1   ,Y2/0.472,0.472,0.444,0.444/
      K=7
      CALL DRAW_FILL(X1,Y1,K)
      ITYP=1
      CALL POLY_FILL(X2,Y1,ITYP)
      ITYP=2
      CALL POLY_FILL(X3,Y1,ITYP)
      ITYP=3
      CALL POLY_FILL(X1,Y2,ITYP)
```

```
      ITYP=4
      CALL POLY_FILL(X2,Y2,ITYP)
      ITYP=5
      CALL POLY_FILL(X3,Y2,ITYP)
      CALL FILTYP(1,1)
      K=7
      RETURN
      END
C*******************************************************************************
      SUBROUTINE BEGEND_OP
C*******************************************************************************
C DISPLAYS BEGIN AND END OPTION FOR POLYGON ON MENU LIST.
      DIMENSION X1(4),Y1(4),Y2(4)
      DATA X1/0.87,0.88,0.88,0.87/
     1    ,Y1/0.395,0.395,0.385,0.385/
     1    ,Y2/0.375,0.375,0.365,0.365/
      K=7
      CALL DRAW_FILL(X1,Y1,K)
      K=7
      CALL DRAW_FILL(X1,Y2,K)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE POLY_FILL(X1,Y1,ITYP)
C*******************************************************************************
C ACTUALLY DISPLAY A RECTANGLE THAT SHOWS THE FILL TYPES
      DIMENSION X1(4),Y1(4)
      CALL GLINE(7)
      CALL GFILL(7,7)
      CALL MOVE(X1(1),Y1(1))
      CALL LINE(X1(4),Y1(4))
      CALL FILTYP(ITYP,2)
      CALL FILL(X1,Y1,4)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE DRAW_FILL(DX,DY,K)
C*******************************************************************************
C ROUTINE TO DRAW AND FILL A BOX BESIDE MENU ITEMS SO THEY CAN BE PICKED.
      DIMENSION DX(4),DY(4)
      CALL GLINE(K)
      CALL GFILL(K,K)
      CALL MOVE(DX(1),DY(1))
```

```
      CALL LINE(DX(4),DY(4))
      CALL FILL(DX,DY,4)
      K=K+1
      RETURN
      END
C*******************************************************************************
      SUBROUTINE RESTORE_PICTURE(IFILE,ISEGARR,ISEG,IPT,STOARR,ICFARR,CIRFAN,
     1     ICF,IFILLAR,IF,NOPEN,ICH,STOCHAR,RCHARA)
C*******************************************************************************
C THIS ROUTINE RESTORES A PICTURE THAT WAS CREATED AND SAVED AT A ANOTHER TIME.
C THE DATA THAT IS READ IN FROM THE GRAPHICS PICTURE FILE IS STORED IN THE
C TABLES THE SAME AS IF THE USER HAD JUST CREATED THE PICTURE. THE ROUTINE
C CHECKS TO SEE WHAT KIND OF PRIMITIVE OR ATTIBUTE THE STRING CONSIST OF
C AND STRIPS THE INFORMATION ACCORDINGLY.
C
C EACH PRIMITIVE IS SET UP IN THE SAME SEGMENT IS WAS INITIALLY CREATED IN.
C
C    INCLUDE 'PARA.'
C LINE1 IS THE INPUT STRING READ IN. NUM IS USED TO RETRIEVE THE SEGMENT AND
C THEN CONVERT IT TO AN INTEGER NUMBER.
C ST IS USED TO CONVERT CHARACTER STRING TO A BYTE FOR TEXT ROUTINE.
C ALL OTHER ARRAYS ARE SET THE SAME AS IN MAIN.FOR.
      BYTE ST(80)
      CHARACTER*132 LINE1
      CHARACTER*3 NUM
      CHARACTER*80 STOCHAR(100)
      DIMENSION ISEGARR(1000,7),STOARR(5000,2),ICFARR(500),CIRFAN(500,3)
      DIMENSION IFILLAR(500,4),RCHARA(100,6)
C INITIALIZE ALL COUNTERS AND FLAGS TO ZERO. SEE MAIN.FOR FOR DESCRIPTIONS.
      ISEG=0
      IPT=0
      IF=0
      ICF=0
      ICH=0
      IPFLAG=0
      ICFLAG=0
      IFFLAG=0
      ITFLAG=0
C SET INITIAL COLOR TO YELLOW AND THE INITIAL LINE STYLE TO SOLID.
      ICOLOR=YELLOW
      ILINE=1
      CALL GLINE(YELLOW)
C THE FOLLOWING IS A LOOP TO READ IN THE DATA FILE UNTIL END OF FILE. EACH TIME
```

```
C A CHECK ON THE CODE OF THE PRIMITIVE OR ATTRIBUTE IS DONE TO SEE WHAT IT IS.
C EXAMPLE: IF(LINE1(2:2).EQ.'8') CHECKS TO SEE IF COLOR HAS CHANGED.
C CODES:
C   1        LINE
C   2   POLYGON
C   3   CIRCLE
C   4   FAN
C   5        TEXT
C   6        POLYGON FILL
C   7   CIRCLE OR FAN FILL
C   8   COLOR
C   9   LINE STYLE
C
 10  CONTINUE
     READ(2,'(Q,A)',END=999) LLINE,LINE1
     IF(LINE1(2:2).EQ.'8') THEN          !CHECKS TO SEE IF COLOR CHANGE.
       CALL GET_N(10,IC,LINE1)            !IF SO CHANGE TO NEW COLOR.
       IF(IC.NE.ICOLOR) THEN
         CALL GLINE(IC)
         ICOLOR=IC
       END IF
     ELSE IF(LINE1(2:2).EQ.'9') THEN         !CHECKS TO SEE IF LINE CHANGE.
       CALL GET_N(14,IL,LINE1)            !IF SO CHANGE TO NEW LINE STYLE
       IF(IL.NE.ILINE) THEN
         CALL LINTYP(IL)
         ILINE=IL
       END IF
     ELSE IF(LINE1(2:2).EQ.'1') THEN               !CHECKS IF INPUT IS A LINE.
       CALL GET_SEGMENT(9,13,NOPEN,LINE1)    !RESTORE LINE ACCORDINGLY.
       CALL GET_PT(16,23,XL,LINE1)
       CALL GET_PT(25,32,YL,LINE1)
       CALL GET_PT(34,41,XL1,LINE1)
       CALL GET_PT(43,50,YL1,LINE1)
       CALL STO_POINT(XL,YL,IPT,STOARR)
       CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,ITFLAG,2,
     1               IPT,IC,IL)
       CALL STO_POINT(XL1,YL1,IPT,STOARR)
       CALL OPENS(NOPEN)
       CALL MOVE(XL,YL)
       CALL LINE(XL1,YL1)
       CALL CLOSES
     ELSE IF(LINE1(2:2).EQ.'2') THEN               !CHECK IF INPUT IS POLYGON.
       CALL GET_SEGMENT(12,16,NOPEN,LINE1)   !RESTORE POLYGON ACCORDINGLY.
```

```
      NUM=LINE1(17:19)
      DECODE(3,110,NUM) N
      IPFLAG=1
      CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,ITFLAG,N,
     1        IPT+1,IC,IL)
      READ(2,'(2(F8.6,1X))',END=999) XL,YL
      CALL STO_POINT(XL,YL,IPT,STOARR)
      ISTORE=IPT
      CALL OPENS(NOPEN)
      DO I=1,N-1
        CALL MOVE(XL,YL)
        READ(2,'(2(F8.6,1X))',END=999) XL,YL
        CALL STO_POINT(XL,YL,IPT,STOARR)
        CALL LINE(XL,YL)
      END DO
      CALL CLOSES
      IPFLAG=0
    ELSE IF(LINE1(2:2).EQ.'3') THEN              !CHECK IF INPUT IS A CIRCLE.
      CALL GET_SEGMENT(11,14,NOPEN,LINE1)   !RESTORE ACCORDINGLY.
      CALL GET_PT(16,23,R,LINE1)
      CALL GET_ANGLE(25,30,THS,LINE1)
      CALL GET_ANGLE(32,37,THE,LINE1)
      CALL GET_PT(39,46,XL,LINE1)
      CALL GET_PT(48,55,YL,LINE1)
      ICFLAG=1
      CALL STO_POINT(XL,YL,IPT,STOARR)
      CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,ITFLAG,
     1        1,IPT,IC,IL)
      CALL STO_CF(NOPEN,R,THS,THE,ICF,ICFARR,CIRFAN)
      ICFLAG=0
      CALL OPENS(NOPEN)
      CALL MOVE(XL,YL)
      CALL CIRCLE(R,THS,THE)
      CALL CLOSES
    ELSE IF(LINE1(2:2).EQ.'4') THEN              !CHECK IF INPUT A FAN.
      CALL GET_SEGMENT(8,11,NOPEN,LINE1)    !RESTORE FAN ACCORDINGLY
      CALL GET_PT(13,20,R,LINE1)
      CALL GET_ANGLE(22,27,THS,LINE1)
      CALL GET_ANGLE(29,34,THE,LINE1)
      CALL GET_PT(36,43,XL,LINE1)
      CALL GET_PT(45,52,YL,LINE1)
      IFFLAG=1
      CALL STO_POINT(XL,YL,IPT,STOARR)
```

```
   CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,ITFLAG,
 1         1,IPT,IC,IL)
   CALL STO_CF(NOPEN,R,THS,THE,ICF,ICFARR,CIRFAN)
   IFFLAG=0
   CALL OPENS(NOPEN)
   CALL MOVE(XL,YL)
   CALL FAN(R,THS,THE)
   CALL CLOSES
 ELSE IF(LINE1(2:2).EQ.'5') THEN                !CHECK IF INPUT IS TEXT STRING.
   CALL GET_SEGMENT(9,12,NOPEN,LINE1)   !RESTORE TEXT ACCORDINGLY.
   CALL GET_PT(14,21,XL,LINE1)
   CALL GET_PT(23,30,YL,LINE1)
   NUM=LINE1(32:34)
   DECODE(3,110,NUM) L
   CALL GET_CHINFO(36,40,CH,LINE1)
   CALL GET_CHINFO(42,46,CW,LINE1)
   CALL GET_CHINFO(48,52,P1,LINE1)
   CALL GET_CHINFO(54,58,P2,LINE1)
   READ(2,'(<L>A)',END=999) LINE1
   DO I=1,L
     ST(I) = ICHAR(LINE1(I:I))
   END DO
   CALL OPENS(NOPEN)
   CALL CHSIZE(CH,CW)
   CALL CHUP(P1,P2)
   CALL CHJUST('CENTER','TOP')
   CALL GTEXT(ICOLOR)
   CALL MOVE(XL,YL)
   CALL TEXT(ST,L)
   CALL CLOSES
   CALL STO_POINT(XL,YL,IPT,STOARR)
   ITFLAG=1
   CALL STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,ITFLAG,1,
1       IPT,IC,IL)
   ITFLAG=0
   ICH=ICH+1
   DO J=1,L
     STOCHAR(ICH)(J:J)=LINE1(J:J)
   END DO
   CALL STO_CHARINFO(RCHARA,ICH,NOPEN,L,CH,CW,P1,P2)
 ELSE IF(LINE1(2:2).EQ.'6'.OR. LINE1(2:2).EQ.'7') THEN
   CALL GET_N(16,IC1,LINE1)          !CHECK IF INPUT IS FILL.
   CALL GET_N(18,ISTYLE,LINE1)    !FILL PRIMITIVE ACCORDINGLY.
```

```
        CALL GET_N(20,ITYPE,LINE1)
        IF(LINE1(2:2).EQ.'6') THEN
          NUM=LINE1(22:24)
          DECODE(3,110,NUM) N
        ELSE
          CALL GET_N(9,N,LINE1)
        END IF
        CALL STO_FILL(IFILLAR,IF,IC1,ISTYLE,ITYPE,NOPEN)
        CALL APPEND(NOPEN)
        CALL LINTYP(ILINE)
        CALL GFILL(IC1,ICOLOR)
        CALL FILTYP(ISTYLE,ITYPE)
        IF(LINE1(2:2).EQ.'6') THEN      !POLYGON FILLED.
          CALL FILL(STOARR(ISTORE,1),STOARR(ISTORE,2),N)
        ELSE IF(N.EQ.3) THEN                    !CIRCLE FILLED.
          CALL FILLC(R)
        ELSE                                !FAN FILLED.
          CALL FILLF(R,THS,THE)
        END IF
        CALL CLOSES
      END IF
      GOTO 10
  110 FORMAT(I3)
  999 CONTINUE
      RETURN
      END
C*************************************************************************
      SUBROUTINE GET_PT(I,J,XL,LINE1)
C*************************************************************************
C ROUTINE TO STRIP (FROM I TO J POSITIONS) THE X OR Y COORDINATE OR THE RADIUS
C FROM THE INPUT LINE AND TO CONVERT IT TO A REAL NUMBER.
      CHARACTER*132 LINE1
      CHARACTER*8 X
      X=LINE1(I:J)
      DECODE(8,115,X) XL
  115 FORMAT(F8.6)
      RETURN
      END
C*************************************************************************
      SUBROUTINE GET_SEGMENT(I,J,NOPEN,LINE1)
C*************************************************************************
C ROUTINE TO STRIP (FROM I TO J POSITONS) THE SEGMENT NUMBER FROM THE INPUT
C LINE AND CONVERT IT TO AN INTERGER NUMBER.
```

```
      CHARACTER*132 LINE1
      CHARACTER*4 SEGMENT
      SEGMENT=LINE1(I:J)
      DECODE(8,110,SEGMENT) NOPEN
  110 FORMAT(I4)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE GET_ANGLE(I,J,TH,LINE1)
C*******************************************************************************
C ROUTINE TO STRIP (FROM I TO J POSITIONS) THE BEGINNING OR ENDING ANGLE FROM
C THE INPUT LINE AND CONVERT IT TO A REAL NUMBER
      CHARACTER*132 LINE1
      CHARACTER*6 ANGLE
      ANGLE=LINE1(I:J)
      DECODE(6,110,ANGLE) TH
  110 FORMAT(F6.2)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE GET_N(I,N,LINE1)
C*******************************************************************************
C ROUTINE TO STRIP SUCH THINGS AS THE COLOR NUMBER, LINE STYLE NUMBER, CODE
C OF THE PRIMITIVE OR ATTIBUTE AND ETC. FROM THE INPUT LINE AND CONVERT IT TO
C AN INTEGER NUMBER.
      CHARACTER*1 NN
      CHARACTER*132 LINE1
      NN=LINE1(I:I)
      DECODE(1,110,NN) N
  110 FORMAT(I1)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE GET_CHINFO(I,J,P,LINE1)
C*******************************************************************************
C ROUTINE TO STRIP (FROM I TO J POSITIONS) THE CHARACTER HEIGHT, CHARACTER
C WIDTH, AND CHARACTER POSITIONING FROM THE INPUT LINE AND CONVERT IT TO A
C REAL NUMBER.
      CHARACTER*5 PP
      CHARACTER*132 LINE1
      PP=LINE1(I:J)
      DECODE(5,110,PP) P
  110 FORMAT(F5.2)
```

```
      RETURN
      END
C*******************************************************************************
      SUBROUTINE SAVE_PICTURE(ISEGARR,ISEG,STOARR,CIRFAN,ICFARR,ICF,
     1     IFILLAR,IF,STOCHAR,RCHARA,ICH)
C*******************************************************************************
C THIS ROUTINE OUTPUTS THE SAVED PICTURE TO A FILE THAT THE USER HAS NAMED.
C THIS FILE WILL BE USED TO RECONSTRUCT A SAVED PICTURE ON SAME MACHINE OR
C ON ANOTHER MACHINE.
      INCLUDE 'PARA.'
C FOR A DESCRIPTION OF THE FOLLOWING STORAGE ARRAYS SEE MAIN.FOR.
      CHARACTER*80 STOCHAR(100)
      DIMENSION ISEGARR(1000,7),STOARR(5000,2),CIRFAN(500,3),ICFARR(500)
      DIMENSION IFILLAR(500,4),RCHARA(100,6)
C SET DEFAULT COLOR AND LINE STYLE.
      IC = YELLOW
      IL=1
C WITH EACH PRIMITIVE OR ATTRIBUTE CHANGE A CODE IS PLACE AT THE BEGINNING OF
C THE OUTPUT LINE.
C CODES:
C   1        LINE
C   2   POLYGON
C   3   CIRCLE
C   4   FAN
C   5   TEXT
C   6   POLYGON FILL
C   7   CIRCLE OR FAN FILL
C   8   COLOR
C   9   LINE STYLE
C
C FOR A LINE THE CODE, TEXT STRING 'LINE', SEGMENT NUMBER, 2(NUMBER OF POINTS),
C X, Y, X1, Y1 (2 END POINTS) ARE OUTPUT TO THE DATA FILE.
C
C FOR A POLYGON THE CODE, 'POLYGON', SEGMENT NUMBER, N(NUMBER OF VERTICES), AND
C LIST OF THE X AND Y VERTICES ARE OUTPUT TO THE DATA FILE.
C
C FOR A CIRCLE THE CODE, 'CIRCLE', SEGMENT NUMBER, RADIUS, BEGINNING ANGLE,
C ENDING ANGLE, AND X AND Y CENTER POINT ARE OUTPUT TO THE DATA FILE.
C
C FOR A FAN THE CODE, 'FAN', SEGMENT NUMBER, RADIUS, BEGINNING ANGLE, ENDING
C ANGLE, AND X AND Y CENTER POINT ARE OUTPUT TO THE DATA FILE.
C
C FOR A TEXT STRING THE CODE, 'TEXT', SEGMENT NUMBER, X AND Y POSITIONING,
```

```
C LENGTH OF TEXT STRING, HEIGTH OF EACH CHARACTER, WIDTH, POSITION (VERTICAL
C VS HORIZONTAL) ARE OUTPUT TO THE DATA FILE.
C
C FOR POLYGON FILL THE FILL CODE, 'FILL', POLYGON CODE (2), SEGMENT NUMBER TO
C APPEND TO, COLOR CODE OF FILL , INTERIOR STYLE NUMBER, HATCHING LINE TYPE,
C NUMBER OF VERTICES INVOLVED, LIST OF VERTICES (X AND Y COORDINATES) ARE
C OUTPUT TO THE DATA FILE.
C
C FOR CIRCLE OR FAN FILL THE FILL CODE, 'FILL', CIRCLE OR FAN CODE, SEGMENT
C NUMBER TO APPEND TO, COLOR OF FILL, INTERIOR STYLE NUMBER, HATCHING LINE
C TYPE, RADIUS, BEGINNING ANGLE, ENDING ANGLE, AND CENTER POINT (X AND Y) ARE
C OUTPUT THE DATA FILE.
C
C FOR COLOR THE CODE, 'COLOR', AND CODE FOR EXACT COLOR ARE OUTPUT TO FILE.
C COLOR CODES:
C   1       RED
C   2       GREEN
C   3   BLUE
C   4   CYAN
C   5   YELLOW
C   6   MAGENT
C   7   WHITE
C
C FOR LINE TYPE THE CODE, 'LINESTYLE', AND LINE STYLE CODE ARE OUTPUT TO FILE.
C LINE STYLE CODES:
C   1   SOLID
C   2       DASH
C   3   LONG DASH
C   4       DOT-DASH
C   5   LONG DOT-DASH
C
C WRITE DEFAULT COLOR AND LINE TO OUTPUT FILE.
     WRITE(2,90) IC
     WRITE(2,95) IL
C FIRST CHECK TO SEE IF SEGMENT HAS BEEN DELETED OR NOT THEN CHECK INCOMING
C COLOR AND LINE WITH IC AND IL. IF DIFFERNT THEN RECORD NEW COLOR AND/OR LINE
C THEN BEGIN RECORDING EACH LINE, POLYGON, CIRCLE, FAN OR TEXT STRING.
     DO I=1,ISEG
       DO K=1,IF
        IF(ISEGARR(I,1).EQ.IFILLAR(K,1)) JJ=K
       END DO
       IF(ISEGARR(I,2).NE.0) THEN
         IF(ISEGARR(I,6).NE.IC) THEN
```

```
      WRITE(2,90) ISEGARR(I,6)
      IC=ISEGARR(I,6)
    END IF
    IF(ISEGARR(I,7).NE.IL) THEN
      WRITE(2,95) ISEGARR(I,7)
      IL=ISEGARR(I,7)
    END IF
    IF(ISEGARR(I,3).EQ.POLYLIN) THEN
      N=ISEGARR(I,5)
      WRITE(2,100) ISEGARR(I,3),ISEGARR(I,1),ISEGARR(I,4),
1          (STOARR(K,1),STOARR(K,2),K=N,N+1)
    ELSE IF(ISEGARR(I,3).EQ.POLYGON) THEN
      N=ISEGARR(I,5)
      N1=ISEGARR(I,4)+ISEGARR(I,5)-1
      N2=ISEGARR(I,4)
      WRITE(2,110) ISEGARR(I,3),ISEGARR(I,1),ISEGARR(I,4),
1     (STOARR(K,1),STOARR(K,2),K=N,N1)
      IF(IFILLAR(JJ,1).EQ.ISEGARR(I,1))
1      WRITE(2,125) ISEGARR(I,3),IFILLAR(JJ,1),IFILLAR(JJ,2),
1        IFILLAR(JJ,3),IFILLAR(JJ,4),ISEGARR(I,4),(STOARR(J,1),
1        STOARR(J,2),J=N,N1)
    ELSE IF(ISEGARR(I,3).EQ.CRCLE) THEN
      K=1
      DO WHILE (ISEGARR(I,1).NE.ICFARR(K))
       K=K+1
      END DO
      WRITE(2,115) ISEGARR(I,3),ISEGARR(I,1),CIRFAN(K,1),CIRFAN(K,2),
1        CIRFAN(K,3),STOARR(ISEGARR(I,5),1),STOARR(ISEGARR(I,5),2)
      IF(ISEGARR(I,1).EQ.IFILLAR(JJ,1))
1       WRITE(2,130) ISEGARR(I,3),ICFARR(K),IFILLAR(JJ,2),IFILLAR(JJ,3)
1         ,IFILLAR(JJ,4),CIRFAN(K,1),CIRFAN(K,2),CIRFAN(K,3),
1          STOARR(ISEGARR(I,5),1),STOARR(ISEGARR(I,5),2)
    ELSE IF(ISEGARR(I,3).EQ.FANN) THEN
      K=1
      DO WHILE (ISEGARR(I,1).NE.ICFARR(K))
       K=K+1
      END DO
      WRITE(2,120) ISEGARR(I,3),ISEGARR(I,1),CIRFAN(K,1),CIRFAN(K,2),
1        CIRFAN(K,3),STOARR(ISEGARR(I,5),1),STOARR(ISEGARR(I,5),2)
      IF(ISEGARR(I,1).EQ.IFILLAR(JJ,1))
1       WRITE(2,130) ISEGARR(I,3),ICFARR(K),IFILLAR(JJ,2),IFILLAR(JJ,3)
1         ,IFILLAR(JJ,4),CIRFAN(K,1),CIRFAN(K,2),CIRFAN(K,3),
1          STOARR(ISEGARR(I,5),1),STOARR(ISEGARR(I,5),2)
```

```
          ELSE IF(ISEGARR(I,3).EQ.CHARA) THEN
            K=1
            DO WHILE(ISEGARR(I,1).NE.INT(RCHARA(K,1)))
             K=K+1
            END DO
            N=ISEGARR(I,5)
            WRITE(2,135) ISEGARR(I,3),ISEGARR(I,1),STOARR(N,1),STOARR(N,2),
     1                   INT(RCHARA(K,2)),(RCHARA(K,J),J=3,6)
            L=INT(RCHARA(K,2))
           WRITE(2,140) STOCHAR(K)(1:L)
           END IF
         END IF
       END DO
  90   FORMAT(1X,'8',1X,'COLOR',1X,I1)
  95   FORMAT(1X,'9',1X,'LINESTYLE',1X,I1)
  100  FORMAT(1X,I1,1X,'LINE',1X,I4,1X,I1,4(1X,F8.6))
  110  FORMAT(1X,I1,1X,'POLYGON',1X,I4,1X,I3,<N2>(/,1X,F8.6,1X,F8.6))
  115  FORMAT(1X,I1,1X,'CIRCLE',1X,I4,1X,F8.6,2(1X,F6.2),2(1X,F8.6))
  120  FORMAT(1X,I1,1X,'FAN',1X,I4,1X,F8.6,2(1X,F6.2),2(1X,F8.6))
  125    FORMAT(1X,'6',1X,'FILL',1X,I1,1X,I4,3(1X,I1),1X,I3,
     1        <N2>(/,1X,F8.6,1X,F8.6))
  130  FORMAT(1X,'7',1X,'FILL',1X,I1,1X,I4,3(1X,I1),1X,F8.6,2(1X,F6.2),
     1      2(1X,F8.6))
  135  FORMAT(1X,I1,1X,'TEXT',1X,I4,2(1X,F8.6),1X,I3,4(1X,F5.3))
  140  FORMAT(A)
       CLOSE(UNIT=10)
       RETURN
       END
C*******************************************************************************
       SUBROUTINE STO_SEG(ISEG,ISEGARR,NOPEN,IPFLAG,ICFLAG,IFFLAG,
     1                            ITFLAG,N,IPT,ICOLOR,ILINE)
C*******************************************************************************
C ROUTINE TO STORE THE CURRENT OPEN SEGMENT AND FLAG THE SEGMENT WITH A 1 TO
C INDICATE SEGMENT IS GOOD. INDICATES IF A POLYGON, POLYLINE, CIRCLE OR FAN.
C INDICATES NUMBER OF POINTS INVOLVED. INDICATES CORRESPONDING POINT IN STOARR,
C INDICATES COLOR, AND INDICATES LINE TYPE.
       INCLUDE 'PARA.'
       DIMENSION ISEGARR(1000,7)
       ISEG=ISEG+1
       ISEGARR(ISEG,1)=NOPEN
       ISEGARR(ISEG,2)= GOOD
       IF(IPFLAG.EQ.1) THEN
         ISEGARR(ISEG,3)=POLYGON
```

```
      ELSE IF(ICFLAG.EQ.1) THEN
          ISEGARR(ISEG,3) = CRCLE
      ELSE IF(IFFLAG.EQ.1) THEN
          ISEGARR(ISEG,3) = FANN
      ELSE IF(ITFLAG.EQ.1) THEN
          ISEGARR(ISEG,3)=CHARA
      ELSE
          ISEGARR(ISEG,3)= POLYLIN
      END IF
      ISEGARR(ISEG,4)=N
      ISEGARR(ISEG,5)=IPT
      ISEGARR(ISEG,6)=ICOLOR
      ISEGARR(ISEG,7)=ILINE
      RETURN
      END
C*************************************************************************
      SUBROUTINE STO_POINT(XL,YL,IPT,STOARR)
C*************************************************************************
C ROUTINE TO STORE X,Y IN THE ARRAY STOARR.
      DIMENSION STOARR(5000,2)
      IPT=IPT+1
      STOARR(IPT,1)=XL              !STORE X,Y COORDINATES.
      STOARR(IPT,2)=YL
      RETURN
      END
C*************************************************************************
      SUBROUTINE STO_CF(NOPEN,R,THS,THE,ICF,ICFARR,CIRFAN)
C*************************************************************************
C ROUTINE TO STORE RADIUS(R), BEGINNING ANGLE(THS), ENDING ANGLE(THE) FOR
C CIRCLES AND FANS.
      DIMENSION ICFARR(500),CIRFAN(500,3)
      ICF=ICF+1
      ICFARR(ICF)=NOPEN
      CIRFAN(ICF,1)=R
      CIRFAN(ICF,2)=THS
      CIRFAN(ICF,3)=THE
      RETURN
      END
C*************************************************************************
      SUBROUTINE STO_CHARINFO(RCHARA,ICH,NOPEN,L,CH,CW,P1,P2)
C*************************************************************************
C ROUTINE TO STORE TEXT STRING INFORMATION. THE SEGMENT(NOPEN), THE CHARACTER
C LENGTH(L), CHARACTER HEIGHT(CH), CHARACTER WIDTH(CW), CHARACTER POSITIONING
```

```
C HORIZONTAL VS. VERTICAL(P1 AND P2)
      DIMENSION RCHARA(100,6)
      RCHARA(ICH,1)=NOPEN
      RCHARA(ICH,2)=L
      RCHARA(ICH,3)=CH
      RCHARA(ICH,4)=CW
      RCHARA(ICH,5)=P1
      RCHARA(ICH,6)=P2
      RETURN
      END
C*************************************************************************************
      SUBROUTINE PLACE_MARKER(IMARK,XL,YL)
C*************************************************************************************
C THIS ROUTINE PLACES A TEMPORARY MARKER SO THE USER CAN SEE POINT SELECTED.
      IMARK=IMARK+1
      CALL OPENS(IMARK)
      CALL CHSIZE(0.01,0.01)
      CALL MARK(XL,YL)
      CALL CLOSES
      RETURN
      END
C*************************************************************************************
      SUBROUTINE CURSOR(IEC)
C*************************************************************************************
C ROUTINE THAT SETS THE ECHO AND CHANGES THE CURSOR.
      INCLUDE 'PARA.'
      CALL ECHLC(1,0.5,0.5)
      CALL ECHO(LOCT,1,0)
      CALL ECHO(LOCT,1,IEC)
      RETURN
      END
C*************************************************************************************
      SUBROUTINE DELETE_MARKER(IMARK)
C*************************************************************************************
C ROUTINE WILL DELETE THE TEMPORY MARKERS.
      DO I=1,IMARK
        CALL DELETE(I)
      END DO
      RETURN
      END
C*************************************************************************************
      SUBROUTINE PICK_MODE
C*************************************************************************************
```

```
C CHANGES FROM LOCATE MODE TO PICK MODE
      INCLUDE 'PARA.'
      CALL DASSDV(BUTN,1,LOCT,1)
      CALL DISDEV(LOCT,1)
      CALL ECHO(PICK,1,1)
      CALL ENBDEV(PICK,1)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE LOCT_MODE
C*******************************************************************************
C CHANGES FROM PICK OR KEYBOARD MODE TO LOCATE MODE.
      INCLUDE 'PARA.'
      CALL DISDEV(PICK,1)
      CALL DISDEV(KEYB,1)
      CALL ECHO(LOCT,1,1)
      CALL ASSDEV(BUTN,1,LOCT,1)
      CALL ENBDEV(LOCT,1)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE KEYB_MODE
C*******************************************************************************
C CHANGES FROM LOCATE MODE TO KEYBOARD MODE.
      INCLUDE 'PARA.'
      CALL DASSDV(BUTN,1,LOCT,1)
      CALL DISDEV(LOCT,1)
      CALL ECHO(KEYB,1,1)
      CALL ENBDEV(KEYB,1)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE MAKE_CIRFAN(KTIME,IFLAG,R,THS,THE,XO,YO,IMARK,IERROR,
     1                       ILINE,MFLAG,XL,YL)
C*******************************************************************************
C THIS ROUTINE IS THE BEGINNING ROUTINE TO FINDING THE RADIUS, BEGINNING AND
C ENDING ANGLE OF EITHER A CIRCLE OR FAN.
      INCLUDE 'PARA.'
C INSTRUCTIONS ARE GIVEN BASED UPON WHETHER THE USER ACCEPTED THE DEFAULT
C ON CIRCLE AND FAN INPUT OR SELECTED THE TABLET FROM THE CHOICE.
      IF(MFLAG.EQ.0) THEN
        IF(IFLAG.EQ.1) THEN
          CALL INSTRUCTION_MESSAGE(ILINE,10)
```

```
          GOTO 10
        ELSE
          CALL INSTRUCTION_MESSAGE(ILINE,11)
          GOTO 10
        END IF
      ELSE IF(IFLAG.EQ.1) THEN
        CALL INSTRUCTION_MESSAGE(ILINE,6)
      ELSE
        CALL INSTRUCTION_MESSAGE(ILINE,7)
      END IF
C GET USER INPUT FOR CENTER OR CIRCLE, ARC OR FAN.
      CALL AWAIT(KTIME,ID,IN)
      CALL GLOCAT(IL,XL,YL)
 10   CONTINUE
      CALL INSTRUCTION_MESSAGE(ILINE,12)
      CALL CHOICE_MESSAGES(ILINE,3)
      CALL PLACE_MARKER(IMARK,XL,YL)
C CHECK TO SEE IF CANCEL OPTION CHOSEN. IF YES EXIT FROM ROUTINE.
      CALL FIND_INTVAL(XL,YL,IERROR,ILINE)
      IF (IERROR.EQ YES) GOTO 20
      XO=XL
      YO=YL
C GET FIRST POINT TO CALCULATE THE BEGINNING ANGLE.
      CALL FIND_ANGLE(KTIME,XL,YL,XO,YO,THS,IERROR)
C CHECK TO SEE IF USER CHOSE TO CANCEL INSTEAD. IF YES EXIT FROM ROUTINE.
      CALL FIND_INTVAL(XL,YL,IERROR,ILINE)
      IF (IERROR.EQ.YES) GOTO 20
      CALL PLACE_MARKER(IMARK,XL,YL)
      IF (IERROR.EQ.1) THEN       !ERROR IN FINDING THE BEGINNING ANGLE.
        CALL DELETE_MARKER(IMARK) !RETURN TO DIGITIZE ROUTINE.
        GOTO 20
      END IF
C CALCULATE RADIUS SQUARED USING C^2 = A^2 + B^2.
      R=(XL-XO)*(XL-XO)+(YL-YO)*(YL-YO)
      IF(R.EQ.0) THEN                  !RADIUS CANNOT BE ZERO-ERROR
        CALL ERROR_MESSAGE(ILINE,3)    !RETURN TO DIGITIZE ROUTINE
        IERROR=1
        GOTO 20
      END IF
C CALCULATE RADUIS.
      R=SQRT(R)
C GET SECOND POINT TO DETERMINE ENDING ANGLE.
      CALL FIND_ANGLE(KTIME,XL,YL,XO,YO,THE,IERROR)
```

```
C CHECK TO SEE IF USER CHOSE TO CANCEL INSTEAD. IF YES EXIT FROM ROUTINE.
      CALL FIND_INTVAL(XL,YL,IERROR,ILINE)
      IF (IERROR.EQ.YES) GOTO 20
      CALL PLACE_MARKER(IMARK,XL,YL)
      IF(IERROR.EQ.1) THEN        !ERROR IN FINDING ENDING ANGLE.
        CALL DELETE_MARKER(IMARK) !RETURN TO DIGITIZE ROUTINE.
        GOTO 20
      END IF
C DISPLAY THE CENTER POINT.
      CALL DISPLAY_XY(XO,YO)
C DISPLAY THE RADIUS, BEGINNING ANDLE, AND ENDING ANGLE.
      CALL DISPLAY_RTH(R,THS,THE)
   20 CONTINUE
      RETURN
      END
C*******************************************************************************
      SUBROUTINE FIND_ANGLE(KTIME,XL,YL,XO,YO,T,IERROR)
C*******************************************************************************
C THIS ROUTINE USES ARCTANGENT TO FIND THE BEGINNING AND ENDING ANGLE.
C AN ERROR CHECK IS DONE EACH TIME. THE CENTER POINT IS USED AS CENTER OF AXIS
C TO DETERMINE WHAT QUADRANT BEGINNING AND ENDING ANGLE ARE IN.
      CALL AWAIT(KTIME,ID,IN)
      CALL GLOCAT(IL,XL,YL)
      IF(XL.GT.XO .AND. YL.GT.YO) THEN       !FIRST QUAD FROM CENTER POINT.
        T=ATAND((YL-YO)/(XL-XO))
      ELSE IF(XL.LT.XO .AND. YL.GT.YO) THEN   !SEC QUAD FROM CENTER POINT.
        T=180.0-ATAND((YL-YO)/(XO-XL))
      ELSE IF(XL.LT.XO .AND. YL.LT.YO) THEN   !THRID QUAD FROM CENTER POINT.
        T= 180.0+ATAND((YO-YL)/(XO-XL))
      ELSE IF(XL.GT.XO .AND. YL.LT.YO) THEN   !FOURTH QUAD FROM CENTER POINT.
        T= 360.0-ATAND((YO-YL)/(XL-XO))
      ELSE IF(XO.EQ.XL) THEN           !THE REST IS DONE IF POSSIBLE
        IF(YL.GT.YO) THEN        !0,90,180,270 OR 360 DEGREE ANGLES.
          T=90.0
        ELSE IF(YL.LT.YO) THEN
          T=270.0
        ELSE
          CALL ERROR_MESSAGE(ILINE,4)
          IERROR=1
        END IF
      ELSE IF(YL.EQ.YO) THEN
        IF(XL.GT.XO) THEN
          T=0.0
```

```
        ELSE IF(XL.LT.XO) THEN
          T=180.0
        ELSE
          CALL ERROR_MESSAGE(ILINE,4)
          IERROR=1
        END IF
      END IF
      RETURN
      END
C*****************************************************************************
      SUBROUTINE MAKE_RECTANGLE(XX,YY,X,Y)
C*****************************************************************************
C ROUTINE TO DRAW A RECTANGLE.
      CALL MOVE(XX,YY)
      CALL LINE(X,YY)
      CALL LINE(X,Y)
      CALL LINE(XX,Y)
      CALL LINE(XX,YY)
      RETURN
      END
C*****************************************************************************
      SUBROUTINE FIND_INTVAL(XL,YL,IERROR,ILINE)
C*****************************************************************************
C ROUTINE TO FIND INTEGER VALUE OF COORINATES AND CHECK CANCEL OPTION.
      IERROR=0
      IX=INT(10.0*XL)
      IY=INT(10.0*YL)
      IF(IX.EQ.8 .AND. IY.EQ.1) THEN
        IERROR=1
        CALL CONFIRM_MESSAGE(ILINE,8)
      END IF
      RETURN
      END
C*****************************************************************************
      SUBROUTINE BEGIN_FILL(ISEGARR,ISEG,STOARR,IPT,ICFARR,CIRFAN,ICF,ICOLOR
     1                      ,ILINE,IFILLAR,IF,ISTYLE,ITYPE)
C*****************************************************************************
C FILL ROUTINE THAT FINDS OUT IF POLYGON, CIRCLE OR FAN IS TO BE FILLED AND
C FILLS IT WITH THE APPROPRIATE FILL STYLE AND COLOR.
C THE APPROPRIATE SEGMENT IS APPENDED TO. THE USER IS TO INDICATE FILL BY
C CLICKING ON OUTTER BOUNDS OF PRIMATIVE TO BE FILLED. IF POLYLINE IS PICKED
C USER IS NOTIFIED OF THIS.
      INCLUDE 'PARA.'
```

```
      DIMENSION ISEGARR(1000,7),STOARR(5000,2),ICFARR(500),CIRFAN(500,3)
      DIMENSION IFILLAR(500,4)
C CHANGE TO PICK MODE SO SEGMENT CAN BE DETECTED.
      CALL PICK_MODE
      CALL INSTRUCTION_MESSAGE(ILINE,8)
C AWAIT USER INPUT.
      CALL AWAIT(32767,ID,IN)
      CALL GPICK(IP,IS,IPID)
      ICFLAG=0
      IFFLAG=0
      DO I=1,ISEG
        IF(ISEGARR(I,1).EQ.IS) THEN
          IF(ISEGARR(I,3).EQ.POLYGON) THEN      !POLYGON TO BE FILLED
             CALL POLYGON_FILLED(ISEGARR,I,STOARR,ICOLOR)
            CALL STO_FILL(IFILLAR,IF,ICOLOR,ISTYLE,ITYPE,IS)
             CALL CONFIRM_MESSAGE(ILINE,5)
         ELSE IF(ISEGARR(I,3).EQ.CRCLE) THEN    !CIRCLE TO BE FILLED
           ICFLAG=1
           CALL FANCIR_FILL(ISEGARR,ICFARR,CIRFAN,ICF,STOARR,IS,I,ICFLAG
     1              ,IFFLAG,ICOLOR,ILINE)
           CALL STO_FILL(IFILLAR,IF,ICOLOR,ISTYLE,ITYPE,IS)
           CALL CONFIRM_MESSAGE(ILINE,6)
        ELSE IF(ISEGARR(I,3).EQ.FANN) THEN      !FAN TO BE FILLED
           IFFLAG=1
           CALL FANCIR_FILL(ISEGARR,ICFARR,CIRFAN,ICF,STOARR,IS,I,ICFLAG
     1              ,IFFLAG,ICOLOR,ILINE)
           CALL STO_FILL(IFILLAR,IF,ICOLOR,ISTYLE,ITYPE,IS)
           CALL CONFIRM_MESSAGE(ILINE,7)
        ELSE IF(ISEGARR(I,3).EQ.POLYLIN) THEN
           CALL ERROR_MESSAGE(ILINE,5)    !A LINE SELECTED TO BE FILLED
        ELSE IF(ISEGARR(I,3).EQ.CHARA) THEN
           CALL ERROR_MESSAGE(ILINE,10)    !TEXT SELECTED TO BE FILLED.
        END IF
       END IF
      END DO
 20   CONTINUE
C CHANGE BACK TO LOCATE MODE.
      CALL LOCT_MODE
C RESTORE PREVIOUS COLOR AND LINE STYLE.
      CALL GLINE(ICOLOR)
      CALL LINTYP(ILINE)
      RETURN
      END
```

```
C*******************************************************************************
      SUBROUTINE FANCIR_FILL(ISEGARR,ICFARR,CIRFAN,ICF,STOARR,IS,I,ICFLAG
     1                       ,IFFLAG,ICOLOR,ILINE)
C*******************************************************************************
C ROUTINE TO FILL EITHER A CIRCLE OR A FAN. IF ICFLAG=1 A CIRCLE IS TO BE
C FILLED ELSE FAN IS TO BE FILLED. THIS IS APPENDED TO THE CORRESPONDING
C SEGMENT.
      INCLUDE 'PARA.'
      DIMENSION ISEGARR(1000,7),ICFARR(500),CIRFAN(500,3),STOARR(5000,2)
      DO J= 1,ICF
        IF(ICFARR(J).EQ.IS) THEN
          CALL APPEND(ISEGARR(I,1))
          CALL LINTYP(ISEGARR(I,7))
          CALL GFILL(ICOLOR,ISEGARR(I,6))
          CALL MOVE(STOARR(ISEGARR(I,5),1),STOARR(ISEGARR(I,5),2))
          IF (IFFLAG.EQ.1) THEN
            CALL FILLF(CIRFAN(J,1),CIRFAN(J,2),CIRFAN(J,3))
          ELSE IF (ICFLAG.EQ.1) THEN
            CALL GLINE(ISEGARR(I,6))
            CALL EDGE(OFF)
            CALL CIRCLE(CIRFAN(J,1),CIRFAN(J,2),CIRFAN(J,3))
            CALL FILLC(CIRFAN(J,1))
          END IF
        CALL CLOSES
        END IF
      END DO
      RETURN
      END
C*******************************************************************************
      SUBROUTINE STO_FILL(IFILLAR,IF,ICOLOR,ISTYLE,ITYPE,IS)
C*******************************************************************************
C THIS ROUTINE STORES THE SEGMENT(IS), COLOR(ICOLOR), STYLE TYPE (ISTYLE), AND
C LINETYPE(ITYPE) OF A PARTICULAR FILLED ITEM.
      DIMENSION IFILLAR(500,4)
C INCREMENT IFILLAR COUNTER(IF) TO GET NEXT POSITION IN ARRAY TO STORE INRO.
      IF=IF+1
      IFILLAR(IF,1)=IS
      IFILLAR(IF,2)=ICOLOR
      IFILLAR(IF,3)=ISTYLE
      IFILLAR(IF,4)=ITYPE
      RETURN
      END
C*******************************************************************************
```

```
      SUBROUTINE POLYGON_FILLED(ISEGARR,I,STOARR,ICOLOR)
C*************************************************************************
C THIS ROUTINE FILLS THE POLYGON THAT WAS SELECTED TO BE FILLED. THE FILL IS
C APPENDED TO THE CORRESPONDING SEGMENT. THE OUTER EDGE TO THE POLYGON IS
C RECONSTRUCTED TO PREVENT THE FILL FROM COVERING THE OUTER EDGE.
      DIMENSION ISEGARR(1000,7),STOARR(5000,2)
      CALL APPEND(ISEGARR(I,1))
      CALL LINTYP(ISEGARR(I,7))
      CALL GFILL(ICOLOR,ISEGARR(I,6))
      CALL FILL(STOARR(ISEGARR(I,5),1),STOARR(ISEGARR(I,5),2),
      1                    ISEGARR(I,4))
      CALL CLOSES
      RETURN
      END
C*************************************************************************
      SUBROUTINE TEXT_SUB(CH,CW,P1,P2,KTIME,ILINE,NOPEN)
C*************************************************************************
C THIS ROUTINE DISPLAYS THE SUB-MENU FOR THE TEXT SIZE AND POSITIONING OPTIONS.
C THE ROUTINE TURNS OFF ALL PRIMITIVES CREATED SO FAR, CHANGES TO PICK MODE
C SO THE USER CAN SELECT THE SIZE AND POSITIONING HE WANTS. EVERY SIZE
C AND POSITIONING IS PUT INTO A SEPARATE SEGMENT.
      INCLUDE 'PARA.'
C SET X AND Y TO INITIAL POSITION FOR FIRST CHARACTER SIZE OPTION.
      X=0.3
      Y=0.6
C SET INITIAL CHARACTER HEIGHT AND WIDTH.
      CH=0.005
      CW=0.03
C SET INITIAL CHARACTER POSITIONING (VERTICAL AND HORIZONTAL).
      P1=0.0
      P2=1.0
C SET FIRST SEGMENT TO BE OPENED FOR PLACING OF THESE OPTIONS.
      N=20
C TURN SEGMENT VISIBILITY OF ALL CREATED SO FAR TO OFF.
      CALL PCVIS(5,OFF)
      CALL PICTR(7)
      CALL PCVIS(7,ON)
C SET INITIAL CHARACTER POSITIONING TO HORIZONTAL.
      CALL CHUP(P1,P2)
C LOOP TO PLACE CHARACTER SIZE OPTIONS ON THE SCREEN, EACH IN A DIFFERENT
C SEGMENT.
      DO I=1,6
        CALL OPENS(N)
```

```
        CALL MOVE(X,Y)
        CALL CHSIZE(CH,CW)
        CALL TEXT('SIZE',4)
        CALL CLOSES
        N=N+1
        CH=CH+0.005
        Y=Y-0.05
      END DO
C ALSO PLACE DEFAULT SIZE ON SCREEN.
      CALL OPENS(N)
      CALL CHSIZE(0.01,0.03)
      CALL MOVE(X,Y)
      CALL TEXT('DEFAULT',7)
      CALL CLOSES
      N=N+1
C SET X AND Y TO INITIAL POSITION FOR FIRST POSITIONING OPTION.
      X=0.5
      Y=0.9
C LOOP TO PLACE POSITIONING OPTIONS ON THE SCREEN, EACH IN A DIFFERENT SEGMENT.
      DO I=1,6
        CALL OPENS(N)
        CALL MOVE(X,Y)
        CALL CHUP(P1,P2)
        CALL TEXT('POSITION',8)
        CALL CLOSES
        N=N+1
        P1=P1+0.2
        P2=P2-0.2
        Y=Y-0.1
      END DO
C ALSO PLACE DEFAULT SIZE ON SCREEN.
      CALL OPENS(N)
      CALL CHUP(0.0,1.0)
      CALL MOVE(X,Y)
      CALL TEXT('DEFAULT',7)
      CALL CLOSES
C CHANGE TO PICK MODE SO OPTIONS IN THE SEGMENTS CAN BE SELECTED.
      CALL PICK_MODE
      J=1
      CALL INSTRUCTION_MESSAGE(ILINE,20)
C RESET INITIAL CHARACTER HEIGHT AND WIDTH AND INITIAL POSITIONING. THIS IS
C SET INITIAL CHARACTER HEIGHT AND WIDTH AND INITIAL POSITIONING TO DEFAULT
C VALUES. THIS IS DONE AS AN ERROR CHECK SO IF SIZE OR POSITONING IS PICKED
```

```
C TWICE THE DEFAULT VALUE WILL BE CHOSEN FOR THE OTHER.
     CH=0.01
     CW=0.03
     P1=0.0
     P2=1.0
C LOOP TO LET THE USER SELECT THE SIZE AND POSITIONING HE WANTS.
C THE SEGMENT IS SELECTED BY PICK AND THE APPROPRIATE SIZE AND POSITIONING
C ARE GIVEN FOR THAT PARTICUALR SEGMENT.
  10 CONTINUE
     CALL AWAIT(KTIME,ID,IN)
     CALL GPICK(IP,IS,IPID)
     IF(IS.EQ.20) THEN
       CH=0.005
     ELSE IF(IS.EQ.21 .OR. IS.EQ.26) THEN
       CH=0.01
     ELSE IF(IS.EQ.22) THEN
       CH=0.015
     ELSE IF(IS.EQ.23) THEN
       CH=0.02
     ELSE IF(IS.EQ.24) THEN
       CH=0.025
     ELSE IF(IS.EQ.25) THEN
       CH=0.03
     ELSE IF(IS.EQ.27 .OR. IS.EQ.33) THEN
       P1=0.0
       P2=1.0
     ELSE IF(IS.EQ.28) THEN
       P1=0.2
       P2=0.8
     ELSE IF(IS.EQ.29) THEN
       P1=0.4
       P2=0.6
     ELSE IF(IS.EQ.30) THEN
       P1=0.6
       P2=0.4
     ELSE IF(IS.EQ.31) THEN
       P1=0.8
       P2=0.2
     ELSE IF(IS.EQ.32) THEN
       P1=1.0
       P2=0.0
     END IF
     J=J+1
```

```
      J=J+1
      IF(J.LE.2) GOTO 10
C CHANGE MODE BACK TO LOCATE MODE.
      CALL DISDEV(PICK,1)
      CALL ASSDEV(BUTN,1,LOCT,1)
      CALL ENBDEV(LOCT,1)
C DELETE SUB-MENU
      DO I=20,35
        CALL DELETE(I)
      END DO
C TURN ALL SEGMENT VISIBILITY BACK ON.
      CALL PCVIS(7,OFF)
      CALL PCVIS(5,ON)
      CALL PICTR(5)
      RETURN
      END
C*************************************************************************
      SUBROUTINE INSTRUCTION_MESSAGE(ILINE,I)
C*************************************************************************
C THIS ROUTINE CONTAINS THE INSTRUCTION MESSAGES. THESE TRY
C TO INFORM THE USER HOW TO GO ABOUT CONSTUCTING A MENU ITEM.
C ALSO TELLS USER THINGS SUCH AS WHEN TO CONFIRM OR CANCEL A OPTION,
C WHAT WILL HAPPEN IF A CERTAIN ITEM IS CHOSEN AND ETC.
      INCLUDE 'PARA.'
      CALL LINTYP(1)
      CALL MSGCOL(YELLOW)
      IF(I.EQ.12) THEN
        CALL MSGPOS(6,55)
        CALL MSGPUT('YOU MAY CANCEL AT ANY TIME',26)
        CALL MSGPOS(7,55)
        CALL MSGPUT('USE TABLET CROSS TO DO SO',25)
        GOTO 10
      ELSE IF(I.EQ.13) THEN
        CALL MSGPOS(6,53)
        CALL MSGPUT('PLEASE CONFIRM OR CANCEL POLYGON',33)
        GOTO 10
      END IF
      CALL MSGCLR
      IF(I.EQ.1) THEN
        CALL MSGPOS(2,1)
        CALL MSGPUT('Please CONFIRM or CANCEL point being outside',44)
        CALL MSGPOS(3,1)
        CALL MSGPUT('Window. Clipping will occur if CONFIRMED.',40)
```

```
ELSE IF(I.EQ.2) THEN
  CALL MSGPOS(2,1)
  CALL MSGPUT('Please CONFIRM or CANCEL ERASING of screen',43)
  CALL MSGPOS(3,1)
  CALL MSGPUT('If erased all will be lost.',28)
ELSE IF (I.EQ.3) THEN
  CALL MSGPOS(2,1)
  CALL MSGPUT('CONFIRM or CANCEL deletion',26)
ELSE IF (I.EQ.4) THEN
  CALL MSGPOS(2,1)
  CALL MSGPUT('After POLYGON has been created',30)
  CALL MSGPOS(3,1)
  CALL MSGPUT('Click on the POLYGON END.',26)
  CALL MSGCOL(CYAN)
  CALL MSGPOS(4,1)
  CALL MSGPUT('If you wish to cancel POLYGON creation, click on the
1 POLYGON END and',69)
  CALL MSGPOS(5,1)
  CALL MSGPUT('choose the cancel option when it appears on the screen.'
1 ,55)
ELSE IF(I.EQ.5) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('Give TWO OPPOSITE POINTS of RECTANGLE',37)
ELSE IF(I.EQ.6) THEN
  CALL MSGPOS(2,1)
  CALL MSGPUT('Pick CENTER of CIRCLE, BEGINNING of',36)
  CALL MSGPOS(3,1)
  CALL MSGPUT('CIRCLE and last pick ENDING of CIRCLE',38)
ELSE IF(I.EQ.7) THEN
  CALL MSGPOS(2,1)
  CALL MSGPUT('Pick CENTER of FAN, BEGINNING of FAN',36)
  CALL MSGPOS(3,1)
  CALL MSGPUT('and last pick ENDING of FAN.',28)
ELSE IF(I.EQ.8) THEN
  CALL MSGPOS(2,1)
  CALL MSGPUT('Click once on outer edge of either a POLYGON',44)
  CALL MSGPOS(3,1)
  CALL MSGPUT('CIRCLE or FAN, then it will be filled with fill type',
1  52)
  CALL MSGPOS(4,1)
  CALL MSGPUT('You have selected.',18)
ELSE IF(I.EQ.9) THEN
  CALL MSGPOS(3,1)
```

```
   CALL MSGPUT('Select outer bound of LINE, POLYGON, CIRCLE, ARC',48)
   CALL MSGPOS(4,1)
   CALL MSGPUT('or FAN to be deleted.',21)
 ELSE IF(I.EQ.10) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('You have chosen center of CIRCLE, now choose',44)
   CALL MSGPOS(3,1)
   CALL MSGPUT('Beginning of CIRCLE and then ending of CIRCLE',45)
   CALL MSGPOS(4,1)
   CALL MSGPUT('Using the tablet and cross.',28)
 ELSE IF(I.EQ.11) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('You have chosen center of FAN, now choose',41)
   CALL MSGPOS(3,1)
   CALL MSGPUT('Beginning of FAN and then ending of FAN',39)
   CALL MSGPOS(4,1)
   CALL MSGPUT('Using the tablet and cross.',28)
 ELSE IF(I.EQ.14) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('Before entering TEXT, you may choose size and positoning
1 ',56)
   CALL MSGPOS(3,1)
   CALL MSGPUT('of text string. You may accept the default size and',51)
   CALL MSGPOS(4,1)
   CALL MSGPUT('positioning (horizontal) and bypass this choice.',48)
   CALL MSGPOS(5,1)
   CALL MSGCOL(CYAN)
   CALL MSGPUT('If you decide on the default begin entering TEXT through
1 ',56)
   CALL MSGPOS(6,1)
   CALL MSGPUT('KEYBOARD (max 50 characters). After hitting return,',50)
   CALL MSGPOS(7,1)
   CALL MSGPUT('position cursor on screen where text is to be put.',50)
   CALL MSGCOL(YELLOW)
   CALL MSGPOS(8,1)
   CALL MSGPUT('THE DELETE KEY CAN BE USED FOR MISTAKES IN KEYBOARD
1 ENTRY.',60)
 ELSE IF(I.EQ.15) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('Enter X,Y point through keyboard (0<=X<=1,0<=Y<=1)',
1     50)
   CALL MSGPOS(3,1)
   CALL MSGPUT('enter X return, enter Y return',30)
```

```
   CALL MSGPOS(4,1)
   CALL MSGPUT('or',2)
   CALL MSGPOS(5,1)
   CALL MSGPUT('Pick center or CIRCLE or FAN by positioning cursor.',51)
 ELSE IF(I.EQ.16) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('Enter RADIUS (>0), BEGINNING ANGLE (In degrees),',48)
   CALL MSGPOS(3,1)
   CALL MSGPUT('ENDING ANGLE (in degrees) through keyboard.',42)
   CALL MSGPOS(4,1)
   CALL MSGPUT('RETURN IS THE END OF INPUT FOR EACH OF THE ABOVE.',48)
 ELSE IF(I.EQ.17) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('To create you have option to enter RADIUS, BEGINNING'
1     ,52)
   CALL MSGPOS(3,1)
   CALL MSGPUT('and ENDING of CIRCLE or FAN through keyboard.',45)
   CALL MSGPOS(4,1)
   CALL MSGPUT('Place cursor on KEYBOARD if this is your choice.',48)
   CALL MSGCOL(CYAN)
   CALL MSGPOS(5,1)
   CALL MSGPUT('Default entry is tablet. You may choose tablet or',49)
   CALL MSGPOS(6,1)
   CALL MSGPUT('You can begin using cursor to indicate the',42)
   CALL MSGPOS(7,1)
   CALL MSGPUT('Center, beginning and ending of CIRCLE or FAN.',46)
   CALL MSGCOL(YELLOW)
   CALL MSGPOS(8,1)
   CALL MSGPUT('FURTHER INSTRUCTIONS WILL BE GIVEN.',35)
 ELSE IF(I.EQ.18) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('PLEASE CONFIRM OR CANCEL POSITONING OF TEXT INPUT',49)
 ELSE IF(I.EQ.19) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('Now begin entering text through keyboard.',41)
   CALL MSGPOS(3,1)
   CALL MSGPUT('Return is the signal for end of input.',38)
 ELSE IF(I.EQ.20) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('You must indicate a character size or choose default.'
1 ,53 )
   CALL MSGCOL(CYAN)
   CALL MSGPOS(3,1)
```

```
        CALL MSGPUT('Also you must indicate a positioning or choose default.'
     1 , 55)
      ELSE IF(I.EQ.21) THEN
        CALL MSGPOS(2,1)
        CALL MSGPUT('You have reached 256 vertices which is the limit on',51)
        CALL MSGPOS(3,1)
        CALL MSGPUT('on a polygon vertices.',23)
      ELSE IF(I.EQ.22) THEN
        CALL MSGPOS(2,1)
        CALL MSGPUT('This is only a picture put here so you can see some',51)
        CALL MSGPOS(3,1)
        CALL MSGPUT('of the objects, color types, line styles and',44)
        CALL MSGPOS(4,1)
        CALL MSGPUT('character string input available for you to create.',51)
      END IF
      IF(I.EQ.6 .OR. I.EQ.10) THEN
        CALL MSGPOS(5,1)
        CALL MSGCOL(CYAN)
        CALL MSGPUT('If BEGINNING POINT = ENDING POINT, then',39)
        CALL MSGPOS(6,1)
        CALL MSGPUT('complete CIRCLE is created.',26)
      END IF
   10 CONTINUE
      CALL LINTYP(ILINE)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE ERROR_MESSAGE(ILINE,I)
C*******************************************************************************
C THIS ROUTINE CONTAINS ALL THE ERROR MESSAGES. THESE MESSAGES
C LET THE USER KNOW SOMETHING WENT WRONG AND ATTEMPTS TO GIVE
C THEM INFORMATION TO CORRECT THE ERROR.
      INCLUDE 'PARA.'
      CALL LINTYP(1)
      CALL MSGCOL(GREEN)
      IF(I.EQ.7) THEN
        CALL MSGPOS(5,1)
        CALL MSGPUT('YOU ARE ONLY ALLOWED 256 VERTICES IN CREATING A',48)
        CALL MSGPOS(6,1)
        CALL MSGPUT('POLYGON. THIS IS AN EARLY WARNING.',35)
        GOTO 10
      ELSE IF(I.EQ.9) THEN
        CALL MSGPOS(8,1)
```

```
   CALL MSGPUT('YOU CANNOT SELECT THIS MENU ITEM UNTIL YOU FINISH THE
 1 POLYGON.',62)
   GOTO 10
 END IF
 CALL MSGCLR
 IF (I.EQ.1) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('MUST END PREVIOUS POLYGON BEFORE BEGINNING',42)
   CALL MSGPOS(3,1)
   CALL MSGPUT('A NEW POLYGON',12)
 ELSE IF(I.EQ.2) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('MUST BEGIN POLYGON FIRST',24)
 ELSE IF(I.EQ.3) THEN
   CALL MSGPUT(5,1)
   CALL MSGPUT('YOU CANNOT HAVE A ZERO RADIUS',29)
   CALL MSGPOS(6,1)
   CALL MSGPUT('TO CORRECT THIS WILL HAVE TO SELECT CIRCLE OR FAN',49)
   CALL MSGPOS(7,1)
   CALL MSGPUT('AGAIN',5)
 ELSE IF(I.EQ.4) THEN
   CALL MSGPOS(5,1)
   CALL MSGPUT('ERROR IN CREATING CIRCLE OR FAN',31)
   CALL MSGPOS(6,1)
   CALL MSGPUT('TO CORRECT THIS YOU WILL HAVE TO SELECT CIRCLE
 1              OR FAN AGAIN',60)
 ELSE IF(I.EQ.5) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('YOU HAVE SELECTED A POLYLINE.',29)
 ELSE IF(I.EQ.6) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('POLYGON MUST HAVE AT LEAST THREE VERTICES',41)
   CALL MSGPOS(3,1)
   CALL MSGPUT('ATTEMPTED POLYGON DELETED. YOU WILL HAVE TO TRY
 1 AGAIN',54)
 ELSE IF(I.EQ.8) THEN
   CALL MSGPOS(2,1)
   CALL MSGPUT('ERROR IN ENTERING X,Y COORDINATES',33)
   CALL MSGPOS(3,1)
   CALL MSGPUT('TO CORRECT THIS YOU WILL HAVE TO SELECT CIRCLE
 1              OR FAN AGAIN',60)
 ELSE IF(I.EQ.10) THEN
   CALL MSGPOS(2,1)
```

```
        CALL MSGPUT('YOU HAVE SELECTED A CHARACTER STRING.',37)
      END IF
      IF(I.EQ.5 .OR. I.EQ.10) THEN
        CALL MSGPOS(3,1)
        CALL MSGPUT('TO USE FILL YOU MUST SELECT EITHER A',36)
        CALL MSGPOS(4,1)
        CALL MSGPUT('POLYGON, CIRCLE OR FAN.',23)
      END IF
      CALL LINTYP(ILINE)
   10 CONTINUE
      RETURN
      END
C*******************************************************************************
      SUBROUTINE CONFIRM_MESSAGE(ILINE,I)
C*******************************************************************************
C CONTAINS ALL CONFIRMATION MESSAGE. THESE MESSAGES LET THE USER KNOW THAT
C AN ACTION HAS BEEN RECEIVED OR COMPLETED.
      INCLUDE 'PARA.'
      CALL LINTYP(1)
      CALL MSGCOL(CYAN)
      IF(I.EQ.3) THEN
        CALL MSGPOS(6,1)
        CALL MSGPUT('CIRCLE Completed',17)
        GOTO 10
      ELSE IF(I.EQ.4) THEN
        CALL MSGPOS(6,1)
        CALL MSGPUT('FAN Completed',13)
        GOTO 10
      ELSE IF(I.EQ.9) THEN
        CALL MSGPOS(7,1)
        CALL MSGCOL(RED)
        CALL MSGPUT('COLOR RED',9)
        GOTO 10
      ELSE IF(I.EQ.10) THEN
        CALL MSGPOS(7,1)
        CALL MSGCOL(GREEN)
        CALL MSGPUT('COLOR GREEN',11)
        GOTO 10
      ELSE IF(I.EQ.11) THEN
        CALL MSGPOS(7,1)
        CALL MSGCOL(BLUE)
        CALL MSGPUT('COLOR BLUE',10)
        GOTO 10
```

```
      ELSE IF(I.EQ.12) THEN
        CALL MSGPOS(7,1)
        CALL MSGPUT('COLOR CYAN',10)
        GOTO 10
      ELSE IF(I.EQ.13) THEN
        CALL MSGPOS(7,1)
        CALL MSGCOL(YELLOW)
        CALL MSGPUT('COLOR YELLOW',12)
        GOTO 10
      ELSE IF(I.EQ.14) THEN
        CALL MSGPOS(7,1)
        CALL MSGCOL(MAGENT)
        CALL MSGPUT('COLOR MAGENT',12)
        GOTO 10
      ELSE IF(I.EQ.15) THEN
        CALL MSGPOS(7,1)
        CALL MSGCOL(WHITE)
        CALL MSGPUT('COLOR WHITE',11)
        GOTO 10
      END IF
      CALL MSGCLR
      IF(I.EQ.1) THEN
        CALL MSGPOS(2,1)
        CALL MSGPUT('YOU MAY BEGIN A NEW DRAWING',27)
      ELSE IF(I.EQ.2) THEN
        CALL MSGPOS(2,1)
        CALL MSGPUT('POLYGON completed',17)
      ELSE IF(I.EQ.5) THEN
        CALL MSGPOS(5,1)
        CALL MSGPUT('POLYGON FILLED',14)
      ELSE IF(I.EQ.6) THEN
        CALL MSGPOS(5,1)
        CALL MSGPUT('CIRCLE FILLED',13)
      ELSE IF(I.EQ.7) THEN
        CALL MSGPOS(5,1)
        CALL MSGPUT('FAN FILLED',10)
      ELSE IF(I.EQ.8) THEN
        CALL MSGPOS(2,1)
        CALL MSGPUT('You have selected to cancel this operation',42)
        CALL MSGPOS(4,1)
        CALL MSGPUT('OPERATION CANCELED',18)
      END IF
   10 CONTINUE
```

```
      CALL LINTYP(ILINE)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE CHECK_HELP(KTIME,ILINE)
C*******************************************************************************
C ROUTINE TO INDICATE HELP HAS BEEN ASK FOR. THE ROUTINE DISPLAYS INSTRUCTIONS
C TO GET HELP ON ANY MENU ITEM AND THE CALL THAT ROUTINE APPROPRIATELY.
C ONLY ARGUMENTS NEEDED ARE THE TIME FOR THE AWAIT COMMAND AND THE LINE TYPE.
C THE LINE TYPE IS CHANGED FROM WHATEVER TO SOLID SO MESSAGE WILL BE CLEAR.
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('Now that you have chosen HELP, click',37)
      CALL MSGPOS(3,1)
      CALL MSGPUT('Once over the item you need help on.',37)
      CALL MSGPOS(4,1)
      CALL MSGPUT('REMEMBER!!!!!!!',16)
      CALL MSGPOS(5,1)
      CALL MSGPUT('FOR HELP- always click once over help',38)
      CALL MSGPOS(6,1)
      CALL MSGPUT('And then the item you need help on.',36)
C GET INPUT ON MENU ITEM HELP IS NEEDED ON.
      CALL AWAIT(KTIME,ID,IN)
      CALL GLOCAT(IL,XL,YL)
      IX=INT(100.0*XL)
      IY=INT(100.0*YL)
      CALL LINTYP(1)
      IF(IX.GE.86) THEN
        IF(IY.GE.85 .AND. IY.LE.90) THEN
          CALL HELP_COLOR
        ELSE IF(IY.GE.68 .AND. IY .LE.81) THEN
          CALL HELP_LINTYP
        ELSE IF(IY.GE.55 .AND. IY.LE.65) THEN
          CALL HELP_EXPICT
        ELSE IF(IY.GE.44 .AND. IY.LE.52) THEN
          CALL HELP_FILL
        ELSE IF(IY.GE.36 .AND. IY.LE.42) THEN
          CALL HELP_POLYGON
        ELSE IF(IY.GE.27 .AND. IY.LE.32) THEN
          CALL HELP_CIRFAN
        ELSE IF(IY.GE.22 .AND. IY.LE.24) THEN
```

```
          CALL HELP_RECT
        ELSE
          CALL MSGCLR
          CALL MSGPOS(2,1)
          CALL MSGCOL(CYAN)
          CALL MSGPUT('NO INFO',7)
        END IF
      ELSE IF(IY.LE.20) THEN
        IF(IX.GE.15 .AND. IX.LE.21) THEN
          CALL HELP_DISPLAY
        ELSE IF(IX.GE.2 .AND. IX.LE.7) THEN
          CALL HELP_TEXT
        ELSE IF(IX.GE.30 .AND. IX.LE.39) THEN
          CALL HELP_ERASE
        ELSE IF(IX.GE.50 .AND. IX.LE.59) THEN
          CALL HELP_EXIT
        ELSE IF(IX.GE.60 .AND. IX.LE.69) THEN
          CALL HELP_UNDO
        ELSE
          CALL MSGCLR
          CALL MSGPOS(2,1)
          CALL MSGCOL(CYAN)
          CALL MSGPUT('NO INFO',7)
        END IF
      ELSE
        CALL MSGCLR
        CALL MSGPOS(2,1)
        CALL MSGCOL(CYAN)
        CALL MSGPUT('NO INFO',7)
      END IF
      CALL LINTYP(ILINE)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE HELP_COLOR
C*******************************************************************************
C GIVES INSTRUCTIONS IN THE MESSAGE AREA ON CHANGING COLOR.
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This menu item changes the CURRENT COLOR.',41)
      CALL MSGPOS(3,1)
```

```
      CALL MSGPUT('To change the current COLOR, click once over box',49)
      CALL MSGPOS(4,1)
      CALL MSGPUT('that is the desired new COLOR. You MUST change COLOR',53)
      CALL MSGPOS(5,1)
      CALL MSGPUT('before creating CIRCLE, FAN or RECTANGLE. Once these',53)
      CALL MSGPOS(6,1)
      CALL MSGPUT('items are selected the COLOR cannot changed.',46)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE HELP_LINTYP
C*******************************************************************************
C GIVES INSTRUCTIONS IN THE MESSAGE AREA ON CHANGING LINE TYPE.
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This menu item changes the current LINE STYLE.',47)
      CALL MSGPOS(3,1)
      CALL MSGPUT('To change the LINE STYLE, click once over box',45)
      CALL MSGPOS(4,1)
      CALL MSGPUT('that is the desired new LINE STYLE. You must change',51)
      CALL MSGPOS(5,1)
      CALL MSGPUT('LINE STYLE before creating CIRCLE, FAN or RECTANGLE.',52)
      CALL MSGPOS(6,1)
      CALL MSGPUT('Once these items are selected, LINE STYLE cannot be change
      1d.',61)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE HELP_EXPICT
C*******************************************************************************
C GIVES INSTURCTIONS IN THE MESSAGE AREA ON HOW TO CHANGE THE CURSOR.
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This is only an example picture to show you some of',51)
      CALL MSGPOS(3,1)
      CALL MSGPUT('objects, colors, line styles and character string',49)
      CALL MSGPOS(4,1)
      CALL MSGPUT('inputs available to you.',24)
      RETURN
```

```
      END
C*************************************************************************
      SUBROUTINE HELP_FILL
C*************************************************************************
C GIVES INSTRUCTIONS IN MESSAGE AREA ON FILLING A POLYGON, CIRCLE, OR FAN.
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This menu item FILLS a particular POLYGON, CIRCLE',49)
      CALL MSGPOS(3,1)
      CALL MSGPUT('or FAN.',7)
      CALL MSGPOS(4,1)
      CALL MSGPUT('To fill, first pick FILL STYLE then click once on',49)
      CALL MSGPOS(5,1)
      CALL MSGPUT('the outter bounds of a POLYGON, CIRCLE, FAN or RECTANGLE.'
      1,57)
      RETURN
      END
C*************************************************************************
      SUBROUTINE HELP_POLYGON
C*************************************************************************
C GIVES INSTRUCTIONS IN THE MESSAGE AREA ON CREATING A POLYGON .
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This menu item lets you create a polygon.',41)
      CALL MSGPOS(3,1)
      CALL MSGPUT('First click over BEGIN, create POLYGON, and last',48)
      CALL MSGPOS(4,1)
      CALL MSGPUT('click once over END to signal polygon completed.',48)
      CALL MSGPOS(6,1)
      CALL MSGPUT('You will need to CONFIRM or CANCEL polygon after it',51)
      CALL MSGPOS(7,1)
      CALL MSGPUT('is created.',11)
      RETURN
      END
C*************************************************************************
      SUBROUTINE HELP_CIRFAN
C*************************************************************************
C GIVES INSTRUCTIONS IN THE MESSAGE AREA ON CREATING A CIRCLE, ARC, OR FAN.
      INCLUDE 'PARA.'
```

```
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('These menu items allow you to create a CIRCLE or a fan.
     1 ',56)
      CALL MSGPOS(3,1)
      CALL MSGPUT('You will need to indicate whether keyboard or pad entry of
     1 CIRCLE or FAN data.',78)
      CALL MSGPOS(4,1)
      CALL MSGPUT('Default is the pad. To create by pad click once in
     1 position for center',71)
      CALL MSGPOS(5,1)
      CALL MSGPUT('of CIRCLE or FAN, once for beginning of CIRCLE or FAN,
     1 and once for ending.',77)
      CALL MSGPOS(6,1)
      CALL MSGPUT('By keyboard, enter radius(r>0), beginning angle(degrees),
     1 ending angle(degrees).',79)
      CALL MSGPOS(7,1)
      CALL MSGPUT('For X,Y either click where you want center of CIRCLE or
     1 FAN or enter X,Y by',75)
      CALL MSGPOS(8,1)
      CALL MSGPUT('Keyboard(0<=X<=1,0<=Y<=1).',27)
      RETURN
      END
C*************************************************************************
      SUBROUTINE HELP_RECT
C*************************************************************************
C GIVES INSTRUCTIONS IN THE MESSAGE AREA ON CREATING A RECTANGLE.
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This menu item lets you create a RECTANGLE.',43)
      CALL MSGPOS(3,1)
      CALL MSGPUT('To create give two opposite corner points of the',48)
      CALL MSGPOS(4,1)
      CALL MSGPUT('RECTANGLE you wish to create.',29)
      RETURN
      END
C*************************************************************************
      SUBROUTINE HELP_DISPLAY
C*************************************************************************
C LETS USER SEE X,Y COORINATES.
```

```
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This menu item lets you DISPLAY the last X and Y',48)
      CALL MSGPOS(3,1)
      CALL MSGPUT('coordinates recorded.',21)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE HELP_TEXT
C*******************************************************************************
C GIVES INSTRUCTIONS IN THE MESSAGE AREA ON ENTERING TEXT THROUGH KEYBOARD.
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This menu lets you enter TEXT through the keyboard.',51)
      CALL MSGPOS(3,1)
      CALL MSGPUT('You have the choice to enter a sub-menu to choose the',53)
      CALL MSGPOS(4,1)
      CALL MSGPUT('character size and positioning of the text string or',52)
      CALL MSGPOS(5,1)
      CALL MSGPUT('you can use the default character size and positioning',
     1 54)
      CALL MSGPOS(6,1)
      CALL MSGPUT('(horizontal). Then enter TEXT through keyboard (return',
     1 54)
      CALL MSGPOS(7,1)
      CALL MSGPUT('is the signal for end of input). Last choose area you',53)
      CALL MSGPOS(8,1)
      CALL MSGPUT('want text to be in with the cursor.',35)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE HELP_ERASE
C*******************************************************************************
C ALLOWS THE USER ERASE A COMPLETE PICTURE. ALL SEGMENTS CREATED ARE GONE.
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This item lets you ERASE the entire working area.',50)
```

```
      CALL MSGPOS(3,1)
      CALL MSGPUT('After this, entire picture you have created is lost.',55)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE HELP_EXIT
C*******************************************************************************
C EXIT FROM THE PROGRAM
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This allows you to EXIT out of graphics mode. After',52)
      CALL MSGPOS(3,1)
      CALL MSGPUT('this you will have the option of saving the picture',52)
      CALL MSGPOS(4,1)
      CALL MSGPUT('and re-entering the program to do another picture.',50)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE HELP_UNDO
C*******************************************************************************
C GIVES INSTRUCTION IN THE MESSAGE AREA ON DELETING A CERTAIN LINE, POLYGON,
C CIRCLE OR FAN.
      INCLUDE 'PARA.'
      CALL MSGCLR
      CALL MSGPOS(2,1)
      CALL MSGCOL(CYAN)
      CALL MSGPUT('This allows you to UNDO a LINE, POLYGON, CIRCLE, ARC,',54)
      CALL MSGPOS(3,1)
      CALL MSGPUT('TEXT, or RECTANGLE you have created. After choosing UNDO,'
      1,57)
      CALL MSGPOS(4,1)
      CALL MSGPUT('click on outter bounds of primitive you wish to get rid of
      1.',60)
      RETURN
      END
C*******************************************************************************
      SUBROUTINE CHOICE_MESSAGES(ILINE,I)
C*******************************************************************************
C ROUTINE TO DISPLAY IN THE MESSAGE AREA CONFIRM CANCEL CHOICES AND KEYBOARD
C INPUT OPTION (DEFAULT IS LOCATOR. HELP IS INCLUDED IN THIS ONE)
      INCLUDE 'PARA.'
```

```
      CALL LINTYP(1)
      CALL MSGCOL(CYAN)
      IF(I.EQ.1) THEN
        CALL MSGPOS(2,60)
        CALL MSGPUT('--------------------',20)
        CALL MSGPOS(3,60)
        CALL MSGPUT('- CONFIRM   CANCEL -',20)
        CALL MSGPOS(4,60)
        CALL MSGPUT('--------------------',20)
      ELSE IF(I.EQ.2) THEN
        CALL MSGPOS(2,55)
        CALL MSGPUT('---------------------------',27)
        CALL MSGPOS(3,55)
        CALL MSGPUT('- KEYBOARD  HELP  TABLET -',27)
        CALL MSGPOS(4,55)
        CALL MSGPUT('---------------------------',27)
      ELSE IF(I.EQ.3) THEN
        CALL MSGPOS(2,70)
        CALL MSGPUT('----------',10)
        CALL MSGPOS(3,70)
        CALL MSGPUT('- CANCEL -',10)
        CALL MSGPOS(4,70)
        CALL MSGPUT('----------',10)
      ELSE IF(I.EQ.4) THEN
        CALL MSGPOS(2,60)
        CALL MSGPUT('------------------------',24)
        CALL MSGPOS(3,60)
        CALL MSGPUT('- SIZE AND POSITIONING -',24)
        CALL MSGPOS(4,60)
        CALL MSGPUT('------------------------',24)
      END IF
      CALL LINTYP(ILINE)
      RETURN
      END
C*****************************************************************************
      SUBROUTINE CHECK_CONFCANC(I)
C*****************************************************************************
C ROUTINE TO CHECK TO SEE IF WHATEVER NEEDS TO BE CONFIRMED IS CONFIRMED OR
C CANCELED SUCH AS ERASE, CIRCLE CORRECT AND ETC. I=3 IS INDICATION NEITHER
C CONFIRM OR CANCEL WAS CHOSEN.
      INCLUDE 'PARA.'
      CALL AWAIT(32767,ID,IN)
      CALL GLOCAT(IL,XL,YL)
```

```
C GET INTEGER VALUES OF X AND Y COORDINATES.
      IX=INT(10.0*XL)
      IY=INT(10.0*YL)
      I=0
      IF(IX.EQ.7 .AND. IY.EQ.1) THEN
        I=YES
      ELSE IF(IX.EQ.8 .AND. IY.EQ.1) THEN
        I=NO
      ELSE
        I=3
      END IF
      CALL MSGCOL(BLUE)
      CALL MSGPOS(4,1)
      IF(I.EQ.YES) THEN
        CALL MSGPUT('REQUEST CONFIRMED',17)
      ELSE
        CALL MSGPUT('REQUEST CANCELED',16)
      END IF
      RETURN
      END
C*******************************************************************************
      SUBROUTINE INPUT_CHOICE(ICHOICE,ILINE,KTIME,XL,YL)
C*******************************************************************************
C THIS ROUTINES ALLOWS THE CHOICE FOR INPUT FOR CIRCLES AND FANS DATA THROUGH
C THE KEYBOARD OR PAD. THERE IS ALSO A HELP OPTION AT THIS POINT. ICHOICE=1
C INDICATES INPUT FROM KEYBOARD, ICHOICE=2 INDICATES HELP, AND ICHOICE=3
C INDICATES THE PAD-DEFAULT.
      ICHOICE=0
      CALL AWAIT(KTIME,ID,IN)
      CALL GLOCAT(IL,XL,YL)
      IX=(100.0*XL)
      IY=(100.*YL)
      IF(IX.GE.69 .AND. IX.LE.74 .AND. IY.EQ.13) THEN
        ICHOICE=1
      ELSE IF(IX.GE.77 .AND. IX.LE.81 .AND. IY.EQ.13) THEN
        CALL HELP_CIRFAN
        ICHOICE=2
      ELSE IF(IX.GE.85 .AND. IX.LE.89 .AND. IY.GE.11 .AND. IY.LE.13) THEN
        ICHOICE=3
      END IF
      RETURN
      END
C*******************************************************************************
```

```
      SUBROUTINE INPUT_KEYBOARD(ILINE,KTIME,IERROR,R,THS,THE)
C*************************************************************************
C ROUTINE TO ACCEPT CIRCLE, ARC, OR FAN INFORMATION FROM KEYBOARD(RADIUS,
C BEGINNING ANGLE AND ENDING ANGLE. THE INPUT DATA IS FIRST CONVERTED TO A
C CHARACTER STRING, THEN CONVERTED TO REAL. A ERROR CHECK IS DONE TO SEE IF
C R=0.
C
C ALSO THE INPUT IS CHECKED TO SEE IF THE CANCEL OPTION IS CHOSEN.
C IF SO THE ERROR FLAG IS SET SO THE CIRCLE OR FAN WILL NOT BE CREATED.
C THE RADIUS IS INPUT FIRST, THEN BEGINNING ANGLE, THEN ENDING ANGLE.
C
C IF USER INPUTS AN INTEGER RADIUS OR ANGLE A DECIMAL IS ADDED TO THE STRING.
C THE CIRCLE AND FAN ROUTINES ACCEPT REAL VALUES ONLY.
      INCLUDE 'PARA.'
C RADIUS IS KEYBOARD INPUT FOR ACTUAL RADUIS.
C B IS INPUT FOR BOTH BEGINNING AND ENDING ANGLE.
      INTEGER RADIUS(8),B(5)
C ST- CHARACTER STRING TO TEMPORARILY STORE CONVERTED BYTE IN.
      CHARACTER*8 ST
      BYTE TEST(4),TEST1(4)
      EQUIVALENCE(TEST,RADIUS(2))
      EQUIVALENCE(TEST1,B(2))
C ENABLE KEYBOARD.
      CALL ECHO(KEYB,1,1)
      CALL ENBDEV(KEYB,1)
C INSTRUCTIONS GIVEN ON HOW INFORMATION SHOULD BE ENTERED THROUGH KEYBOARD.
      CALL INSTRUCTION_MESSAGE(ILINE,16)
      CALL INSTRUCTION_MESSAGE(ILINE,12)
C IPER- FLAG TO SEE IF DECIMAL ENTERED ON INPUT OF DATA. IF IPER NOT SET TO ONE
C THEN DECIMAL IS ADDED TO DATA.
      IPER=0
C DISPLAY CANCEL OPTION. USER MAY CANCEL OPERATION.
      CALL CHOICE_MESSAGES(ILINE,3)
C ACCEPT INPUT (WILL BE EITHER KEYBOARD OR TABLET- ONLY THESE TWO DEVICES
C ENABLED.
      CALL AWAIT(KTIME,ID,IN)
      IF(ID.EQ.KEYB) THEN
        CALL GKEYBD(RADIUS(2),L)    !ACCEPT RADIUS THROUGH KEYBOARD.
        DO N=1,L
          ST(N:N)=CHAR(TEST(N))     !CONVERT FORM BYTE TO CHARACTER STRING.
          IF(ST(N:N).LT.'0' .OR. ST(N:N).GT.'9') THEN
           IF(ST(N:N).NE.'.') THEN
             CALL ERROR_MESSAGE(ILINE,4)   !ERROR IN INPUT. WILL EXIT ROUTINE
```

```
     GOTO 30                    !AT THIS POINT AND DISPLAY ERROR
    END IF              !MESSAGE TO USER.
   END IF
IF(ST(N:N).EQ.'.') IPER=1    !FLAG DECIMAL WAS ENTERED.
END DO
IF(IPER.NE.1) THEN           !DECIMAL POINT NEEDS TO BE ADDED.
  L=L+1
  ST(L:L)='.'
END IF
IPER=0                       !RESET DECIMAL FLAG TO ZERO.
DECODE(L,100,ST) R           !CONVERT CHARACTER RADIUS TO REAL NO.
IF(R.EQ.0.0) THEN               !IF RADIUS = 0 THEN ERROR MESSAGE AND
  CALL ERROR_MESSAGE(ILINE,3)  !EXIT ROUTINE.
  GOTO 30
END IF
CALL AWAIT(KTIME,ID,IN)      !NEXT INPUT FROM USER.
IF(ID.EQ.KEYB) THEN
  CALL GKEYBD(B(2),L)        !BEGINNING ANGLE ENTERED THROUGH KEYBOARD.
  DO N=1,L
    ST(N:N)=CHAR(TEST1(N))
    IF(ST(N:N).LT.'0' .OR. ST(N:N).GT.'9') THEN
      IF(ST(N:N).NE.'.') THEN       !ERROR IN INPUT. DISPLAY ERROR
        CALL ERROR_MESSAGE(ILINE,4) !MESSAGE AND EXIT ROUTINE.
        GOTO 30
      END IF
    END IF
  IF(ST(N:N).EQ.'.') IPER=1   !DECIMAL FLAG SET.
  END DO
  IF(IPER.NE.1) THEN       !DECIMAL FLAG NOT SET. ADD DECIMAL
    L=L+1                  !TO ANGLE.
    ST(L:L)='.'
  END IF
  IPER=0                   !RESET DECIMAL FLAG TO ZERO.
  DECODE(L,110,ST) THS     !CONVERT ANGLE TO REAL NUMBER.
  CALL AWAIT(KTIME,ID,IN) !AWAIT USER INPUT.
  IF(ID.EQ.KEYB) THEN
    CALL GKEYBD(B(2),L)    !ACCEPT ENDING ANGLE THROUGH KEYBOARD.
    DO N=1,L
      ST(N:N)=CHAR(TEST1(N))
      IF(ST(N:N).LT.'0' .OR. ST(N:N).GT.'9') THEN
        IF(ST(N:N).NE.'.') THEN       !ERROR IN INPUT. DISPLAY ERROR
          CALL ERROR_MESSAGE(ILINE,4) !MESSAGE AND EXIT ROUTINE.
          GOTO 30
```

```
          END IF
        END IF
        IF(ST(N:N).EQ.'.') IPER=1     !SET FLAG TO ONE. DECIMAL FOUND.
        END DO
        IF(IPER.NE.1) THEN       !DECIMAL FLAG NOT SET. ADD DECIMAL TO
          L=L+1          !TO STRING.
          ST(L:L)='.'
        END IF
        IPER=0                   !RESET DECIMAL FLAG TO ZERO.
        DECODE(L,110,ST) THE     !CONVERT ANGLE TO REAL NUMBER.
      ELSE
        GOTO 20                  !CANCEL OPTION SELECTED.
      END IF
    ELSE
      GOTO 20                    !CANCEL OPTION SELECTED.
    END IF
  ELSE
    GOTO 20                      !CANCEL OPTION SELECTED.
  END IF
  GOTO 40                        !NORMAL INPUT. SKIP ERROR FLAG BEING SET.
20 CONTINUE
  CALL CONFIRM_MESSAGE(ILINE,8) !DISPLAY OF ERROR MESSAGE.
30 CONTINUE
  IERROR=1                      !ERROR FLAG IS SET.
40 CONTINUE
100 FORMAT(F8.6)
110 FORMAT(F5.2)
  RETURN
  END
```

VITA

Mary Watts Jones was born in Hazard, Kentucky on November 1, 1960. She attended elementary school in Hindman, Kentucky and was graduated from Knott County Central High School in June 1978. The following August she entered Morehead State University in Morehead, Kentucky, and in August 1982 she received a Bachelor of Science degree in Mathematics. In the fall of 1983 she accepted a graduate research assistantship at The University of Tennessee Space Institute. She received a Master of Science Degree in Computer Science in December 1985.

The author is a member of SIAM and Phi Kappa Phi. Ms Jones will be employed by the University of Dayton Research Institute, Dayton, Ohio, after graduation.