IMPLEMENTATION OF EXTENDED GRAPHIC PRIMITIVES

by

MAXINE F. YEE

B. S., University of California, Berkeley, 1962

AN ABSTRACT OF A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1982

Approved by:

Major Professor

TABLE OF CONTENTS

THIS BOOK CONTAINS NUMEROUS PAGES WITH THE ORIGINAL PRINTING BEING SKEWED DIFFERENTLY FROM THE TOP OF THE PAGE TO THE BOTTOM.

THIS IS AS RECEIVED FROM THE CUSTOMER.

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

LIST OF FIGURES

CHAPTER I INTRODUCTION

A. OVERVIEW

This report is last of a sequence of four reports which extend the capabilities of the Chromatics CG series computer system. Background infcrmation concerning the system is given in reports by M. Dillinger and S. Mitchell(Dil80, Mit81).

EGP(Extended Graphic Primitive) is a small computer graphic package which allows the user to construct a selected set of two-dimensional figures and to display them on the Chromatics terminal screen. It is an interactive program designed to extend the existing graphic primitive set. While the primitives such as the vector, the circle, the rectangle and the arc are desirable and in many cases, quite adequate, frequently, the existing set of primitives is too limited to accommodate the various needs of the users. In many situations, a number of additional figures is necessary for the develcpment of flow charts, hierarchical diagrams, or illustration layout. Without the additions, the user must write programs tc create objects which are needed. The time needed to develop, code and debug such a prcgram might be substantial; particularly if the user is un-familiar with the language and/or the system. To overcome this hindrance, the extension of the primitive set is essential.

The goal of this project is:

1) Design and implement a program to generate a selected set of primitives.
2) Test and debug the program.
3) Provide a detailed documentation for future programmers who might like to make modification or further extensions.
4) Provide a short user's guide.

Primitives created in this project are:

1) Vector (figure 1).
2) Arrow (figure 2).
3) Double-headed arrow (figure 3).
4) Rectangle (figure 4).
5) Bubble (figure 5).

B. PAPER ORGANIZATION

The remainder of this report is structured this way: Chapter II is a simple user's guide. It describes the implemented commands, the specification of the different variabls and the calling formats. Chapter III is an overview of the program, with explanation on the internal structure and logic of the various modules.

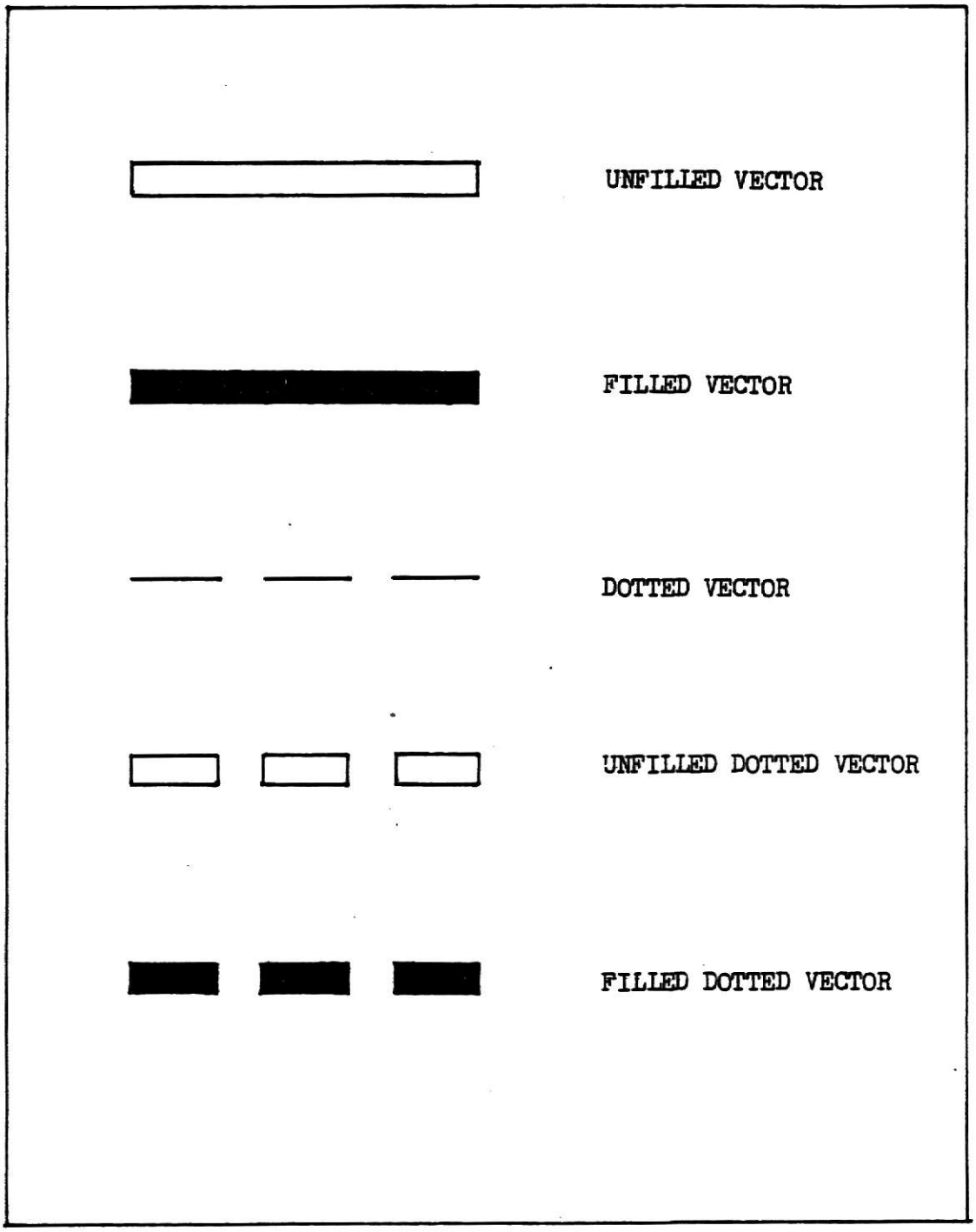


FIGURE 1 VECTOR

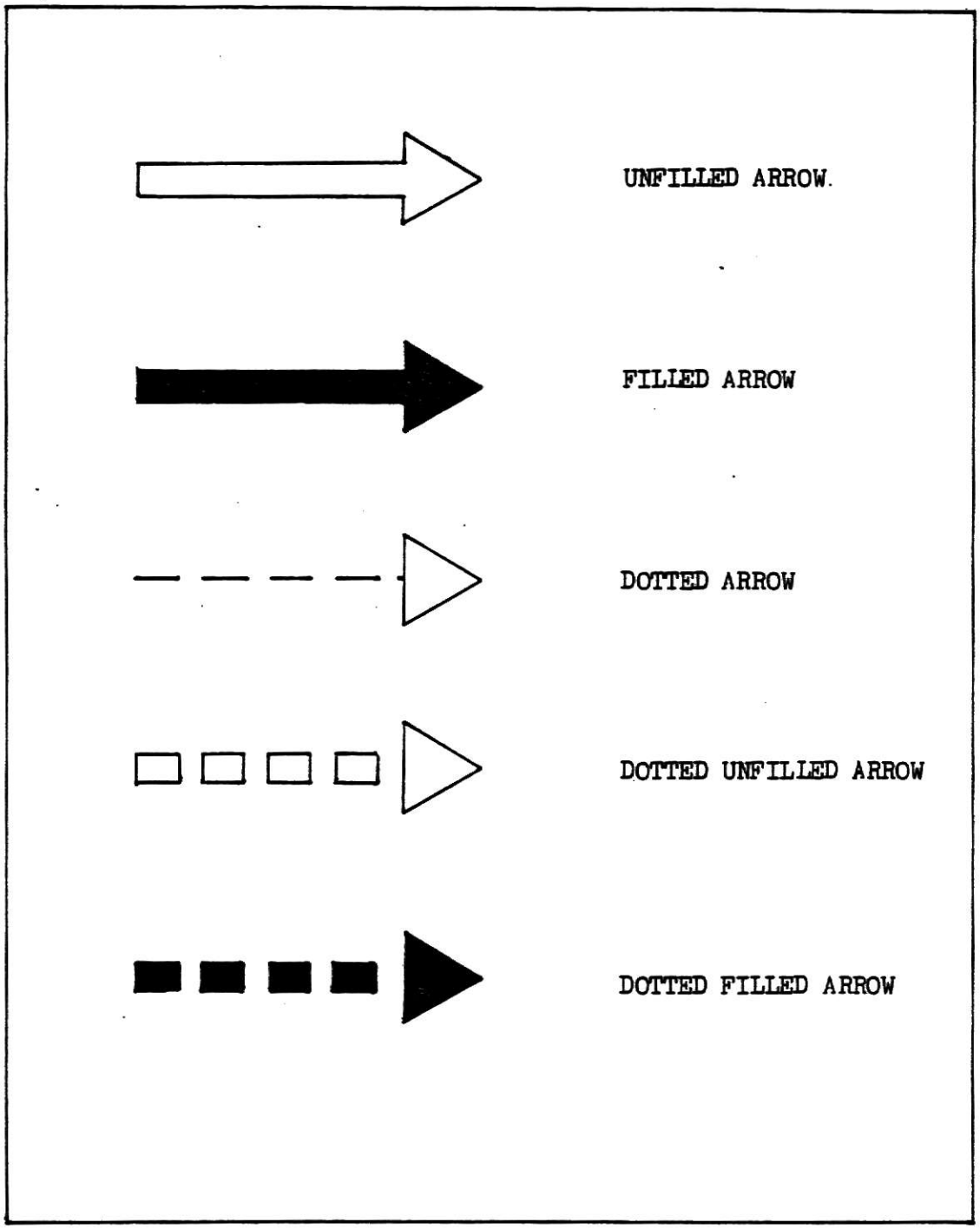


FIGURE 2 ARROW

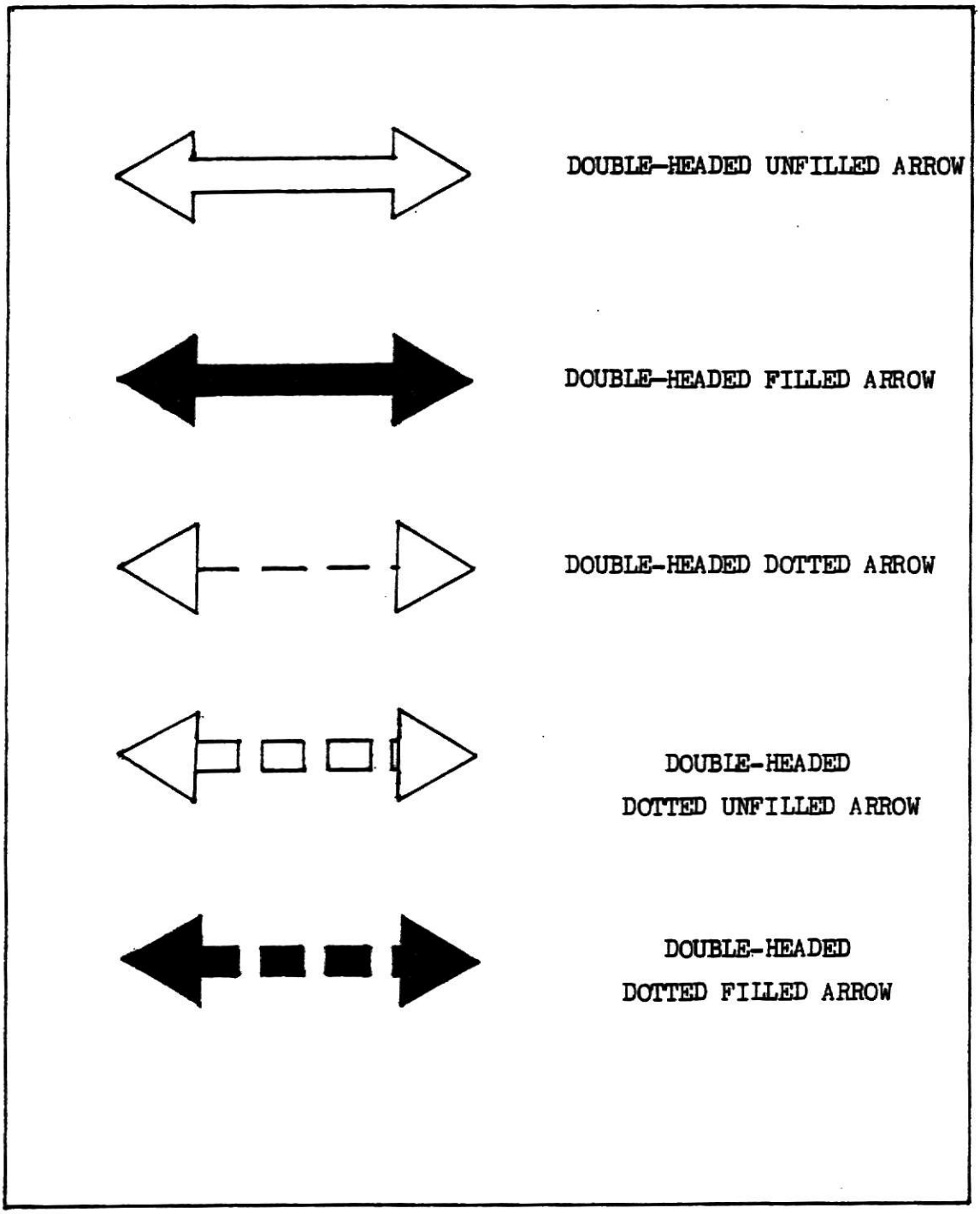


FIGURE 3 DOUBLE-HEADED ARROW

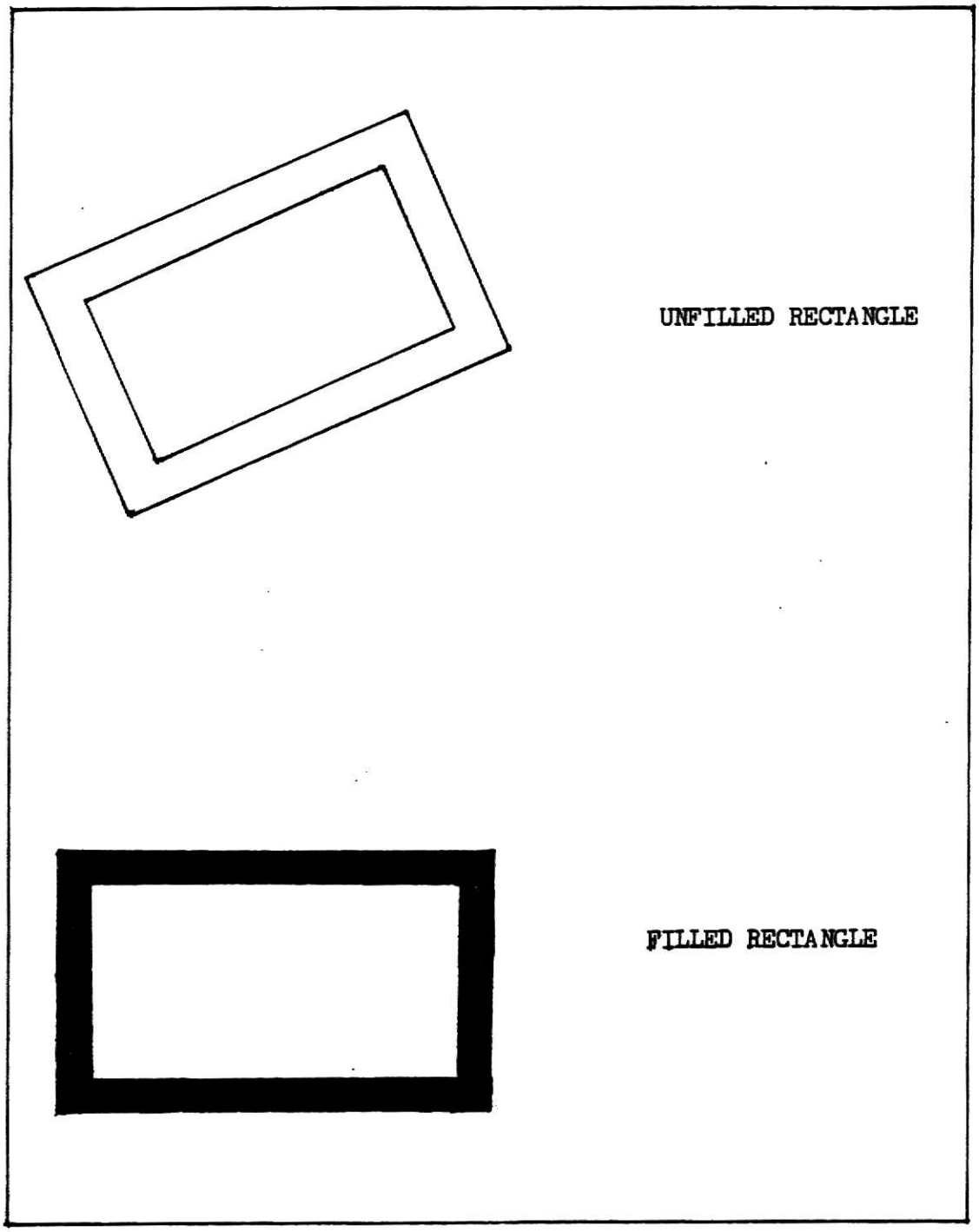
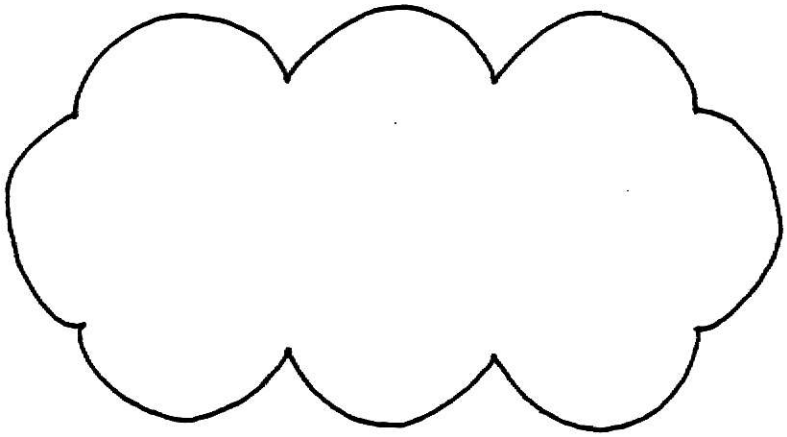
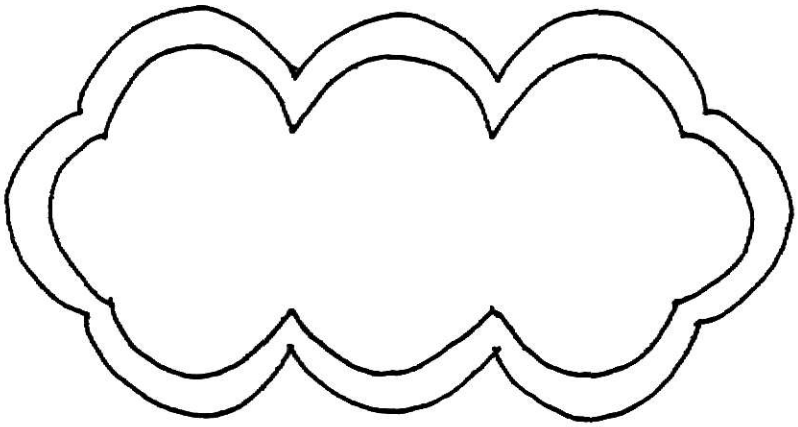


FIGURE 4 RECTANGLE



BUBBLE



UNFILLED BUBBLE

FIGURE 5 BUBBLE

C. EVALUATION

EGP is not a self contained program, nor is it an integrated graphic package by itself. Rather, it provides additional primitives for the progammer to use in conjuction with the existing set. In the course of designing the program, no emphasis was placed on supplying input facilities other than the keyboard. This is too restrictive. The addition of the cursor, bitpad and data file as input devices to the system would be highly desirable. The driver developed by M. Dillinger accomplished some of these objectives (Dil80).

A problem arises with the lack of memory space. The progrm is too large to be loaded and run in the default size of the Basic memory. Therefore , it is segmented into two parts: EGP1 contains codes necessary to draw the vector, the arrow and double-headed arrow while EGP2 contains codes for the generation of rectangle and bubble. In addition to the space limitation, pictures generated on the screen and the create buffer can not be selectively modified. Application of the program reguires that the complete picture be redrawn at each change-an inconvenience , since freguently only a small portion of the picture needs modification. Thus, memory overlays, user interfaces, functions for selective addition, replacement, and deletion should be incorporated into the package. S. Mitchell's project (Mit81) dealt with this aspect of the system.

Pictures generated from this program are two

demensional line drawings only. There are no transformations for scaling, translation or rotation of any object. Features such as these would be an excellent addition.

Finally, after working with Basic, its weakness have become more apparent to me. Althcugh I have tried to use structured programming techniques and to write most of the routines in small functional modules, the readability of the program is greatly hampered because of all the GOTOS and GOSUBS statements. The lack of parameter passing in a subroutine call makes it necessary to write duplicate plotting routines to acccmplish similar tasks. The recognition of only the first two characters as an identifier is annoying because an identifier can not be named more distinctly in the context of its function. Control structure is awkward. It must be emphasized that the price to pay for the simplicity in using Basic as a language is pccr program structuring. I would like to see this program translated into another language, perhaps Pascal, which I consider to have outstanding clarity, extensive contrcl structures and user defined data types. It is a very desirable language for an implementation.

On the positive side, this project has accomplished what it intended to do. The program has been tested extensively on all the different branches and proven to work correctly for scme selected data sets.

CHAPTER II USER'S GUIDE

This chapter is intended to serve as a user's guide for the EGP program. The reader is assumed to know BASIC and to be familiar with the operations of the Chromatics CG terminal.

To recapitulate, the EGP program allows the user to generate and to display straight-line drawings on the Chromatics CRT screen. EGP commands are, in reality, calls to various routines designed to perform graphic functions. The following section describes the implemented commands and their specificaticns. The final section is devoted to a terminal session, which serves as an illustration of the uses of the various commands.

1. IMPLEMENTED COMMANDS

Two sets of variables, namely, the state variables and the input variables are common for all the implemented commands. To avoid repetition, these variables and their specifications will be discussed first.

State Variables

BC Attribute Value	Background color (see color table, figure 6). Integer. 1 to 7.
FC Attribute Value	Foreground color. Integer. 1 to 7. Note: BC can be set to any value in the range, but FC must be chosen with care; some values will not work.

COLOR CODE	COLOR
0	BLACK
1	BLUE
2	GREEN
3	CYAN
4	RED
5	MAGENTA
6	YELLOW
7	WHITE

FIGURE 6 COLOR TABLE

DO$	Dotted, string variable.
Attribute	Character.
Value	D for dotted mode, XD for undotted mode.
FI$	String variable.
Attribute	Character.
Value	F or XF. note: F for filled mode, XF for unfilled mode.
BI$	String variable.
Attribute	Character.
Value	B or XB. note: B for blink mode, XB for unblink mode.

Input Variables

(XS,YS)	Starting coordinate of an object.
Attribute	Integer, single precision.
Value	Depends on user's input, limited by the number of dots on the CRT screen.
(XN,YN)	Ending coordinate of an objcet.
Attribute	Integer, single precision.
Value	Depends on the user's input, limited by the number of dots on the CRT screen.
W	Width of a line segment.
Attribute	Integer, single precision.
Value	2 to 20 for VEC; 1 to 6 for other commands.
HT	Height of the rectangle and bubble.
Attribute	Integer, single precision.
Value	Depends on user's input, limited by the number of dots on the CRT screen.

THE VEC COMMAND

Purpose

Allows a general vector (with all options) to be drawn on the CRT screen.

State Variables

Assume the same attributes and values as discussed at the beginning of this chapter.

Input Variables

Assume all of the characteristics of the input variables discussed at the beginning of this chapter.

Local Variable

X, Y, L, L1	Simple variables.
Attribute	Single precision real numbers.
Value	Depends on the result of calculation.
X(I+1) Y(I+1) X(I+2) Y(I+2) X(J+1) Y(J+1)	Array variables, represent the Coordinate points of the vector.
Attribute	Single precision real numbers.
Value	Depends on the result of calculation.
XM#, YM#	Simple variables.
Attribute	Double precision real numbers.
Value	Depends on the result of calculation.
CS#, SI#	Simple variables.
Attribute	Double precision real numbers.
Value	Depends on the result of calculation.
D	Simple variable, represents the accumulated length of a given line segment.
Attribute	Integer, initial value: 12.
Value	Depends on the result of calculation.
SEG	Simple variable, represents the length of each segment of a dotted line.

Attribute	Integer.
Value	12.
SP	Simple variable, represents the spacing between the dotted line.
Attribute	Integer.
Value	6.
K	Simple variable, represents the terminal point used in the For-Next loop.
Attribute	Integer.
Value	Depends on the result of calculation.

Parameter for Plotting Routine

(XS,YS)
(XN,YN)
X(I+1)
Y(I+1)
X(I+2)
Y(I+2)

specifications of the above variables have been discussed.

EXAMPLE OF CALLING SEQUENCE

```
      VEC   RETURN        Invoke the vector routine.
      2,4   RETURN        Set FG and BG colors.
   D,F,XB   RETURN        Set options.
   50,300   RETURN        Starting coordinate.
145,450,6   RETURN        Ending coord. and width.
```

The VEC command places the window in the wide vector plot submode. After typing in the command, the state variables must be set according to the values described. Starting and ending coordinates must be within the window boundaries. Otherwise, figures produced will be unpredictable. To change state variables or tc end the VEC command, enter 999 as the first coordinate point.

THE ARR COMMAND

Purpose

Allows a general arrow (with all options) to be drawn on the CRT screen.

State Variables

Assume the same attributes and values as described at the beginning of this chapter.

Input Variables

Assume all of the characteristics of the input variables as discussed at the beginning of this chapter.

Local Variables

X, Y, L2	Simple variables.
Attribute	Single precision reals.
Value	Vary, depending upon the results from calculations.
SI#, CI#	Simple variables, represent sine and cosine of an angle.
Attribute	Double precision real numbers.
Value	Vary, depending upon the result from calculations.
AH$	String variable.
Attribute	Character (3).
Value	FIX or VAR.
X(P) Y(P) X(P+I) Y(P+I)	Where 0<=I<=6, P=2. These are array variabls; they represent the coordinate points of the arrow.
Attribute	Single precision real numbers.
Value	Vary, depending on the results of different calculations.

Parameter for Plotting Subroutine

XS
YS

XN
YN
X(P)
Y(P)
X(P+I) Where 0<=I<=6, P=2.
Y(P+I)

Specifications of these variables have been discussed.

EXAMPLE OF CALLING SEQUENCE

```
      ARR    RETURN        Invoke the arrow routine.
      3,6    RETURN        Set BG and FG colors.
   D,F,XB    RETURN        Set options.
  115,170    RETURN        Starting coordinate.
215,170,4    RETURN        Ending coord. and width.
      VAR    RETURN        Set size for arrow head.
```

The ARR command places the window in the plot arrow submode. Restrictions about the state variables and coordinate inputs are the same as that of the VEC command. The ending of command or the changing of options can be accomplished by entering 999.

THE DAR COMMAND

Purpose

Allows a general double-headed arrow (with all options) to be drawn on the CRT screen.

State Variables

Assume all of the characteristics of the state variables as discussed at the beginning of this section.

Input Variables

Assume all of the attributes and values of the input variables as discussed at the beginning of this section.

Local Variables

X, Y, L3	Simple variables.
Attribute	Single precision real numbers.
Value	Depending on the results from different calculations.
SI#, CI#	Represent sine and cosine of an angle.
Attribute	Double precision real numbers.
Value	Depending on the results from different calculations.
AH$	String variable.
Attribute	Character (3).
Value	FIX or VAR.
W1	Simple variable, represents the width of the tail of an arrow.
Attribute	Integer.
Value	W/2.
X(P) Y(P) X(P+I) Y(P+I)	Where 0<I<=9, P=2. These variables represent the coordinates of the double headed arrow.
Attribute	Single precision real numbers.
Value	Vary, depending upon the results of

calculations.

Parameters for plctting routine

XS
YS
XN
YN
XM#
YM#
X(P)
Y(P)
X(P+I)
Y(P+I) Where 0<I<=9, P=2.

Specifications of the above variables have been discussed. Calling sequence, status of the window and restrictions are the same as that of the ARR command.

THE REC COMMAND

Purpose

Allows the generation of rectangles in various positions with specified thickness of the line segment on the CRT screen.

State Variables

Assume the same attributes and values as discussed at the beginning of this section with this exception: The value of the string variable DC$ can only be XD.

Input Variables

Assume all of the characteristics of the input variables as discussed at the beginning of this chapter.

Local Variables

X, Y, L	Simple variables.
Attribute	Single precision real numbers.
Value	Depends upon the results of calculations.

X(I)	
Y(I)	
X(I+J)	
Y(I+J)	Where 0<J<=7, I=1.
Attribute	Single precision real numbers.
Value	Depends upon the results of calculations.

Parameters for Plotting Subroutine

XS	
YS	
XN	
YN	
X(I+J)	Where 0<J<=7, I=1.
Y(I+J)	

Specifications of these variables have been discussed.

EXAMPLE OF CALLING SEQUENCE

```
    REC     RETURN          Invoke the rec. routine.
    2,5     RETURN          Set BG and FG colors.
XD,F,XB     RETURN          Set options.
 100,50     RETURN          Starting coordinate.
145,400     RETURN          Ending coordinate.
   30,6     RETURN          Height and width of rec.
```

The REC command places the window in the rectangle plot submode. Rectangles in various positions will be generated as the proper input data is received. Termination of the routine and changing of options can be accomplished by typing in 999.

THE BUB COMMAND

Purpose

Allows the generation of bubble on the CRT screen.

State Variables

Assume all of the characteristics of the state variables as discussed at the beginning of this chapter.

Input Variables

Assume all of the characteristics of the input variables as discussed at the beginning of this chpter.

Local Variables

PI	Simple Variable.
Attribute	Single precision real number.
Value	3.1416.
CV	Simple Variable.
Attribute	Single precision real number.
Value	Depending upon the result of calculation.
A,B,L,R,R1,R2	Simple Variables.
Attribute	Single precision real umbers.
Value	Depending upon the result of calculation.
X(I) X(I+1) X(I+2) X(I+3)	Array variabls
Attribute	Single precision real numbers.
Value	Depending upon the result of calculation.
Y(I) Y(I+1) Y(I+2) Y(I+3)	Array variables.
Attribute	Single precision real numbers.
Value	Depending upon the result of calculation.
X(I),Y(I),Y(J)	Simple variables, 0<=I<=4, 0<=J<=4.
Attribute	Single precision real numbers.
Value	Depending upon results of calculation.

Z,Z(I)	Simple Variables, 0<I<=9.
Attribute	Single precision real numbers.
Value	Depending upon results of calculation.
T,T(I)	Simple variables, 0<I<=3.
Attribute	Single precision real numbers.
Value	Depending upon results of calculation.

Parameter for Plotting Subroutine

X(I)	Where 0<I<=4.
Y(J)	Where 0<J<=4.
R(I)	Where 0<I<=2.
Z(I)	Where 0<I<=9.

Specification of the above variables have been discussed.

EXAMPLE OF CALLING SEQUENCE

```
      BUB     RETURN      Invoke the bubble routine.
      2,7     RETURN      Set FG and BG colors.
XD,XF,XB      RETURN      Set options.
  300,250     RETURN      Starting coordinate.
  400,250     RETURN      Ending coordinate.
    80,4      RETURN      Height of bubble and width.
```

The BUB command places the window in the bubble plot submode. Bubbles generated are in horizonal position only; they can not be filled. Restrictions on the window boundaries and changing of commands are the same as the other routines.

2. SAMPLE TERMINAL SESSION

Insert disk into the disk drive until it clinks; then push the latch down to lock the disk in place; turn on the Chromatics unit and the disk drive unit. The system is now ready for the user to issue commands. As mentioned earlier, the EGP program is too large to be loaded and run even in its extended memory size. Therefore, it is divided into two sections; EGP1 includes all the necessary routines to generate vectors, arrows and double-headed arrows. EGP2 contains other routines for drawing rectangles and bubbles. Assuming that the user is interested in drawing arrows, the following interaction between him and the machine results in arrows being drawn on the screen.

```
   user:  press RESET; press BASIC

machine:  MEMORY SIZE?

   user:  type in 49100; press RETURN

machine:  CHROMATICS DISK BASIC VER 3.0
          COPYRIGHT (C) 1978 BY MICROSOFT
          28243 BYTES FREE
          OK

   user:  type DOS LOAD EGP1; press RETURN

machine:  OK

   user:  type RUN; press RETURN

machine:  TYPE IN THE CCMMAND

   user:  type ARR; press RETURN

machine:  TYPE IN THE BACKGROUND AND
          FOREGROUND COLORS.
```

```
    user:  type 2,4; press RETURN

 machine:  TYPE IN THE ATTRIBUTES
           FOR THE OBJECT

    user:  type D,F,XB; press RETURN

*machine:  TYPE IN FIRST COORD FOR YOUR OBJECT

   *user:  type 50,300; press RETURN

*machine:  TYPE IN SECOND COORD AND WIDTH
           FOR YOUR OBJECT

   *user:  type 145,450,4; press RETURN

*machine:  TYPE IN VAR FOR VARIABLE ARROW HEAD
           OR FIX FOR FIXED HEAD

   *user:  FIX
```

Almost instantaneously, the arrow is constructecd. When it is finished, the machine will ask for more coordinate inputs. The entire process cycles through the steps marked with (*) until the user types in 999 as the first coordinate. At this point, the user can either issue a command to draw other figures or type END to terminate the program.

CHAPTER III DOCUMENTATION

1. OVERVIEW OF THE PROGRAM

Providing a more detailed explanation of the software, this chapter discusses the general organization of the EGP program, its variables, subroutines, equations and inter-relationship between modules. Figure 7 shows the hierarchical structure of the code.

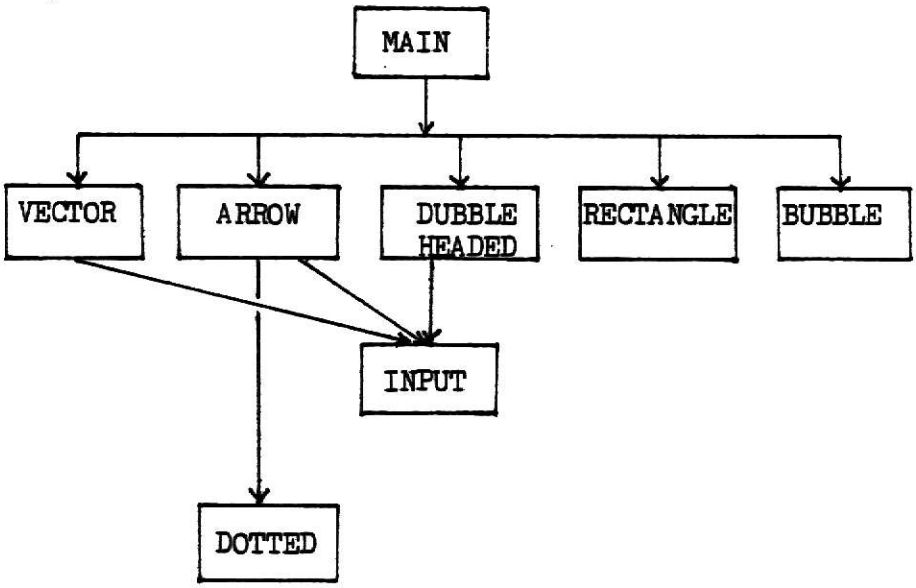


FIGURE 7 HIERARCHICAL CHART

As stated earlier, the primary function of EGP is to generate a selected set of figures to serve as primitives

for the users. To accomplish this task, the program must:

1) Be able to accept user's input commands, and
2) Be able to perform the action specified by these commands.

Achieving these two objectives is a main program subordinated by a set of subroutines. These subroutines are internal to the main program and can be divided into three functional groups. These groups and their constituent routines are listed below.

Primitive Construction Routines

1) Vector
2) Arrow
3) Double-Headed Arrow
4) Rectangle
5) Bubble

Utility Routines

1) Input Routine
2) Set Environment Routine
3) Complex Filled Routine
4) Cancel Plot Mode Routine
5) End Routine

Plotting Routines

1) Plot Vector Routines

2) Plot Arrow Routines

3) Plot Double-Headed Arrow Routine

4) Plot Rectangle Routine

5) Plot Bubble Routine

The Primitive Construction Routines are the most important set of routines in this project. They are the building blocks of the entire system. Their tasks include handling the programming logic, evaluating equations to generate coordinates, testing for input data validity, and calling other subroutines to perform their assignments. These routines are accessed by the user during execution of the EGP program. They are activated, not in the traditional sense of subrortine or procedure calls, but by typing in an appropriate command from the keyboard. Formats of these commands and their attributes were discussed in Chapter II.

The Utility Routines, whose major function is to obtain input data from the user and to set the necessary environments for the proper operation of the program, are transparent to the users. This group of routines is invoked by the primitive construction routines and plotting routines when ever needed.

The End Routine consists of a single statement whose main responsibility is to terminate the EGP program and return control to the command level of CG Basic.

Each of the Plotting Routines is designed to serve a specific primitive construction routine. In response to its calling procedure, they plot the indicated coordinates to create the requsted figure on the screen.

2. DESCRIPTION OF VARIABLES AND THEIR USAGE

Basic simplifies many considerations of data representation due to the inherent simplicity of the language and its limited repertoire of data structures. There exists only two types of variables in Basic, simple variables and array variables; The later being restricted to one or two dimensional arrays. Basic assumes an implicit definition that all arrays contain elements whose subscripts vary from 0 to 10 by default, or from 0 to the upper bound which is declared in a dimension statement in the program. Simple variables do not need to be declared in any statement. Furthermore, there is no distinction between global and local variables. The concept of scope rule governing a particular variable does not apply in this language either. A variable name may be any length, but any alphanumeric character after the first two are ignored. Before a variable is assigned a value, its value is assumed to be zero. One more peculiar element of this language needs to be mentioned. In other languages, a variable name is mapped to only one variable in the entire program. Therefore, for a given variable name, there can only be one set of attributes associated with it. In Basic, however,

the same name may be used to identify both a simple variable and an array structure, and these two data types are not related. With this brief introduction, we shall proceed to describe the variable names, their attriutes and usage.

2.1 Array X and Y

X and Y are two array variables declared in the main routine; each contains 20 elements. Because the exact number of elements to be stored in these structures varies in different routines, and because Basic requires a numeric constant to be specified in the dimension statement for storage allocation purposes, an arbitary number, which represents the maximum number of elements expected, is assigned to each cf these arrays. In many cases, excessive storage locations may result due to this method of assignment. These arrays are used as temporaries to hold the computed coordinate values, which are necessary for the construction of a specific figure. All of the construction routines and plotting routines reference these two arrays; their attributes are in the form of real numbers.

2.2 Variables BC and FC

BC and FC are variables which hold the numeric color codes for background and foreground colors respectively (see color table, figure 6). These color codes are used by the Complex Filled Routines for coloring an object. The programmer inputs two integers, ranging from 0 to 8, to

designate his choice of hue. Care must be exercised to choose the correct color combinations; otherwise, the entire screen may be filled or the program may be BOMBED. For example, the background color can be set to any color except black; the foreground color, however, must contain a color of the border of the object or some composite color that includes it. For a more detailed discussion on this issue, refer to Complex Boundary Fill-Patterns of the Chromatics Operator's Manual (Chr 78b).

2.3 Variables AH$, DO$, FI$, and BI$

AH$, DO$, FI$, and BL$ are a set of string variables whose stored values dictate the appearance of the generated object. For example, if DO$ contains the value D and the object currently being constructed is an arrow, then the final product prcjected on the screen will be a dotted arrow.

The programmer designates his choice of selection by entering the proper values for these string variables through an input statement in the main program. Subsequently, the primitive construction routines use the values of these variabls in a conditional branch instruction as a mechanism to direct the flow of execution.

The attributes of these variables vary from 1 to 3 characters in length.

2.4 Variables XS, YS, XN, YN, W and HT

In order to construct a figure, the program needs coordinate specifications for the object's dimensions. To achieve this, a set of variabls, XS, YS, XN, YN, W and HT is formed. The values of these variables are supplied by the programmer through an input statement of the Input Routine. The attributes for these variables are integers.

2.5 Variable SQ

Because the width and height of the CRT display area are not represented in the same ratio in terms of the X dot distance and the Y dot distance, the object generated by the EGP program appears distorted on the screen. For example, if the position of a rectangle is not plotted vertically or horizonally, it resembles a parallelogram. To correct this deficiency, the variable SQ is introduced. It is used as a scale factor fcr converting the cartesian coordinates to screen coordinates so that objects projected on the display screen would retain their correct perspective. The value of SQ is the square root of integer 2.

2.6 Other Variables

The remaining variables are the results of calculations by the Primitive Construction Routines. L, L1, L2 represent the distances between two given points of a given line segment; XM#, YM# store the values of the midpoints of an object; SI#, CS# contain the values of sine and cosine of an angle. The attributes of these variables are integers, of

single or double precision.

Three other variables, D, SEG and SP are used in the construction of dotted lines; they assume their values through an assignment statement. SEG represents the assigned length of each segment of a dotted line; D is the accumulated distances between the starting point of a given line to the point which is currently being generated; SP represents the spacing between the dotted segments.

Variables used for the Bubble Routine include the following:

PI designates the ratio of the circumference of a circle to its diameter, which is 3.1416. CV is used as a constant to convert the angle from radians to degrees. R, R1, R2 are the radii of the arcs. L and H are the given length and height of the bubble; W specifies whether the figure is a single bubble or a bubble inscribed within another one. The XI's and YI's are the X and Y coordinates; the TI's and ZI's are used to store the various angles. A is the distance between the outer bubble and the inner bubble (if W>1). B represents one half the distance of the chord for a given upper or lower arc. A and B have the following relationships with respect to the length and height of the bubble:

$$A=H/10$$

$$B=(L-4A)/6$$

$$L=4A+6B$$

The decision to give A and B these ratios is arbitrary; other proportions would have worked equally well.

3. DESCRIPTION OF ROUTINES

In contrast to the common convention of high level programming languages, a subroutine in Basic posses no name, no parameters and no actual arguments. Basic subroutines are invoked and terminated through the use of GOSUB and RETURN statements. The statement number immediately following the key word GOSUB indicates the entry point of the called routine. Conforming to this peculiar protocol of procedure invocation, each subroutine in this program is constucted without a parameter list and invoked without any argument. A detaild description of each subroutine follows.

3.1 Main Routine-EGP

This program is organized as a combination of many subroutines. The division of tasks between them is such that each routine is responsible for a specific type of assignment; for example, to generate a set of coordinates or to plot a certain configuration. Consistent with this delegation of labcr, the driver, a small section of the code (statements 10 to 180), steers the sequence of operations by relinquishing control of processing to the proper unit. In conjunction with this major duty, the driver also conducts the following activities:

```
CYCLE FOREVER
     INPUT COMMAND
     INPUT BACKGROUND AND FOREGROUND COLORS
     INPUT ATTRIBUTES FOR THE OBJECT
          IF COMMAND = "VEC" THEN DO ROUTINES FOR
                       WIDE VECTOR
          IF COMMAND = "ARR" THEN DO ROUTINES FOR
                       ARROW
          IF COMMAND = "DAR" THEN DO ROUTINES FOR
                       DOUBLE-HEADED ARROW
          IF COMMAND = "REC" THEN DO ROUTINES
                       FOR RECTANGLE
          IF COMMAND = "BUB" THEN DO ROUTINES FOR
                       BUBBLE
          IF COMMAND = "END" THEN RETURN TO CG
                       BASIC
END CYCLE
```

FIGURE 8 PSEUDOCODE FOR DRIVER

```
CYCLE FOREVER
      INPUT STARTING COORDINATE
      INPUT ENDING COORDINATE
      INPUT WIDTH OF VECTOR
      WHILE STARTING COORDINATE NOT EQUAL TO 999
            DO
                  IF DOTTED THEN DO ROUTINES FOR
                     DOTTED VECTOR
                  IF UNDOTTED THEN DO ROUTINES FOR
                     UNDOTTED VECTOR
            END
      ENDWILE
END CYCLE
```

FIGURE 9 PSEUDOCODE FOR VECTOR

```
CYCLE FOREVER
      INPUT STARTING COORDINATE
      INPUT ENDING COORDINATE
      INPUT WIDTH OF ARROW
      WHILE STARTING COORDINATE NOT EQUAL TO 999
            DO
                  IF DOTTED ARROW AND HEAD = FIXED
                     THEN DO FIXED-HEAD DOTTED
                          ARROW ROUTINES
                     ELSE DO FIXED-HEAD UNDOTTED
                          ARROW ROUTINES
                  IF DOTTED ARROW AND HEAD = VARIABLE
                     THEN DO VARIABLE-HEAD DOTTED
                          ARROW ROUTINES
                     ELSE DO VARIABLE-HEAD UNDOTTED
                          ARROW ROUTINES
             END
       ENDWHILE
 ENDCYCLE
```

FIGURE10 PSEUDOCODE FOR ARROW

```
CYCLE FOREVER
      INPUT STARTING COORDINATE
      INPUT ENDING COORDINATE
      INPUT HEIGHT AND WIDTH
      WHILE STARTING COORDINATE NOT EQUAL TO 999
            DO
                  GENERATE NECESSARY COORDINATE POINTS
                  PLOT THE RECTANGLE
                  IF FILL THEN DO ROUTINES FOR FILL
            END
      ENDWHILE
END CYCLE
```

FIGURE 11 PSEUDOCODE FOR RECTANGLE

```
CYCLE FOREVER
      INPUT STARTING COORDINATE
      INPUT ENDING COORDINATE
      INPUT HEIGHT AND WIDTH
      WHILE STARTING COORDINATE NOT EQUAL TO 999
            DO
                  GENERATE COORDINATES FOR ARCS
                  GENERATE DEGREES FOR ANGLES
                  GENERATE RADII FOR CIRCLES
                  GENERATE CENTERS FOR CIRCLES
                  CALL BUILT-IN ARC FUNCTION FOR PLOTTING
            END
      ENDWHILE
END CYCLE
```

FIGURE 12 PSEUDOCODE FOR BUBBLE

1) Sets up window boundaries for character mode and plot mode operations.
2) Allocates storage for the array variables.
3) Accepts input commands.
4) Accepts values for foreground and background colors.
5) Accepts attributes for the different commands.
6) Prints a short message if an incorrect command is issued.
7) Terminates the program and returns to the command level of CG Basic once the user is finished.

3.2 Primitive Construction Routines

All of the primitive construction routines are structured in the same manner; their primary function is similar, and they pursue a basic sequence of subroutine invocations and exits to accomplish their goals. These sequences are as follows:

1) The Input Routine is called to obtain the necessary data.
2) The desired Primitive Construction Routine is invoked to generate a set of coordinates for a specific object.
3) The Set Environment Routine is activated to establish the proper operating environment.
4) The specific Plotting Routine is called to draw the

requested figure.

5) The Complex Filled Routine is initiated if the requested figure is a filled object.

6) The Cancel Plot Mode Routine is invoked to put the system back to character mode.

7) The process cycles through steps 1 to 6 until the constant 999 is detected from the input data, in which case the routine returns to its caller for more commands.

Before describing each of the primitive construction routines, it is appropriate at this point to discuss all of the supporting components.

3.3 Utility Routines

3.3.1 The Input Routine

The duty of the Input Routine, as the name indicates, is to obtain input data of the starting and ending points of an object. The desired width and the height of an object are also expressed through this routine. This information is typed into the computer by the user as the program is executing, thus producing objects with various orgins and dimensions from each run.

3.3.2 The Set Environment Routine

This routine is responsible for setting the foreground and background colors of an object. It also enables the

plot mode and vector submode functions to prepare the window for plot mode processing. Upon finishing the above tasks, it returns control to the calling routine.

3.3.3 The Complex Filled Routine

The requirement for filling a polygon is that the ploygonal area must be bounded and completely closed; the cursor must be centered inside the polygon, and the colors must be chosen correctly. To fulfil these demands, this routine is written to ensure that the cursor is placed in the center of the object; it then activates the foreground and background lights and sets the corresponding colors. Finally, it issues the command to fill the object.

3.3.4 Cancel Plot Mode Routine

Consisting of only three statements, this tiny section of code puts the window back in character mode and sends the cursor to home position. The system is ready to receive another set of input data for another round of execution.

3.4 Plotting Routines

Although each of the plotting routines creates a unique figure on the screen, the basic scenario implemented in them is similar. They all perform an output function, use the same plot command, and share the same array variables. Based on the values stored in the X and Y array elements, these routines retrieve the values in succession and use

them as coordinate points for the construction of the individual configuration. Upon completing the drawing assignment, these routines will invoke the complex fill module to color the object.

3.5 Primitive Construction Routine

3.5.1 The Vector Routine

The primary function of the Vector Subroutine is to construct different types of wide vectors, be it dotted or undotted, filled or unfilled, depending upon the programmer's specifications. Through the input procedure, we were able to obtain the starting coordinate (XS, YS), the ending coordinate (XN, YN), and the width (W) of the vector. Two more points, (X(I+2),Y(I+2)) which are essential for the construction of the prescribed vector are missing, see figure below:

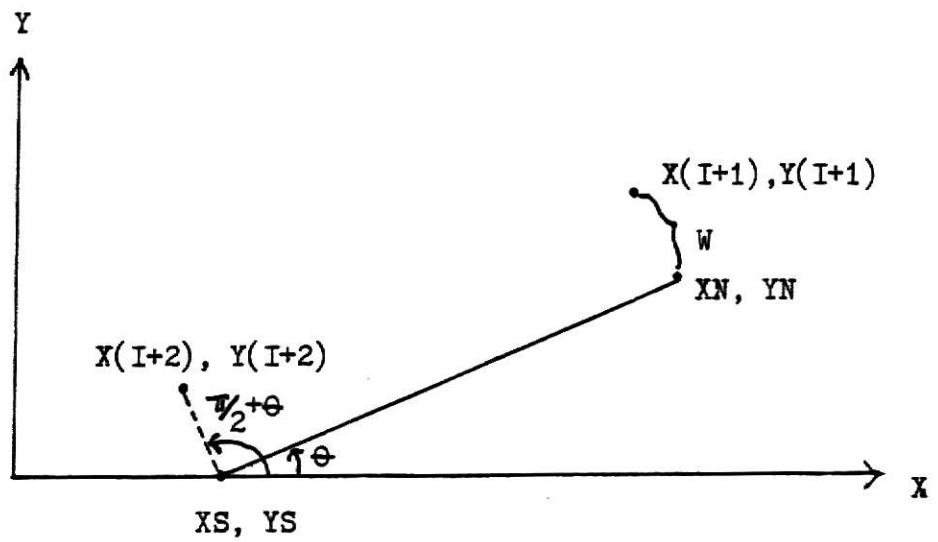


FIGURE 13

We can obtain these points from the following equations:

1) $X(I+1)=XN+W*COS(\pi/2+\theta)$

2) $Y(I+1)=YN+W*SIN(\pi/2+\theta)$

3) $X(I+2)=XS+W*COS(\pi/2+\theta)$

4) $Y(I+2)=YS+W*SIN(\pi/2+\theta)$

From trigonometry identities, we have:

5) $COS(\pi/2+\theta)=COS\,\pi/2\ COS\theta-SIN\,\pi/2\ SIN\theta=-SIN\theta$

6) $SIN(\pi/2+\theta)=SIN\,\pi/2\ COS\theta+COS\,\pi/2\ SIN\theta=COS\theta$ and

7) $$COS\theta=\frac{XN-XS}{\sqrt{(XN-XS)^2+(YN-YS)^2}}\ ,$$

8) $$SIN\theta=\frac{YN-YS}{\sqrt{(XN-XS)^2+(YN-YS)^2}}\ .$$

By setting the denominator of equations 7) and 8) to L, the original set of equations become:

1') $$X(I+1)=XN-W\left(\frac{YN-YS}{L}\right)$$

$$2') \quad Y(I+1) = YN + W\left(\frac{XN-XS}{L}\right)$$

$$3') \quad X(I+2) = XS - W\left(\frac{YN-YS}{L}\right)$$

$$4') \quad Y(I+2) = YS + W\left(\frac{XN-XS}{L}\right)$$

Using the equations in their final forms, the Vector Routine proceeds to calculate the needed coordinates. Having finished, it activates the Plot Routine to do the construction before returning to its caller.

3.5.2 Dotted Vector Routine

This subroutine is called by the Vector Routine. The input data required is transmitted from the input module. Before any calculation is done, a test is made to verify that the given line segment is of sufficient length with which to work. Otherwise, a new data set is requested. To plot a dotted line, we need a collection of points which is colinear to the given line segment. We can obtain these points by evaluating two simple equations repeatedly in a for-next-loop. Each iteration of the loop calculates a new coordinate point. Upon its exit, all of the coordinate points will have been computed; control then returns to the

calling routine. The two equations mentioned above are:

1) $$X(J+1)=XB+D*\left(\frac{XN-XS}{L1}\right)$$

2) $$Y(J+1)=YB+D*\left(\frac{YN-YS}{L1}\right)$$

Where

X(J+1), Y(J+1)	is the coordinate of a particular point.
(XB, YB)	is the starting point of a given line segment.
D	is the distance from the starting point to the current point (X(J+1), Y(J+1)).
$\frac{XN-XS}{L1}$	represents the consine of an angle.
$\frac{YN-YS}{L1}$	represents the sine of an angle.

3.5.3 Arrow and Double-Headed Arrow Routines

It is reasonable to describe simultaneously the Arrow and Double-Headed Arrow routines in this section since they not only possess similar logic but mutually share the utilization of many variables and plotting subrortines. These two procedures are invoked by the driver to create arrows and double-headed arrows in a variety of appearances

contigent upon the request of the programmer. As with the vectors, the arrows and double-headed arrows can be dotted, undotted, filled or unfilled. The arrow heads may be designed to assume a fixed size or they may have dimensions proportional to the given line segment. The crux of these two routines is tc solve a series of eguations sequentially to obtain a set of coordinate points for the corresponding figure. After the necessary computations, these procedures will activate the plotting routines to draw the requested graphical object.

The diagrams of the arrow and double-headed arrow appear below:

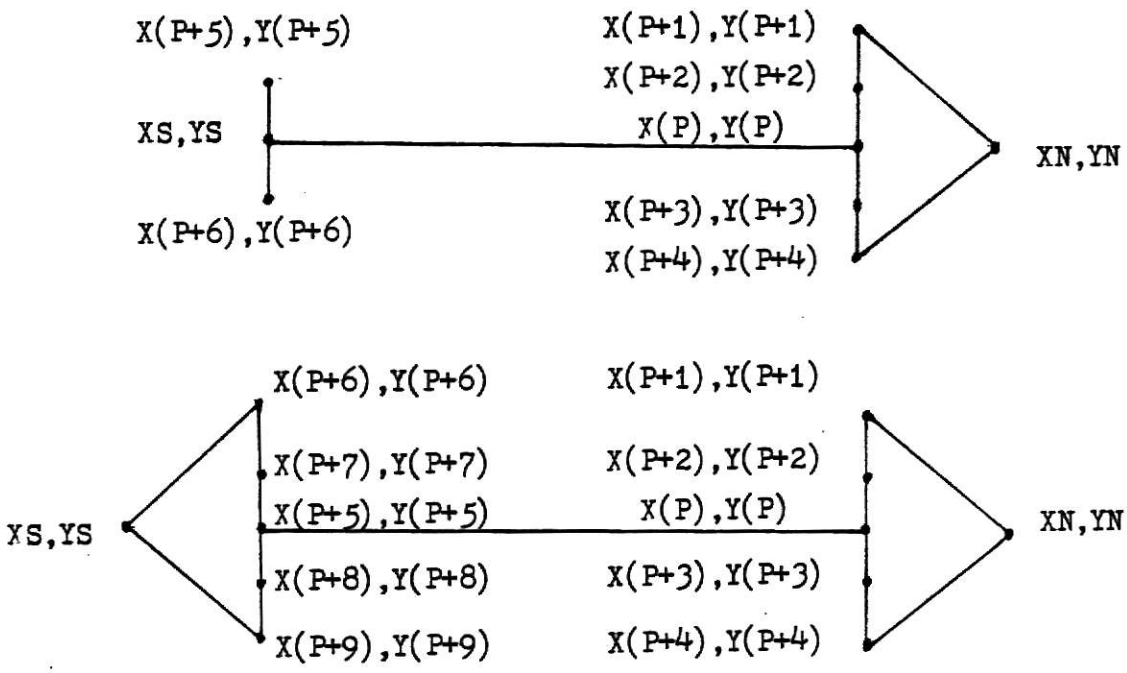


FIGURE 14

3.5.4 The Rectangle Routine

Chromatics terminal provides a built-in function allowing the user to draw rectangles by placing the window in Rectangle Plot Submode. While in this mode, the first and subsequent coordinate pair entered into the system causes a rectangle to be drawn with the two coordinates given as opposite corners. There are two limitations in the existing function which we wish to eliminate:

1) The rectangle created is either in a horizonal or vertical position.
2) The line segments encompassing the rectangle are of single width.

The main objective is, then, to furnish a mean for the programmer to create rectangles in any direction, as long as the prescribed boundaries of the rectangle are within the window boundaries and to allow the user to specify various thicknesses for the line segments. Subroutine Rectangle fulfils just that.

As with most of the primitive construction procedures, this routine is invoked by the main program, and the initial values are supplied via the Input Routine. Other pertinent coordinates are obtained from solving various equations systematically. After calling the Plotting Subroutine to outline the figure and having finished complex filled the object, the routine returns to its caller. The set of

equations used for this routine are enumerated below:

```
X(I)=XS+W*CO#
Y(I)=YS+W*SN#
X(I+1)=X(I)-W*SN#
Y(I+1)=Y(I)+W*CO#
X(I+2)=X(I)+(W-HT)*SN#
X(I+3)=XS-HT*SN#
Y(I+3)=YS+HT*CO#
X(I+4)=XS+(L-W)*CO#
Y(I+4)=YS+(L-W)*SN#
X(I+5)=X(I+4)-W*SN#
Y(I+5)=Y(I+4)+W*CO#
X(I+6)=X(I+4)+(W-HT)*SN#
Y(I+6)=Y(I+4)+(HT-W)*CO#
X(I+7)=XN-HT*SN#
Y(I+7)=YX+HT*CO#
```

where

X(I+J), Y(I+J)	represent the coordinates; 0<=J<=7.
(XS, YS)	is the starting point of the rectangle.
(XN, YN)	is the ending point of the rectangle.
HT	is the height of the rectangle.
W	is the width of the line segment.
L	is the length of the given line segment.

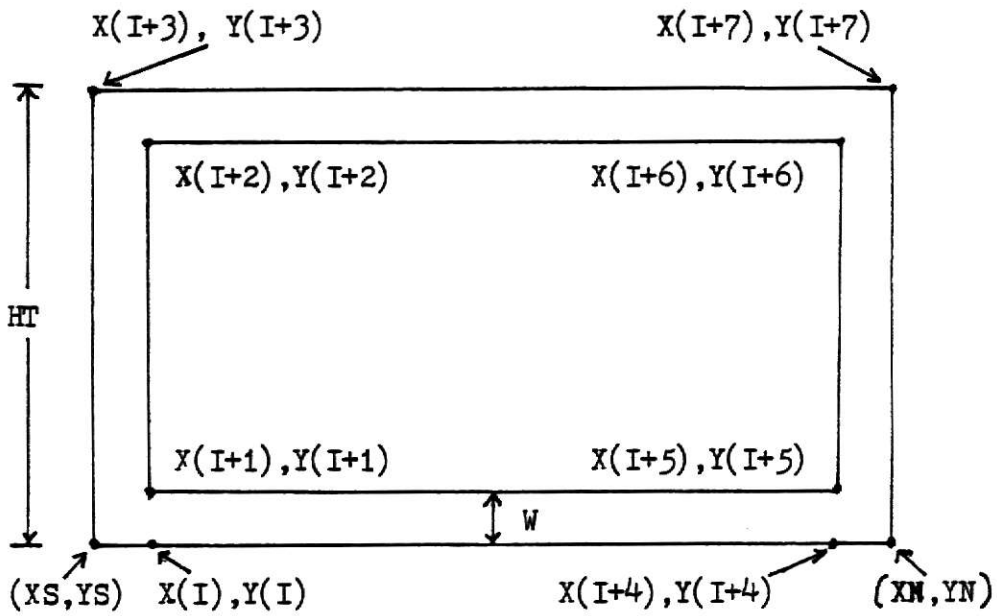

X(I+3), Y(I+3)
X(I+7),Y(I+7)
X(I+2),Y(I+2)
X(I+6),Y(I+6)
HT
X(I+1),Y(I+1)
X(I+5),Y(I+5)
W
(XS,YS)
X(I),Y(I)
X(I+4),Y(I+4)
(XN,YN)

FIGURE 15

3.5.5 The Bubble Routine

The bubble is constructed with the build-in arc function of the system. In order to create an arc, a set of points and angle measurements is needed. For example, the following components are necessary:

1) The center of the circle, of which the arc is a part.
2) The radius of the circle, and
3) The starting and ending points of an arc (these are measurements of degrees in counter clockwise direction).

The bubble is compiled with three upper and three lower arcs, and a pair of left and right arcs. To obtain the required coordinates and angle measurements for each of the arcs; we applied some fundamental knowledge of Analytic Geometry, namely, the equation of the slop and the point-slop equation of a line. It is unnecessary to elaborate on the derivation of each set of points and angles for each one of these arcs, since they can to calculated by the same method with slight variations. Therefore, the subsequent discussion centers only upon the general concept of how one set of points and angles was obtained.

Let the follcwing figure represents one of the loops of the bubble:

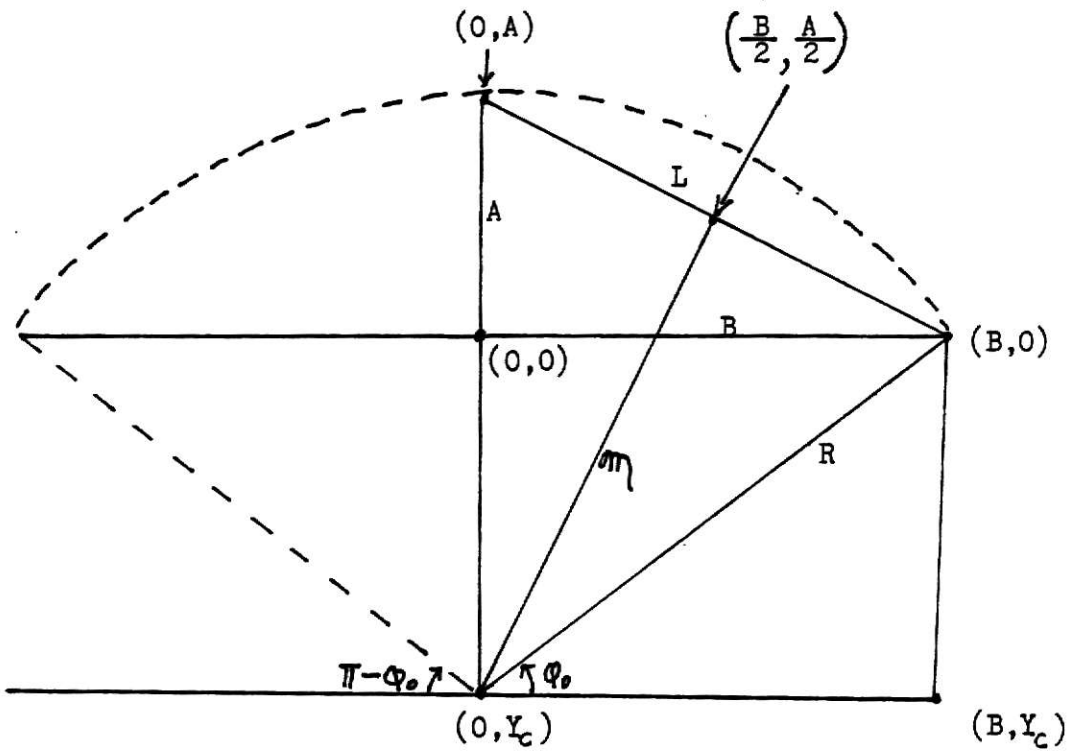

(0,A)
$\left(\frac{B}{2},\frac{A}{2}\right)$
L
A
B
(B,0)
(0,0)
m
R
$\pi-\varphi_o$
φ_o
(0,Y_c)
(B,Y_c)

FIGURE 16

The slop of I is $(A-0)/(0-B)=-A/B$,

Therefore, slop of η_1 is $M=B/A$.

Using the point-slop form of the equation of a line, we obtain:

$(Y-A/2)=M(X-B/2)=B/A(X-B/2)$.

setting $X=0$ and $Y=Y_c$,

the above equation becomes:

$Y_c-A/2=-(B/A)*(B/2)$,

and

$Y_c=-(B**2/2*A)+(A/2)$.

$TAN\varphi_0 = |Y_c|/B$, so that

$\varphi_0 = ATN\ (|Y_c|/B)$ where $|Y_c| = (B**2/2*A)-A/2$.

φ_0 is the angle measurement in radians of the starting position of the arc.

$\pi-\varphi_0$ is the angle measurement in radians of the ending position of the acr.

The radius of the circle is:

$R=A+|Y_c|=A+(B**2/2*A)-A/2=A/2+B**2/2*A$.

The coordinate of the center of the circle is (see figure 17)

$(2A+B,\ 10A-\sqrt{2}R)$.

We now have all the necessary equations for the coordinate points of one of the arcs. The next step is to code these equations into Basic statements which will generate the pertinent data points. As the BUB command concludes its calculation, it activiates the Plotting Routine which will retrieve these information

to draw the final figure.

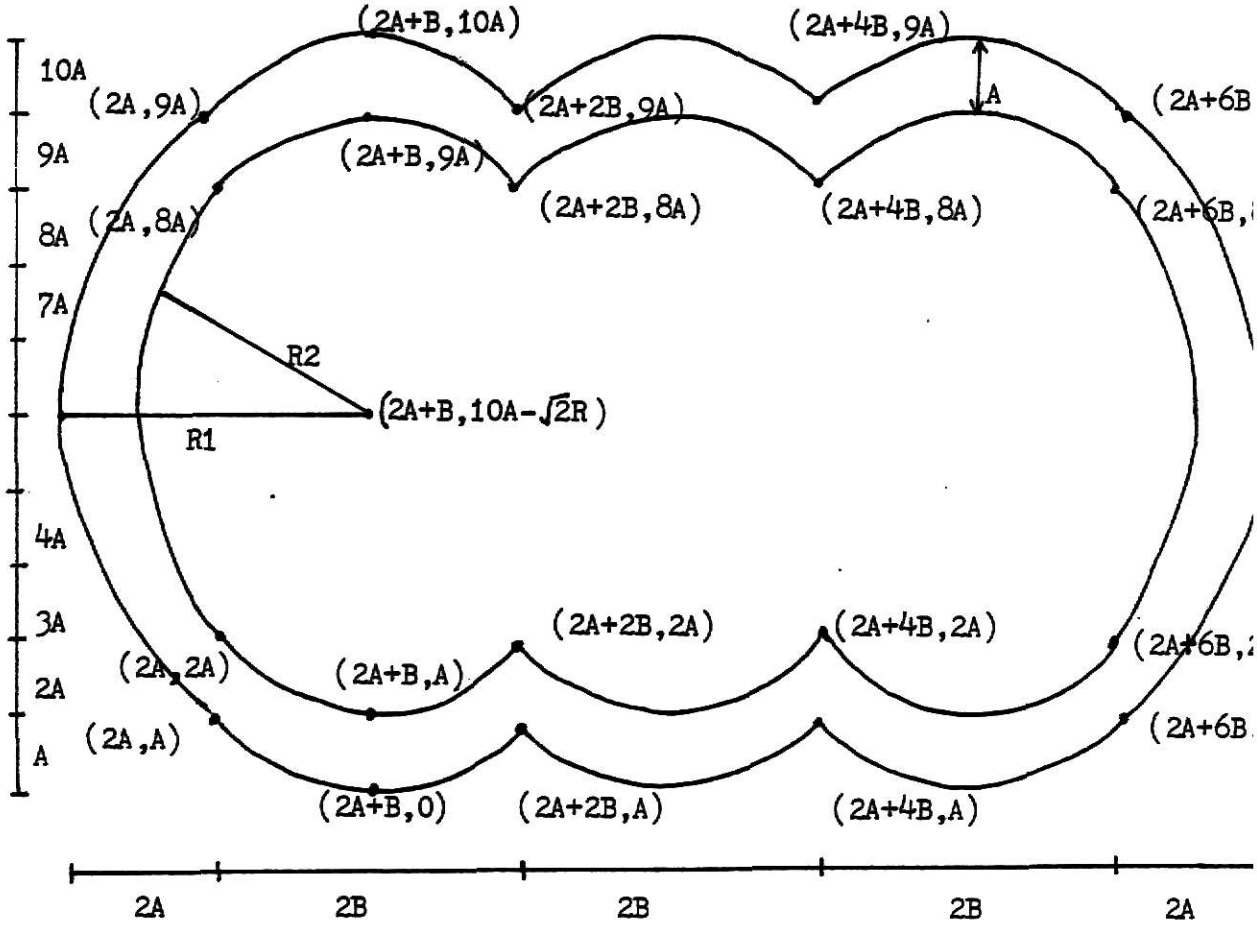


FIGURE 17

BIBLIOGRAPHY

Chr78a Chromatics Disk Software Reference Manual, Chromatis, Inc., Atlanta, Ga., 1978.

Chr78b Chromatics Preliminary Operator's Manual (Revised), Chromatics, Inc., Atlanta, Ga., November, 1978.

Chr79b Chromatis CG Basic Reference Manual, CG File Handling Basic Version 3.0, Chromatics, Inc., Atlanta, Ga., January, 1979.

Dil80 Dillinger, Marilyn, System Driver for Color Graphics Computer, Master's Report, KSU Department of Computer Science, 1980.

Ful72 Fuller, Gordon, Plane Trigonometry with Tables, McGraw-Hill Inc., 1972.

Mit81 Mitchell, Sharlene, Editing and Segmenting Display Files for Color Graphics, Master's Report, KSU Department of Computer Science, 1981.

New79 Newman, W. and Sproull, R. Principles of Interactive Computer Graphics, Second Edition, McGraw-Hill Book Co., 1979.

Tho60 Thomas, George B. Jr., Calculus and Analytic Geometry, Third Edition, Addision_Wesley Publishing Company, Inc., 1960.

APPENDIX

```
5 'AUTHOR-MAXINE YEE
6 'PROGRAM-EGP(EXTENDED GRAPHIC PRIMITIVES)
7 'LANGUAGE-CHROMATICS BASIC VERSION 3.0
8 'PLACE-KANSAS STATE UNIVERSITY,DEPT. OF COMPUTER SCIENCE
9 'DATE-DEC.,1979
10 '*************************************************************************
11 '
12 ' THIS IS A PROGRAM FOR THE GENERATION OF EXTENDED GRAPHIC PRIMITIVES
13 '
14 '*************************************************************************
52 '
54 PRINT"THIS IS A PGOGRAM FOR THE GENERATION OF EXTENDED GRAPHIC PRIMITIVES"
55 PRINT"~J";CHR$(&HC);                    'VISIBLE CURSOR;ERASE PAGE
56 PRINT CHR$(27);"0A1";                  'WINDOW 1
57 PRINT"~W000511511490";                 'SIZE (000,511),(511,490)
58 PRINT"~M~C1~N~C7";                     'BG BLUE,FG WHITE
59 PRINT CHR$(&H15);"~J~P";               'CHAR MODE;CURSOR ON;ROLL OFF
60 PRINT CHR$(&HC);                       'ERASE PAGE
65 CLEAR
70 DIM X(200),Y(200)
75 PRINT CHR$(12);
80 INPUT"TYPE IN THE COMMAND";CMD$
90 IF CMD$="END" THEN 65500
95 PRINT CHR$(12);                        'ERASE PAGE
100 INPUT"TYPE IN THE BACKGROUND AND FOREGROUND COLORS";BC,FC
105 PRINT CHR$(12);                       'ERASE PAGE
107 SQ=SQR(2)
110 INPUT"TYPE IN THE ATTRIBUTES FOR THE OBJECT";DO$,FI$,BL$
115 IF BL$="B" THEN PRINT"~1";ELSE PRINT "~2"
120 IF CMD$="VEC" THEN GOSUB 1500:GOTO 55
130 IF CMD$="ARR" THEN GOSUB 5000:GOTO 55
140 IF CMD$="DAR" THEN GOSUB 10000:GOTO 55
150 IF CMD$="REC" THEN GOSUB 20000:GOTO 55
160 IF CMD$="BUB" THEN GOSUB 30000:GOTO 55
180 PRINT "SORRY, WRONG COMMAND":  GOTO 55
```

```
1500 '
1501 '*************************************************************************
1502 '
1503 '       THIS IS A SUBROUTINE FOR DRAWING WIDE VECTORS
1504 '
1505 '*************************************************************************
1506 '
1510 GOSUB 45000
1525 IF DO$="D" THEN GOSUB 2200:GOTO 1510
1527 YS=YS/SQ:YN=YN/SQ
1530 X=XN-XS
1540 Y=YN-YS
1550 L#=SQR(X^2+Y^2)                    'CAL.LENGTH OF GIVEN LINE
1560 X(I+2)=XS-(W*Y)/L#                 'CAL X2 COORD
1570 Y(I+2)=YS+(W*X)/L#                 'CAL.Y2 COORD
1580 X(I+1)=XN-(W*Y)/L#                 'CAL.X1 COORD
1590 Y(I+1)=YN+(W*X)/L#                 'CAL.Y1 COORD
1592 Y(I+2)=Y(I+2)*SQ
1594 Y(I+1)=Y(I+1)*SQ
1596 YS=YS*SQ:YN=YN*SQ
1600 IF FI$<>"F" THEN GOSUB 54000
                 ELSE GOSUB 54500              'SUB FOR PLOTTING ENV.
1620 PLOT XS,YS,XN,YN
1625 PLOT XN,YN,X(I+1),Y(I+1)
1630 PLOT X(I+1),Y(I+1),X(I+2),Y(I+2)
1640 PLOT X(I+2),Y(I+2),XS,YS
1660 IF FI$="F" THEN XM#=(XS+X(I+1))/2:YM#=(YS+Y(I+1))/2:GOSUB 55000
1670 IF DO$="D" THEN RETURN  ELSE GOSUB 55500
1675 GOTO 1510
1680 RETURN
```

```
2200 '
2201 '*************************************************************************
2202 '
2203 '        THIS IS A SUBROUTINE FOR DRAWING DOTTED LINES
2204 '
2205 '*************************************************************************
2206 YS=YS/SQ:YN=YN/SQ
2207 J=1
2230 X(J)=XS
2240 Y(J)=YS
2241 XB=XS:YB=YS
2270 D = 12                                    'ASSIGN SOLID SEGMENT OF THE DOTTED LINE
2280 SEG = D
2290 SP = SEG\2                                'ASSIGN SPACE OF THE DOTTED LINE
2310 L1= SQR((XN-XS)^2 + (YN-YS)^2)            'CAL LENGTH OF GIVEN LINE
2315 IF L1<SP AND CMD$="VEC" THEN  1510
2317 IF L1<SP AND CMD$="ARR"THEN 5050
2319 IF L1<SP AND CMD$="DAR" THEN 10020
2320 CO# = (XN-XS)/L1                          'CAL COSINE OF AN ANGLE
2330 SN# = (YN-YS)/L1                          'CAL SINE OF AN ANGLE
2340 K = (L1\(SP +SEG))*2 + 1                  'CAL TERMINAL VALUE FOR THE LOOP
2350 FOR J = 1 TO K STEP 1
2360 X(J+1) = XB + D*(CO#)                     'CAL X(I) COORD
2370 Y(J+1) = YB + D*(SN#)                     'CAL Y(I) COORD
2380 '
2390 '
2400 'THIS SECTION OF CODE TESTS THE VARIOUS END POINTS OF A GIVEN LINE
2410 E = J MOD 2
2420 IF E=0 THEN IF (L1-D) <SEG
                 THEN D=D+(L1-D):NEXT J
                 ELSE D=D+SEG:NEXT J
            ELSE GOSUB 54000
2430 '
2440 '
2450 IF W<=1 THEN Y(J)=Y(J)*SQ:Y(J+1)=Y(J+1)*SQ:GOSUB 3100
2465 IF W>1 THEN GOSUB 3000                    'SUB FOR DRAWING DOTTED UNFILLED LINE
2470 IF (L1-D)<=SP AND CMD$="VEC" THEN GOSUB 55500: GOTO 1510   'REMAINING SEG TOO SHORT TO PLOT
2472 IF (L1-D)<=SP AND CMD$="ARR" THEN GOSUB 55500: GOTO 5050   'REMAINING SEG TOO SHORT TO PLOT
2475 IF (L1-D)<=SP AND CMD$="DAR" THEN GOSUB 55500:GOTO 10020
2480 D = D + SP
2485 IF E=1 AND L1-D<=SP AND CMD$="VEC" THEN GOSUB 55500:GOTO 1510
2490 NEXT J
2500 RETURN
```

```
3000 '
3010 '
3020 'THIS IS A SUBROUTINE FOR DRAWING DOTTED WIDE LINES
3030 XN=X(J+1)
3040 YN=Y(J+1)
3050 XS=X(J)
3060 YS=Y(J)
3070 IF XS=XN AND YS=YN THEN RETURN ELSE GOSUB 1527
3080 RETURN
3081 '
3100 '
3110 YS=YS*SQ:YN=YN*SQ
3120 PLOT X(J),Y(J),X(J+1),Y(J+1)
3130 RETURN
5000 '*****************************************************************
5001 '
5002 '          THIS IS A SUBROUTINE FOR DRAWING ARROWS
5003 '
5004 '*****************************************************************
5005 '
5010 '
5020 PRINT "THIS IS A SUBROUTINE FOR DRAWING ARROWS"
5050 GOSUB 45000
5070 INPUT"TYPE IN VAR FOR VARIABLE ARROW HEAD OR FIX FOR FIXED HEAD";AH$
5090 PRINT CHR$(12);                              'ERASE PAGE
5092 YS=YS/SQ:YN=YN/SQ
5100 X = XN-XS
5110 Y=YN-YS
5120 L2= SQR((XN-XS)^2 + (YN-YS)^2)               'CAL LENGTH OF GIVEN LINE
5130 W1= W/2                                      'CAL WIDTH OF TAIL
5140 SI# = Y/L2                                   'CAL SINE OF AN ANGLE
5150 CS# = X/L2                                   'CAL COSINE OF AN ANGLE
5160 IF AH$ = "FIX" AND W1<=1 THEN GOSUB 5410: GOTO 5260
5165 IF AH$ = "FIX" AND W1 >1 THEN GOSUB 5410
5167 IF AH$="FIX" AND DO$="D" THEN 5290
5170 '
```

```
5180 '
5190 'THIS SECTION OF CODE CALCULATES THE VARIABLE SIZE OF AN ARROW HEAD
5191 P=2
5200 X(P) = XS + 9/10*X
5210 Y(P) = YS + 9/10*Y
5220 X(P+1) = X(P) - Y/10
5230 Y(P+1) = Y(P) + X/10
5240 X(P+4) = X(P) + Y/10
5250 Y(P+4) = Y(P) - X/10
5252 Y(P+1)=Y(P+1)*SQ
5253 Y(P+4)=Y(P+4)*SQ
5254 YN=YN*SQ
5255 IF DO$="D" AND W1<=1 THEN 5505
5260 IF W1<=1 THEN GOSUB 54000:YS=YS*SQ:Y(P)=Y(P)*SQ:GOSUB 5610:GOSUB 55500:GOTO 5050

5270 '
5280 '
5290 'THIS SECTION OF CODE CALCULATES COORD FOR THE TAIL OF AN ARROW
5300 X(P+2) = X(P) - W1*SI#
5310 Y(P+2) = Y(P) + W1*CS#
5320 X(P+3) = X(P) + W1*SI#
5330 Y(P+3) = Y(P) - W1*CS#
5340 X(P+5) = XS - W1*SI#
5350 Y(P+5) = YS + W1*CS#
5352 X(P+6) = XS + W1*SI#
5354 Y(P+6) = YS - W1*CS#
5356 YS = YS*SQ
5360 Y(P+2)=Y(P+2)*SQ
5362 Y(P+3)=Y(P+3)*SQ
5364 Y(P+5)=Y(P+5)*SQ
5366 Y(P+6)=Y(P+6)*SQ
5372 IF DO$="D" AND AH$="FIX"THEN 5507
5375 IF DO$="D" AND AH$="VAR"THEN 5507
5380 GOSUB 54000:GOSUB 5720:Y(P)=Y(P)*SQ:GOSUB 5670:GOSUB 55500:GOTO 5050
5400 RETURN
5410 '
```

```
5420 '
5430 'THIS SUBROUTINE CALCULATES FIXED ARROW HEAD
5440 H = 10                              'ASSIGN LENGTH FOR THE ARROW HEAD
5445 P = 2
5450 X(P) = XS + (L2-H)*CS#
5460 Y(P) = YS + (L2-H)*SI#
5470 X(P+1) = X(P) - H*SI#
5472 Y(P+1) = Y(P) + H*CS#
5474 X(P+4) = X(P) + H*SI#
5476 Y(P+4) = Y(P) - H*CS#
5480 Y(P+1) = Y(P+1)*SQ
5482 Y(P+4) = Y(P+4)*SQ
5484 YN=YN*SQ
5502 IF DO$<>"D" THEN 5260
5505 IF W1<=1 THEN GOSUB 54000:Y(P)=Y(P)*SQ:GOSUB 5640:XN=X(P):YN=Y(P):YS=YS*SQ: GOSUB 2206:GOTO 5050
5506 IF AH$="FIX" THEN 5290
5507 IF W1>1 THEN GOSUB 54000:Y(P)=Y(P)*SQ:GOSUB 5640:GOSUB 6100:GOSUB 2270:GOSUB 55500:GOTO 5050
5510 RETURN
5610 '
5611 'THIS IS A PLOTTING SUBROUTINE FOR TRIANGLE
5630 PLOT XS,YS,X(P),Y(P)
5640 PLOT X(P+1),Y(P+1),X(P+4),Y(P+4)
5650 PLOT X(P+4),Y(P+4),XN,YN
5660 PLOT XN,YN,X(P+1),Y(P+1)
5670 IF FI$="F" THEN XM#=(X(P)+XN)/2:YM#=(Y(P)+YN)/2:GOSUB 55000
5710 RETURN
5720 '
5730 '
5930 'THIS IS A PLOTTING SUBROUTINE FOR WIDE ARROW
5950 PLOT X(P+2),Y(P+2),X(P+1),Y(P+1)
5960 PLOT X(P+1),Y(P+1),XN,YN
5970 PLOT XN,YN,X(P+4),Y(P+4)
5980 PLOT X(P+4),Y(P+4),X(P+3),Y(P+3)
5990 PLOT X(P+3),Y(P+3),X(P+6),Y(P+6)
6000 PLOT X(P+6),Y(P+6),X(P+5),Y(P+5)
6010 PLOT X(P+5),Y(P+5),X(P+2),Y(P+2)
6020 RETURN
6100 '
```

```
6101 '
6102 'THIS IS A SUBROUTINE FOR DRAWING DOTTED WIDE ARROWS
6110 XN=X(P+3):YN=Y(P+3)
6130 XS=X(P+6):YS=Y(P+6)
6131 J=1
6141 XB=X(P+6):YB=Y(P+6)
6142 X(J)=X(P+6):Y(J)=Y(P+6)
6150 RETURN
10000 '
10001 '*************************************************************************
10002 '
10003 'THIS IS A SUBROUTINE FOR DRAWING DOUBLE HEADED ARROWS
10004 '
10005 '*************************************************************************
10010 PRINT "THIS IS A SUBROUTINE FOR DRAWING DOUBLE HEADED ARROWS"
10020 GOSUB 45000
10040 INPUT"TYPE IN VAR FOR VARIABLE ARROW HEAD OR FIX FOR FIXED ARROW HEAD";AH$
10050 PRINT CHR$(12);
10060 X=XN-XS
10070 Y=YN-YS
10080 L3=SQR((XN-XS)^2 + (YN-YS)^2)          'CAL LENGTH OF GIVEN LINE
10090 W1=W/2
10100 SI#=Y/L3                               'CAL SINE
10110 CS#=X/L3                               'CAL COSINE
10120 IF AH$="FIX" AND W1<=1 THEN GOSUB 10500:GOSUB 54000:GOSUB 10900:GOSUB 55500:GOTO 10020
10125 IF AH$="FIX" AND W1>1 THEN GOSUB 10500:GOTO 10300
10126 '
```

```
10127 '
10130 'THIS SECTION OF CODES CALCULATES THE VARIABLE SIZE OF THE ARROW HEADS
10140 '
10150 P=2
10160 X(P)=XS+9/10*X
10170 Y(P)=YS+9/10*Y
10180 X(P+1)=X(P)-Y/10
10190 Y(P+1)=Y(P)+X/10
10200 X(P+4)=X(P)+Y/10
10210 Y(P+4)=Y(P)-X/10
10220 X(P+5)=XS+X/10
10230 Y(P+5)=YS+Y/10
10240 X(P+6)=X(P+5)-Y/10
10250 Y(P+6)=Y(P+5)+X/10
10260 X(P+9)=X(P+5)+Y/10
10270 Y(P+9)=Y(P+5)-X/10
10280 IF DO$="D" AND W1<=1 THEN GOSUB 54000:GOSUB 10940:GOTO 11300
10282 IF DO$="D" AND W1>=1 THEN 10310
10290 IF W1<=1 THEN GOSUB 54000:GOSUB 10920:GOSUB 55500:GOTO 10020
10300 '
10310 'THIS SECTION OF CODE CALCULATES WIDE TAIL OF THE DOUBLE HEADED ARROWS
10320 '
10330 X(P+2)=X(P)-W1*SI#
10340 Y(P+2)=Y(P)+W1*CS#
10350 X(P+3)=X(P)+W1*SI#
10360 Y(P+3)=Y(P)-W1*CS#
10370 X(P+7)=X(P+5)-W1*SI#
10380 Y(P+7)=Y(P+5)+W1*CS#
10390 X(P+8)=X(P+5)+W1*SI#
10400 Y(P+8)=Y(P+5)-W1*CS#
10405 IF DO$="D" THEN GOSUB 54000:GOSUB 10940:GOSUB 11300:GOSUB 2270:GOSUB 55500:GOTO10020
10410 GOSUB 54000:GOSUB 11100:GOSUB 55500:GOTO 10020
10500 '
```

```
10510 '
10520 'THIS SECTION OF CODE CALCULATES FIX ARROW HEAD
10530 H=12
10540 P=2
10550 X(P)=XS+(L3-H)*CS#
10560 Y(P)=YS+(L3-H)*SI#
10570 X(P+1)=X(P)-H*SI#
10580 Y(P+1)=Y(P)+H*CS#
10590 X(P+4)=X(P)+H*SI#
10600 Y(P+4)=Y(P)-H*CS#
10610 X(P+5)=XS+H*CS#
10620 Y(P+5)=YS+H*SI#
10630 X(P+6)=X(P+5)-H*SI#
10640 Y(P+6)=Y(P+5)+H*CS#
10650 X(P+9)=X(P+5)+H*SI#
10660 Y(P+9)=Y(P+5)-H*CS#
10665 IF DO$="D" THEN GOSUB 54000:GOSUB 10940:GOSUB 10300
10670 RETURN
10900 '
10910 '
10920 'THIS IS A PLOTTING SUBROUTINE FOR THE DOUBLE HEADED ARROWS
10930 PLOT X(P+5),Y(P+5),X(P),Y(P)
10940 PLOT X(P+1),Y(P+1),XN,YN
10950 PLOT XN,YN,X(P+4),Y(P+4)
10960 PLOT X(P+4),Y(P+4),X(P+1),Y(P+1)
10970 PLOT X(P+6),Y(P+6),X(P+9),Y(P+9)
10980 PLOT X(P+9),Y(P+9),XS,YS
10990 PLOT XS,YS,X(P+6),Y(P+6)
11000 IF FI$="F" THEN XM#=(X(P)+XN)/2:YM#=(Y(P)+YN)/2:GOSUB 55000
11010 IF FI$="F" THEN XM#=(X(P+5)+XS)/2:YM#=(Y(P+5)+YS)/2:GOSUB 55000
11020 RETURN
11100 '
```

```
11110 '
11120 'THIS IS A PLOTTING SUBROUTINE FOR WIDE DOUBLE HEADED ARROWS
11130 PLOT X(P+2),Y(P+2),X(P+1),Y(P+1)
11140 PLOT X(P+1),Y(P+1),XN,YN
11150 PLOT XN,YN,X(P+4),Y(P+4)
11160 PLOT X(P+4),Y(P+4),X(P+3),Y(P+3)
11170 PLOT X(P+3),Y(P+3),X(P+8),Y(P+8)
11180 PLOT X(P+8),Y(P+8),X(P+9),Y(P+9)
11190 PLOT X(P+9),Y(P+9),XS,YS
11200 PLOT XS,YS,X(P+6),Y(P+6)
11210 PLOT X(P+6),Y(P+6),X(P+7),Y(P+7)
11220 PLOT X(P+7),Y(P+7),X(P+2),Y(P+2)
11230 IF FI$="F" THEN XM#=(X(P)+XN)/2:YM#=(Y(P)+YN)/2:GOSUB 55000
11240 RETURN
11300 '
11310 '
11320 'THIS SECTION OF CODE PLOTS DOTTED LINE FOR THE DOUBLE HEADED ARROWS
11330 IF W1<=1 THEN 11500
11340 XS=X(P+8):YS=Y(P+8)
11350 XN=X(P+3):YN=Y(P+3)
11360 XB=X(P+8):YB=Y(P+8)
11370 J=1:X(J)=X(P+8):Y(J)=Y(P+8)
11380 RETURN
11500 '
11510 '
11520 'THIS SECTION OF CODE PLOTS SINGLE DOTTED LINE FOR THE ARROW HEADS
11530 XS=X(P+5):YS=Y(P+5)
11540 XN=X(P):YN=Y(P)
11550 GOSUB 2200:GOSUB 55500:GOTO 10020
11560 RETURN
```

```
20000 '
20010 '*************************************************************************
20020 '
20030 'THIS IS A SUBROUTINE FOR DRAWING RECTANGLES
20040 '
20050 '*************************************************************************
20060 '
20070 GOSUB 45500
20080 IF XS=999 THEN RETURN
20090 X=XN-XS
20100 Y=YN-YS
20120 L=SQR(X^2+Y^2)                    'CAL LENGTH OF GIVEN LINE
20130 CO#=X/L                           'CAL COSINE OF AN ANGLE
20140 SN#=Y/L                           'CAL SINE OF AN ANGLE
20150 '
20160 'FOLLOWING SECTION OF CODE CALCULATES THE VARIOUS COORD. OF THE RECTANGLE
20170 '
20180 I=1
20190 X(I)=XS+W*CO#
20200 Y(I)=YS+W*SN#
20210 X(I+1)=X(I)-W*SN#
20220 Y(I+1)=Y(I)+W*CO#
20230 X(I+2)=X(I)+(W-HT)*SN#
20240 Y(I+2)=Y(I)+(HT-W)*CO#
20250 X(I+3)=XS-HT*SN#
20260 Y(I+3)=YS+HT*CO#
20270 X(I+4)=XS+(L-W)*CO#
20280 Y(I+4)=YS+(L-W)*SN#
20290 X(I+5)=X(I+4)-W*SN#
20300 Y(I+5)=Y(I+4)+W*CO#
20310 X(I+6)=X(I+4)+(W-HT)*SN#
20320 Y(I+6)=Y(I+4)+(HT-W)*CO#
20330 X(I+7)=XN-HT*SN#
20340 Y(I+7)=YN+HT*CO#
20350 GOSUB 54000:GOSUB 20700:GOSUB 55500:GOTO 20070
20360 RETURN
```

```
20700 '
20710 'THE FOLLOWING SECTION OF CODE PLOTS THE RECTANGLE
20720 '
20730 PLOT XS,YS,XN,YN
20740 PLOT XN,YN,X(I+7),Y(I+7)
20750 PLOT X(I+7),Y(I+7),X(I+3),Y(I+3)
20760 PLOT X(I+3),Y(I+3),XS,YS
20770 PLOT X(I+1),Y(I+1),X(I+5),Y(I+5)
20780 PLOT X(I+5),Y(I+5),X(I+6),Y(I+6)
20790 PLOT X(I+6),Y(I+6),X(I+2),Y(I+2)
20800 PLOT X(I+2),Y(I+2),X(I+1),Y(I+1)
20810 IF FI$="F" THEN XM#=(XS+X(I+1))/2:YM#=(YS+Y(I+1))/2:GOSUB 55000
20820 RETURN
30000 '
30010 '*************************************************************************
30020 '
30030 'THIS IS A SUBROUTINE FOR DRAWING BUBBLES
30040 '
30050 '*************************************************************************
30060 '
30070 GOSUB 45500
30080 IF XS=999 THEN RETURN
30090 PI=3.1416
30100 CV=180/PI
30101 SQ=SQR(2)
30110 L=XN-XS                          'CALC LENGTH OF GIVEN LINE
30111 IF L<HT THEN 35000
30120 A=HT/10                          'CALC WIDTH OF THE BUBBLE
30125 A1=A/SQ                          'PUT IN SCALE FACTOR
30130 B=(L-4*A)/6                      'CALC ARC DISTANCE
30140 R=A1/2+(B^2)/(2*A1)              'CALC RADIUS OF THE FOUR ARCS
30150 T=ATN((B^2/(2*A1)-(A1/2))/B)     'CALC STARTING ANGLE FOR THE ARC
30160 I=1
30170 X(I)=XS+(2*A+B)                  'CALC X COORDS OF THE FOUR ARC
30180 X(I+1)=X(I)+2*B
30190 X(I+2)=X(I+1)+2*B
30210 Y(I)=YS+(10*A-SQ*R)              'CALC Y COORDS OF THE OUTER ARC
30220 Y(I+3)=YS+(SQ*R)
30230 T1=T*CV                          'CALC THE VARIOUR ANGLES FOR THE FOUR
                                       'ARC AND CONVERTING THEM FROM RAD TO DEG
```

```
30240 T2=(PI-2*T)*CV
30250 T3=(PI+T)*CV
30260 R2=3*A                                    'CALC RADIUS FOR LEFT AND RIGHT ARC
30270 X2=XS+R2                                  'CAL X AND Y COORDS FOR LEFT AND RT  ARC
30280 Y2=YS+5*A
30290 X4=XN-R2
30300 Y4=Y2
30310 Z=ATN(4/SQ)                               'CALC VARIOUS ANGLES OF THE LT AND RT ARC
30320 Z1=180-Z*CV
30330 Z2=2*Z*CV
30340 Z3=360-Z*CV
30350 Z4=2*Z*CV
30360 IF W<=1 THEN GOSUB 54000:GOSUB 31000:GOSUB 33000:GOSUB 55500:GOTO 30070
30370 R1=A*SQR(8.5)                             'CALC RADIUS FOR THE LT AND RT IN ARC
30380 Y(I+1)=Y(I)-A                             'CALC Y COORDS FOR THE FOUR IN ARC
30390 Y(I+2)=Y(I+3)+A
30400 X1=X2+A                                   'CALC X AND Y COORD FOR LT IN ARC
30410 Y1=Y2
30420 X3=XN-A-R1                                'CAL X AND Y COORD FOR RT IN ARC
30430 Y3=Y2
30440 Z5=ATN(3/(2*SQ))                          'CALC VARIOUS ANGLES FOR LT AND RT IN ARC
30450 Z6=180-Z5*CV
30460 Z7=2*Z5*CV
30470 Z8=360-Z5*CV
30480 Z9=2*Z5*CV
30490 GOSUB 54000:GOSUB 31000:GOSUB 32000:GOSUB 33000:GOSUB 33200
30510 GOSUB 55500 GOTO 30070
31000 '
31010 'THIS IS A SUBROUTINE FOR THE GENERATION OF THE OUTER THREE ARCS
31020 '
31030 FOR J=1 TO 4 STEP 3
31040     FOR I=1 TO 3 STEP 1
31050         IF J=1 THEN PLOT X(I),Y(J),R,T1,T2
                      ELSE PLOT X(I),Y(J),R,T3,T2
31060     NEXT I
31070 NEXT J
31080 RETURN
```

```
32000 '
32010 'THIS IS A PLOTTING SUBROUTINE FOR THE GENERATION OF INNER THREE ARCS
32020 '
32030 FOR J=2 TO 3 STEP 1
32040     FOR I=1TO 3 STEP 1
32050         IF J=2 THEN PLOT X(I),Y(J),R,T1,T2
                     ELSE PLOT X(I),Y(J),R,T3,T2
32060     NEXT I
32070 NEXT J
32080 RETURN
33000 '
33010 'THIS IS A PLOTTING SUBROUTINE FOR THE GENERATION OF LT AND RT OUTER ARC
33020 PLOT X2,Y2,R2,Z1,Z2
33030 PLOT X4,Y4,R2,Z3,Z4
33040 RETURN
33200 '
33210 'THIS IS A PLOTTING SUBROUTINE FOR THE GENERATION OF LT AND RT ARC
33220 PLOT X1,Y1,R1,Z6,Z7
33230 PLOT X3,Y3,R1,Z8,Z9
33240 RETURN
45000 '
45010 '
45020 'THIS SUBROUNTINE HANDLES THE INPUT OF COORD
45025 I=1
45030 INPUT"TYPE IN FIRST COORD FOR YOUR OBJECT";XS,YS
45031 INPUT"TYPE IN SECOND COORD AND WIDTH FOR YOUR OBJECT";XN,YN,W
45040 RETURN
45500 '
45510 'THIS SUBROUTINE HANDLES THE INPUT OF COORD FOR RECTANGLES AND BUBBLES
45520 '
45530 INPUT"TYPE IN FIRST COORD FOR YOUR OBJECT";XS,YS
45540 INPUT"THPE IN SECOND COORD FOR YOUR OBJECT";XN,YN
45550 INPUT"TYPE IN THE HEIGHT AND WIDTH OF YOUR OBJECT";HT,W
45560 RETURN
```

```
54000 '
54010 '
54020 'THIS IS A SUBROUTINE TO HANDLE THE PLOTTING ENVIRONMENT AND SET COLOR
54050 PRINT CHR$(1);"M";                'BACKGROUND LIGHT ON
54060 PRINT CHR$(1);"C";CHR$(48+BC);    'SET BACKGROUND COLOR
54070 PRINT CHR$(1);"N";                'BACKGROUND LIGHT OFF
54080 PRINT CHR$(1);"C";CHR$(48+FC);    'SET FOREGROUND COLOR
54085 PRINT CHR$(1);"G";                'PLOT ON
54087 IF CMD$="BUB" THEN PRINT ")" ELSE PRINT "'";
54090 RETURN
54500 '
54510 '
54520 'THIS IS A SUBROUTINE TO HANDLE THE PLOTTING ENVIRONMENT
54530 PRINT "~G";                       'PLOT ON
54535 PRINT "'";                        'VECTOR SUBMODE
54550 RETURN
55000 '
55010 '
55020 'THIS SUBROUTINE COMPLEX FILLS AN OBJECT
55030 PRINT "~U"; : PLOT XM#,YM#                'MOVE CURSOR TO COORD
55035 PRINT "~J";
55040 PRINT "~M";                               'BACKGROUND LIGHT ON
55050 PRINT CHR$(1);"C";CHR$(48+BC);            'SET BACKGROUND COLOR
55060 PRINT "~N";                               'BACKGROUNG LIGHT OFF
55070 PRINT CHR$(1);"C";CHR$(48+FC);            'SET FOREGROUND COLOR
55080 PRINT "~>"; CHR$(32);                     'FILL OBJECT WITH SOLID COLOR
55090 RETURN
55500 '
55510 '
55520 'THIS SUBROUTINE CANCELS THE PLOTTING ENVIRONMENT
55530 PRINT CHR$(21);                           'CANCELS PLOT MODE
55540 PRINT CHR$(28);                           'CURSOR HOME
55550 RETURN
65500 END
```

IMPLEMENTATION OF EXTENDED GRAPHIC PRIMITIVES

by

MAXINE F. YEE

B. S., University of California, Berkeley, 1962

AN ABSTRACT OF A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1982

ABSTRACT

EGP(Extended Graphics Primitives) is an interactive program which expands the existing graphics primitives. It allows the user to construct a selected set of two-dimensional figures in terms of dots and to display these figures on the Chromatic 1999 terminal screen. The goals of this project include:

(1) Design and implement a program to generate the following figures:
   A. Wide Vector.
   B. Arrow.
   C. Double-headed arrow.
   D. Rectangle.
   E. Bubble.
(2) Test and Degug the program.
(3) Provide a detailed documentation for future programmers who might like to make modifications or further extensions.
(4) Provide a short user's guide.

The program is not a self-contained primer, nor is it an integrated graphics package. User's interfaces, memory overlays, selective modification of objects were developed to make the package more versatile.

The program has been tested with several different data sets and the results were successful.