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:

[Signature]
Major Professor
# TABLE OF CONTENTS

**LIST OF FIGURES**............................................................................................................. iii

**I. INTRODUCTION**

A. Overview.................................................................................................................... 2

B. Paper Organization..................................................................................................... 2

C. Evaluation.................................................................................................................... 8

**II. USER'S GUIDE**

A. Implemented Commands............................................................................................ 10
   1. The VAR Command.................................................................................................. 13
   2. The ARR Command.................................................................................................. 15
   3. The DAR Command.................................................................................................. 17
   4. The REC Command.................................................................................................. 19
   5. The BUB Command.................................................................................................. 21

B. Sample Terminal Session........................................................................................... 23

**III. DOCUMENTATION**

A. Overview of the Program.............................................................................................. 25
   1. Primitive Construction Routines............................................................................... 26
   2. Utility Routines........................................................................................................ 26
   3. Plotting Routines..................................................................................................... 26

B. Description of Variables and Their Usage.................................................................. 28
   1. Array X and Y.......................................................................................................... 29
   2. Variables BC and FC............................................................................................... 29
   3. Variables AH$, DO$, FP$ and HI$............................................................................. 30
   4. Variables XS, YS, XN, YN, W and HT................................................................... 30
   5. Variable SQ............................................................................................................ 31
This is as received from the customer.

From the page to the bottom, the top of the page differs from being skewed with the original printing being.

Numerous pages contain this book.
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.
6. Other Variables...................................................... 31

C. Description of Routines............................................... 33

1. Main Routine.......................................................... 33

2. Primitive Construction Routines.................................... 39

3. Utility Routines
   i. The Input Routines.................................................. 40
   ii. The Set Environment Routine................................... 40
   iii. The Complex Filled Routine.................................... 41
   iv. Cancel Plot Mode Routine....................................... 41

4. Plotting Routines..................................................... 41

5. Primitive Construction Routine..................................... 42
   i. The Vector Routine................................................ 42
   ii. The Dotted Vector Routine...................................... 44
   iii. The Arrow and Double Headed Arrow Routines.................. 45
   iv. The Rectangle Routine.......................................... 47
   v. The Bubble Routine.............................................. 50

BIBLIOGRAPHY.......................................................... 54

APPENDIX................................................................. 55
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Vector</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Arrow</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Double-Headed Arrow</td>
<td>5</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Rectangle</td>
<td>6</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Bubble</td>
<td>7</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Color Table</td>
<td>11</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Hierarchical Chart</td>
<td>25</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Pseudocode For Driver</td>
<td>34</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Pseudocode For Vector</td>
<td>35</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Pseudocode For Arrow</td>
<td>36</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Pseudocode For Rectangle</td>
<td>37</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Pseudocode For Bubble</td>
<td>38</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Diagram for Points X(I+2), Y(I+2)</td>
<td>42</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Diagrams Of Arrow and Double-Headed Arrow</td>
<td>46</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Diagram Of Rectangle</td>
<td>49</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Diagram Of An Arc</td>
<td>51</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Diagram Of the Bubble</td>
<td>53</td>
</tr>
</tbody>
</table>
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 information 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 development of flow charts, hierarchical diagrams, or illustration layout. Without the additions, the user must write programs to create objects which are needed. The time needed to develop, code and debug such a program might be substantial; particularly if the user is unfamiliar 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).

E. 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 variables 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 2 ARROW
Figure 3 Double-Headed Arrow
FIGURE 4 RECTANGLE
FIGURE 5 BUBBLE

BUBBLE

UNFILLED BUBBLE
C. EVALUATION

EGF is not a self contained program, nor is it an integrated graphic package by itself. Rather, it provides additional primitives for the programmer to use in conjunction 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 program is too large to be loaded and run in the default size of the Basic memory. Therefore, it is segmented into two parts: EGF1 contains codes necessary to draw the vector, the arrow and double-headed arrow while EGF2 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 requires that the complete picture be redrawn at each change—an inconvenience, since frequently 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
dimensional 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. Although 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 GOSUES statements. The lack of parameter passing in a subroutine call makes it necessary to write duplicate plotting routines to accomplish similar tasks. The recognition of only the first two characters as an identifier is annoying because an identifier cannot 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 poor program structuring. I would like to see this program translated into another language, perhaps Pascal, which I consider to have outstanding clarity, extensive control 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 some 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 specifications. 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC Attribute Value</td>
<td>1 to 7</td>
<td>Background color (see color table, figure 6).</td>
</tr>
<tr>
<td>FC Attribute Value</td>
<td>1 to 7</td>
<td>Foreground color. Note: BC can be set to any value in the range, but FC must be chosen with care; some values will not work.</td>
</tr>
<tr>
<td>COLOR CODE</td>
<td>COLOR</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>BLACK</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BLUE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>GREEN</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CYAN</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>RED</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MAGENTA</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>YELLOW</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>WHITE</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6 Color Table**
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DO$</strong></td>
<td>Dotted, string variable. Value: D for dotted mode, XD for undotted mode.</td>
</tr>
<tr>
<td><strong>P1$</strong></td>
<td>String variable. Attribute: Character. Value: F or XF. Note: F for filled mode, XF for unfilled mode.</td>
</tr>
<tr>
<td><strong>B1$</strong></td>
<td>String variable. Attribute: Character. Value: B or XB. Note: B for blink mode, XB for unblink mode.</td>
</tr>
</tbody>
</table>

**Input Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(XS,YS)</td>
<td>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.</td>
</tr>
<tr>
<td>(XN,YN)</td>
<td>Ending coordinate of an object. Attribute: Integer, single precision. Value: Depends on the user's input, limited by the number of dots on the CRT screen.</td>
</tr>
<tr>
<td>W</td>
<td>Width of a line segment. Attribute: Integer, single precision. Value: 2 to 20 for VEC; 1 to 6 for other commands.</td>
</tr>
<tr>
<td>HT</td>
<td>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.</td>
</tr>
</tbody>
</table>
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

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

SP

Simple variable, represents the spacing between the dotted line.

Attribute Value

K

Simple variable, represents the terminal point used in the For-Next loop.

Attribute Value

Parameter for Plotting Routine

(IS,IS)
(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
2,4 RETURN
D,F,XB RETURN
50,300 RETURN
145,450,6 RETURN

Invoke the vector routine.
Set FG and BG colors.
Set options.
Starting coordinate.
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 to 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

\[ X, Y, L2 \]
* Attribute
  * Simple variables.
  * Single precision reals.
  * Value
    * Vary, depending upon the results from calculations.

\[ SI\%, CI\% \]
* Attribute
  * Simple variables, represent sine and cosine of an angle.
  * Value
    * Double precision real numbers.
    * Vary, depending upon the result from calculations.

\[ AH\$ \]
* Attribute
  * Character (3).
  * Value
    * FIX or VAR.

\[ X(P), Y(P), X(P+I), Y(P+I) \]
* Where \( 0 \leq I \leq 6, P = 2 \).
  * These are array variables; 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

\[ XS, YS \]
\(XN\)
\(YN\)
\(X(P)\)
\(Y(P)\)
\(X(P+I)\)
\(Y(P+I)\)

Where \(0 \leq I \leq 6, \ p = 2.\)

Specifications of these variables have been discussed.

**EXAMPLE OF CALLING SEQUENCE**

<table>
<thead>
<tr>
<th>ARR</th>
<th>RETURN</th>
</tr>
</thead>
<tbody>
<tr>
<td>3,6</td>
<td>RETURN</td>
</tr>
<tr>
<td>D,F,XB</td>
<td>RETURN</td>
</tr>
<tr>
<td>115,170</td>
<td>RETURN</td>
</tr>
<tr>
<td>215,170,4</td>
<td>RETURN</td>
</tr>
<tr>
<td>VAR</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

Invoke the arrow routine.
Set BG and FG colors.
Set options.
Starting coordinate.
Ending coord. and width.
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

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

Parameters for plotting routine

$X_S$
$Y_S$
$X_N$
$Y_N$
$X_M$
$Y_M$
$X(P)$
$Y(P)$
$X(P+I)$
$Y(P+I)$

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) | Where 0<J<=7, I=1. |
| Y(I+J) | Single precision real numbers. |
| Attribute | 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**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REC</td>
<td>Invoke the rec. routine.</td>
</tr>
<tr>
<td>2,5</td>
<td>Set BG and FG colors.</td>
</tr>
<tr>
<td>XD,F,XB</td>
<td>Set options.</td>
</tr>
<tr>
<td>100,50</td>
<td>Starting coordinate.</td>
</tr>
<tr>
<td>145,400</td>
<td>Ending coordinate.</td>
</tr>
<tr>
<td>30,6</td>
<td>Height and width of rec.</td>
</tr>
</tbody>
</table>

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 chapter.

Local Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>Simple Variable.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Single precision real number.</td>
</tr>
<tr>
<td>Value</td>
<td>3.1416.</td>
</tr>
<tr>
<td>CV</td>
<td>Simple Variable.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Single precision real number.</td>
</tr>
<tr>
<td>Value</td>
<td>Depending upon the result of calculation.</td>
</tr>
<tr>
<td>A, B, L, R, R1, R2</td>
<td>Simple Variables.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Single precision real numbers.</td>
</tr>
<tr>
<td>Value</td>
<td>Depending upon the result of calculation.</td>
</tr>
<tr>
<td>X(I)</td>
<td>Array variables</td>
</tr>
<tr>
<td>Attribute</td>
<td>Single precision real numbers.</td>
</tr>
<tr>
<td>Value</td>
<td>Depending upon the result of calculation.</td>
</tr>
<tr>
<td>Y(I)</td>
<td>Array variables.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Single precision real numbers.</td>
</tr>
<tr>
<td>Value</td>
<td>Depending upon the result of calculation.</td>
</tr>
<tr>
<td>X(I), Y(I), X(J)</td>
<td>Simple variables, 0&lt;=I&lt;=4, 0&lt;=J&lt;=4.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Single precision real numbers.</td>
</tr>
<tr>
<td>Value</td>
<td>Depending upon results of calculation.</td>
</tr>
</tbody>
</table>
Z, Z(I)  
Attribute  Simple Variables, 0<\textit{I}<=9.
Value    Single precision real numbers.

T, T(I)  
Attribute  Simple variables, 0<\textit{I}<=3.
Value    Single precision real numbers.

Parameter for Plotting Subroutine

\textbf{I(I)}  
Where 0<\textit{I}<=4.

\textbf{Y(J)}  
Where 0<\textit{J}<=4.

\textbf{R(I)}  
Where 0<\textit{I}<=2.

\textbf{Z(I)}  
Where 0<\textit{I}<=9.

Specification of the above variables have been discussed.

EXAMPLE OF CALLING SEQUENCE

\begin{verbatim}
BUB 2,7 RETURN    \text{Invoke the bubble routine.}
XD, XF, XB 300, 250 RETURN    \text{Set FG and BG colors.}
400, 250 RETURN    \text{Set options.}
80, 4 RETURN      \text{Starting coordinate.}
   \text{Ending coordinate.}
   \text{Height of bubble and width.}
\end{verbatim}

The BUB command places the window in the bubble plot submode. Bubbles generated are in horizontal position only; they cannot 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 COMMAND
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,E,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 constructed. 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.
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.

![Hierarchical Chart]

**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 subroutine 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 requested 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 attributes 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 arbitrary number, which represents the maximum number of elements expected, is assigned to each of 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 AHS$, DOS$, FIS$, and BIS$

AHS$, DOS$, FIS$, and BIS$ are a set of string variables whose stored values dictate the appearance of the generated object. For example, if DOS$ contains the value D and the object currently being constructed is an arrow, then the final product projected 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 variables 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 variables, 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 horizontally, it resembles a parallelogram. To correct this deficiency, the variable SQ is introduced. It is used as a scale factor for 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 possesses 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 constituted without a parameter list and invoked without any argument. A detailed 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 labor, 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
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
  ENDOWHILE
ENDCYCLE

FIGURE 10  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
  ENDCYCLE
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 origins 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 polygonal area must be bounded and completely closed; the cursor must be centered inside the polygon, and the colors must be chosen correctly. To fulfill 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 \((X_S, Y_S)\), the ending coordinate \((X_N, Y_N)\), 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:

![Diagram of vector construction](image)

**Figure 13**
We can obtain these points from the following equations:

1) \[ X(I+1) = X_N + W \cdot \cos(\pi/2 + \theta) \]
2) \[ Y(I+1) = Y_N + W \cdot \sin(\pi/2 + \theta) \]
3) \[ X(I+2) = X_S + W \cdot \cos(\pi/2 + \theta) \]
4) \[ Y(I+2) = Y_S + W \cdot \sin(\pi/2 + \theta) \]

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) \[ \cos \theta = \frac{X_N - X_S}{\sqrt{(X_N - X_S)^2 + (Y_N - Y_S)^2}} \]

8) \[ \sin \theta = \frac{Y_N - Y_S}{\sqrt{(X_N - X_S)^2 + (Y_N - Y_S)^2}} \]

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

1') \[ X(I+1) = X_N - \bar{W} \left( \frac{Y_N - Y_S}{L} \right) \]
\[ 2') \quad Y(I+1) = YN + W \left( \frac{XN-XS}{L} \right) \]

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

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

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) \[ X(J+1) = X_0 + D \cos \left( \frac{X_0 - X_S}{L_1} \right) \]

2) \[ Y(J+1) = Y_0 + D \sin \left( \frac{Y_0 - Y_S}{L_1} \right) \]

Where

- \( X(J+1), Y(J+1) \) is the coordinate of a particular point.
- \( (X_0, Y_0) \) is the starting point of a given line segment.
- \( D \) is the distance from the starting point to the current point \( (X(J+1), Y(J+1)) \).
- \( \frac{X_0 - X_S}{L_1} \) represents the cosine of an angle.
- \( \frac{Y_0 - Y_S}{L_1} \) 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 subroutines. These two procedures are invoked by the driver to create arrows and double-headed arrows in a variety of appearances
contingent 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 to solve a series of equations 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:

![Diagram](image)

**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 horizontal 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:

\[
X(I) = XS + W \times CO\
Y(I) = YS + W \times SN\
X(I+1) = X(I) - W \times SN\
Y(I+1) = Y(I) + W \times CO\
X(I+2) = X(I) + (W-HT) \times SN\
X(I+3) = XS - HT \times SN\
Y(I+3) = YS + HT \times CO\
X(I+4) = XS + (L-W) \times CO\
Y(I+4) = YS + (L-W) \times SN\
X(I+5) = X(I+4) - W \times SN\
Y(I+5) = Y(I+4) + W \times CO\
X(I+6) = X(I+4) + (W-HT) \times SN\
Y(I+6) = Y(I+4) + (HT-W) \times CO\
X(I+7) = XN - HT \times SN\
Y(I+7) = YX + HT \times CO
\]

where

\(X(I+J), \ Y(I+J)\) represent the coordinates; \(0 \leq J \leq 7\).

(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.
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 following figure represents one of the loops of the bubble:
The slop of I is \((A-0)/(0-B)=-A/B\),
Therefore, slop of \(\gamma\) 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
\[Y_c=-(B**2/2*A)+(A/2)\ .\]
\[TANQ = |Y_c|/B, \ \text{so that}\]
\[Q = ATN \left( \frac{|Y_c|}{B} \right) \ \text{where} \ |Y_c| = \ (B**2/2*A)-A/2.\]
\(Q\) is the angle measurement in radians of the starting position of the arc.
\(\Pi - Q\) is the angle measurement in radians of the ending position of the arc.
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)
\[(2A+B, 10A-\sqrt{2R}).\]
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 activates the Plotting Routine which will retrieve these information
to draw the final figure.
BIBLIOGRAPHY


APPENDIX
*AUTHOR-MAXINE YEE

*PROGRAM-EGP(EXTENDED GRAPHIC PRIMITIVES)

*LANGUAGE-CHRONIATICS BASIC VERSION 3.0

*PLACE-KANSAS STATE UNIVERSITY, DEPT. OF COMPUTER SCIENCE

*DATE-DEC., 1979

' THIS IS A PROGRAM FOR THE GENERATION OF EXTENDED GRAPHIC PRIMITIVES

' "THIS IS A PROGRAM FOR THE GENERATION OF EXTENDED GRAPHIC PRIMITIVES"
PRINT"J";CHR$(&HC);   'VISIBLE CURSOR; ERASE PAGE
PRINT Chr$(27);"OA1";  'WINDOW 1
PRINT"W000511511490";  'SIZE (000, 511), (511, 490)
PRINT"N";"C1";"C7";  'BG BLUE, FG WHITE
PRINT Chr$(&H15);"J";"P";  'CHAR MODE; CURSOR ON; ROLL OFF
PRINT Chr$(&HC);  'ERASE PAGE
CLEAR
DIM X(200), Y(200)
PRINT Chr$(12);
INPUT"TYPE IN THE COMMAND";CMD$  'ENABLES COMMAND PROCESSING
PRINT Chr$(12);           'ERASE PAGE
INPUT"TYPE IN THE BACKGROUND AND FOREGROUND COLORS";BC$,FC  'COLORS
PRINT Chr$(12);           'ERASE PAGE
SQ=SQR(2)
INPUT"TYPE IN THE ATTRIBUTES FOR THE OBJECT";DO$,FI$,BL$
IF BL$="B" THEN PRINT"1"; ELSE PRINT "2"
IF CMD$="VEC" THEN GOSUB 1500: GOTO 55
IF CMD$="ARR" THEN GOSUB 5000: GOTO 55
IF CMD$="DAR" THEN GOSUB 10000: GOTO 55
IF CMD$="REC" THEN GOSUB 20000: GOTO 55
IF CMD$="BUB" THEN GOSUB 30000: GOTO 55
PRINT "SORRY, WRONG COMMAND"; GOTO 55
THIS IS A SUBROUTINE FOR DRAWING DOTTED LINES

ASSIGN SOLID SEGMENT OF THE DOTTED LINE

ASSIGN SPACE OF THE DOTTED LINE

CAL LENGTH OF GIVEN LINE

CAL COSINE OF AN ANGLE

CAL SINE OF AN ANGLE

CAL TERMINAL VALUE FOR THE LOOP

CAL X(I) COORD

CAL Y(I) COORD

THIS SECTION OF CODE TESTS THE VARIOUS END POINTS OF A GIVEN LINE

IF E=0 THEN IF (L1-D) < SEG
          THEN D=D+(L1-D):NEXT J
          ELSE D=D+SEG:NEXT J
          ELSE GOSUB 54000

SUB FOR DRAWING DOTTED UNFILLED LINE

REMAINING SEG TOO SHORT TO PLOT

REMAINING SEG TOO SHORT TO PLOT

D = D + SP

IF E=1 AND L1-D<=SP AND CMD$="VEC" THEN GOSUB 55500:GOTO 1510

NEXT J

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" 
5030 GOSUB 45000 
5040 INPUT"TYPE IN VAR FOR VARIABLE ARROW HEAD OR FIX FOR FIXED HEAD";AH$ 
5050 PRINT CHR$(12); 'ERASE PAGE 
5060 YS=YS/SQ:YN=YN/SQ 
5100 X = XN-XS 
5110 Y=YN-YS 
5120 L2= SQR((XN-XS)^2 + (YN-YS)^2) 'CAL LENGTH OF GIVEN LINE 
5130 W1= W/2 'CAL WIDTH OF TAIL 
5140 SI# = Y/L2 'CAL SINE OF AN ANGLE 
5150 CS# = X/L2 'CAL COSINE OF AN ANGLE 
5160 IF AH$ = "FIX" AND W1<1 THEN GOSUB 5410: GOTO 5260 
5165 IF AH$ = "FIX" AND W1 >1 THEN GOSUB 5410 
5170 IF AH$="FIX" AND DO$="D" THEN 5290 
5170 '
5180 ' THIS SECTION OF CODE CALCULATES THE VARIABLE SIZE OF AN ARROW HEAD
5190  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#  
5360  Y(P+6) = XS + W1*SI#
5370  X(P+6) = YS - W1*CS#  
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 '  

5505 GOSUB 55550:GOTO 5400
5420 'THIS SUBROUTINE CALCULATES FIXED ARROW HEAD
5430  H = 10 'ASSIGN LENGTH FOR THE ARROW HEAD
5440  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 D0<$>"D" THEN 5260
5506  IF AH$="F" THEN 5290
5507  IF WI>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$="F" THEN XH$=(X(P)+XN)/2:YH$=(Y(P)+YN)/2:GOSUB 55000
5710  RETURN
5720 '
5730 '
5930 'THIS IS A PLOTTING SUBROUTINE FOR WIDE ARROW
5950  PLOT X(P+2),Y(P+2),X(P+1),Y(P+1)
5960  PLOT X(P+1),X(P+1),XN,YN
5970  PLOT XN,YN,X(P+4),Y(P+4)
5980  PLOT X(P+4),Y(P+4),X(P+3),Y(P+3)
5990  PLOT X(P+3),Y(P+3),X(P+6),Y(P+6)
6000  PLOT X(P+6),Y(P+6),X(P+5),Y(P+5)
6010  PLOT X(P+5),Y(P+5),X(P+2),Y(P+2)
6020  RETURN
6100 ' PAGE 61
6101 ' THIS IS A SUBROUTINE FOR DRAWING DOTTED WIDE ARROWS
6110 XN=X(P+3):YN=Y(P+3)
6130 XS=X(P+6):YS=Y(P+6)
6131 J=1
6140 XB=X(P+6):YB=Y(P+6)
6142 X(J)=X(P+6):Y(J)=Y(P+6)
6150 RETURN
10000 '  
10001  '***************************************************************************************************
10002  ' THIS IS A SUBROUTINE FOR DRAWING DOUBLE HEADED ARROWS
10003 '***************************************************************************************************
10010 PRINT "THIS IS A SUBROUTINE FOR DRAWING DOUBLE HEADED ARROWS"
10020 GOSUB 45000
10040 INPUT "TYPE IN VAR FOR VARIABLE ARROW HEAD OR FIX FOR FIXED ARROW HEAD";AH$
10050 PRINT CHR$(12); 
10060 X=XN-XS
10070 Y=YN-YS
10080 L3=SQR((XN-XS)^2 +(YN-YS)^2) 'CAL LENGTH OF GIVEN LINE
10090 W=W/2
10100 SIN=Y/L3 'CAL SINE
10110 COS=X/L3 'CAL COSINE
10120 IF AH$="FIX" AND W<=1 THEN GOSUB 10500:GOSUB 54000:GOSUB 10900:GOSUB 55500:GOTO 10020
10125 IF AH$="FIX" AND W>1 THEN GOSUB 10500:GOTO 10300
10126 '
10127  
10130  'THIS SECTION OF CODES CALCULATES THE VARIABLE SIZE OF THE ARROW HEADS  
10140  '  
10150  P=2  
10160  X(P)=XS+9/10*X  
10170  Y(P)=YS+9/10*Y  
10180  X(P+1)=X(P)-Y/10  
10190  Y(P+1)=Y(P)+X/10  
10200  X(P+4)=X(P)+Y/10  
10210  Y(P+4)=Y(P)-X/10  
10220  X(P+5)=XS+X/10  
10230  Y(P+5)=YS+Y/10  
10240  X(P+6)=X(P+5)-Y/10  
10250  Y(P+6)=Y(P+5)+X/10  
10260  X(P+9)=X(P+5)+Y/10  
10270  Y(P+9)=Y(P+5)-X/10  
10280  IF DO$="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:GOTO 10020  
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$="D" THEN GOSUB 54000:GOSUB 10940:GOSUB 10300  
10670  RETURN  
10690  '  
10920  'THIS IS A PLOTTING SUBROUTINE FOR THE DOUBLE HEADED ARROWS  
10930  PLOT X(P+5),Y(P+5),X(P),Y(P)  
10940  PLOT X(P+1),Y(P+1),XN,YN  
10950  PLOT XN,YN,X(P+4),Y(P+4)  
10960  PLOT X(P+4),Y(P+4),X(P+1),Y(P+1)  
10970  PLOT X(P+6),Y(P+6),X(P+9),Y(P+9)  
10980  PLOT X(P+9),Y(P+9),XS,YS  
10990  PLOT XS,YS,X(P+6),Y(P+6)  
11000  IF FI$="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$=(Y(P+5)+YS)/2:GOSUB 55000  
11020  RETURN  
11100  '
11110 ' THIS IS A PLOTTING SUBROUTINE FOR WIDE DOUBLE HEADED ARROWS
11120  PLOT X(P+2), Y(P+2), X(P+1), Y(P+1)
11130  PLOT X(P+1), Y(P+1), XN, YN
11140  PLOT XN, YN, X(P+4), Y(P+4)
11150  PLOT X(P+4), Y(P+4), X(P+3), Y(P+3)
11160  PLOT X(P+3), Y(P+3), X(P+8), Y(P+8)
11170  PLOT X(P+8), X(P+8), X(P+9), Y(P+9)
11180  PLOT X(P+9), Y(P+9), XS, YS
11190  PLOT XS, YS, X(P+6), Y(P+6)
11200  PLOT X(P+6), Y(P+6), X(P+7), Y(P+7)
11210  PLOT X(P+7), Y(P+7), X(P+2), Y(P+2)
11220  IF F1$="F" THEN XM#=(X(P)+XN)/2:YM#=(Y(P)+YN)/2:GOSUB 55000
11230  RETURN
11240 '
11250 ' THIS SECTION OF CODE PLOTS DOTTED LINE FOR THE DOUBLE HEADED ARROWS
11260  IF W1<1 THEN 11500
11270  XS=X(P+8):YS=Y(P+8)
11280  XN=X(P+3):YN=Y(P+3)
11290  XB=X(P+8):YB=Y(P+8)
11300  J=1:X(J)=X(P+8):Y(J)=Y(P+8)
11310  RETURN
11320 '
11330 ' THIS SECTION OF CODE PLOTS SINGLE DOTTED LINE FOR THE ARROW HEADS
11340  XS=X(P+5):YS=Y(P+5)
11350  XN=X(P):YN=Y(P)
11360  GOSUB 2200:GOSUB 55500:GOTO 10020
11370 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) 
20130 CO# = X/L 
20140 SN# = Y/L 
20150 'CAL LENGTH OF GIVEN LINE 
20160 'CAL SIN OF AN ANGLE 
20170 ' 
20180 'FOLLOWING SECTION OF CODE CALCULATES THE VARIOUS COORD. OF THE RECTANGLE 
20190 ' 
20200 I=1 
20210 X(I)=XS+W*CO# 
20220 Y(I)=YS+W*SN# 
20230 X(I+1)=X(I)-W*SN# 
20240 Y(I+1)=Y(I)+W*CO# 
20250 X(I+2)=X(I)+(W-HT)*SN# 
20260 Y(I+2)=Y(I)+(HT-W)*CO# 
20270 X(I+3)=X(I)-HT*SN# 
20280 Y(I+3)=Y(I)+HT*CO# 
20290 X(I+4)=X(I)+(L-W)*CO# 
20300 Y(I+4)=Y(I)+(L-W)*SN# 
20310 X(I+5)=X(I+4)-W*SN# 
20320 Y(I+5)=Y(I+4)+W*CO# 
20330 X(I+6)=X(I+5)+(W-HT)*SN# 
20340 Y(I+6)=Y(I+5)+(HT-W)*CO# 
20350 GOSUB 54000:GOSUB 20700:GOSUB 55500:GOTO 20070 
20360 RETURN
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 ROUTINE FOR DRAWING BUBBLES
30040 '
30050 '********************************************************************************
30060 '
30070 GOSUB 45500
30080 IF XS=999 THEN RETURN
30090 PI=3.1416
30100 CV=180/PI
30110 SQ=SQR(2)
30110 L=XN-XS
30120 IF L<CHT THEN 35000
30130 A=HT/10
30140 A1=A/SQ
30150 B=(L-4*A)/6
30160 R=A1/2+(B^2)/(2*A1)
30170 T=ATN((B^2/2*A1)-(A1/2))/B
30180 I=1
30190 X(I)=XS+(2*A+B)
30200 X(I+1)=X(I)+2*B
30210 X(I+2)=X(I+1)+2*B
30220 Y(I)=YS+(10*A-SQ*R)
30230 Y(I+3)=YS+(SQ*R)
30240 T1=T*CV

'CALC LENGTH OF GIVEN LINE
'CALC WIDTH OF THE BUBBLE
'PUT IN SCALE FACTOR
'CALC ARC DISTANCE
'CALC RADIUS OF THE FOUR ARCS
'CALC STARTING ANGLE FOR THE ARC
'CALC X COORDS OF THE FOUR ARC
'CALC Y COORDS OF THE OUTER ARC
'CALC THE VARIOUS ANGLES FOR THE FOUR
'ARC AND CONVERTING THEM FROM RAD TO DEG
30240 \( T_2 = (\pi - 2^* T)^{\#C} \)
30250 \( T_3 = (\pi + T)^{\#C} \)
30260 \( R_2 = 3^{\#A} \)
30270 \( X_2 = X_S + R_2 \)
30280 \( Y_2 = Y_S + 5^{\#A} \)
30290 \( X_4 = X_N - R_2 \)
30300 \( Y_4 = Y_2 \)
30310 \( Z = \text{ATN}(4^{\#S}/\text{SQ}) \)
30320 \( Z_1 = 180 - Z^{\#C} \)
30330 \( Z_2 = 2^{\#Z^{\#C}} \)
30340 \( Z_3 = 360 - Z^{\#C} \)
30350 \( Z_4 = 2^{\#Z^{\#C}} \)
30360 \( \text{IF } W \leq 1 \text{ THEN GOSUB 54000: GOSUB 31000: GOSUB 33000: GOSUB 55500: GOTO 30070} \)
30370 \( R_1 = A^{\#SQR}(8.5) \)
30380 \( Y(I+1) = Y(I) + A \)
30390 \( Y(I+2) = Y(I+3) + A \)
30400 \( X_1 = X_2 + A \)
30410 \( Y_1 = Y_2 \)
30420 \( X_3 = X_N - A - R_1 \)
30430 \( Y_3 = Y_2 \)
30440 \( Z_5 = \text{ATN}(3/(2^{\#S} \text{SQ})) \)
30450 \( Z_6 = 180 - Z_5^{\#C} \)
30460 \( Z_7 = 2^{\#Z^{\#C}} \)
30470 \( Z_8 = 360 - Z_5^{\#C} \)
30480 \( Z_9 = 2^{\#Z^{\#C}} \)
30490 \( \text{GOSUB 54000: GOSUB 31000: GOSUB 32000: GOSUB 33000: GOSUB 33200} \)
30510 \( \text{GOSUB 55500 GOTO 30070} \)
31000 ' THIS IS A SUBROUTINE FOR THE GENERATION OF THE OUTER THREE ARCS
31020 ' FOR J = 1 TO 4 STEP 3
31030 FOR I = 1 TO 3 STEP 1
31040 IF J = 1 THEN PLOT X(I), Y(J), R, T_1, T_2
31050 ELSE PLOT X(I), Y(J), R, T_3, T_2
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=1 TO 3 STEP 1 
32050         IF J=2 THEN PLOT X(I),Y(J),R,T1,T2 
32060         ELSE PLOT X(I),Y(J),R,T3,T2 
32070     NEXT I 
32080 NEXT J 
32090 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 
33050 ' 
33060 'THIS IS A PLOTTING SUBROUTINE FOR THE GENERATION OF LT AND RT ARC 
33070 PLOT X1,Y1,R1,Z6,Z7 
33080 PLOT X3,Y3,R1,Z8,Z9 
33090 RETURN 
34000 ' 
34010 ' 
34020 'THIS SUBROUTINE HANDLES THE INPUT OF COORD 
34030 I=1 
34040 INPUT"TYPE IN FIRST COORD FOR YOUR OBJECT";XS,YS 
34050 INPUT"TYPE IN SECOND COORD AND WIDTH FOR YOUR OBJECT";XN,YN,W 
34060 RETURN 
34070 ' 
34080 'THIS SUBROUTINE HANDLES THE INPUT OF COORD FOR RECTANGLES AND BUBBLES 
34090 ' 
34100 ' 
34110 ' 
34120 ' 
34130 ' 
34140 ' 
34150 ' 
34160 ' 
34170 ' 
34180 ' 
34190 ' 
34200 ' 
34210 ' 
34220 ' 
34230 ' 
34240 ' 
34250 ' 
34260 ' 
34270 ' 
34280 ' 
34290 ' 
34300 ' 
34310 ' 
34320 ' 
34330 ' 
34340 ' 
34350 ' 
34360 ' 
34370 ' 
34380 ' 
34390 ' 
34400 ' 
34410 ' 
34420 ' 
34430 ' 
34440 ' 
34450 ' 
34460 ' 
34470 ' 
34480 ' 
34490 ' 
34500 ' 
34510 ' 
34520 ' 
34530 INPUT"TYPE IN FIRST COORD FOR YOUR OBJECT";XS,YS 
34540 INPUT"TYPE IN SECOND COORD FOR YOUR OBJECT";XN,YN 
34550 INPUT"TYPE IN THE HEIGHT AND WIDTH OF YOUR OBJECT";HT,W 
34560 RETURN
54000 ' 
54010 ' 
54020 'THIS IS A SUBROUTINE TO HANDLE THE PLOTTING ENVIRONMENT AND SET COLOR 
54030 PRINT CHR$(1);"M"; 'BACKGROUND LIGHT ON  
54040 PRINT CHR$(1);"C";CHR$(48+BC); 'SET BACKGROUND COLOR  
54050 PRINT CHR$(1);"N"; 'BACKGROUND LIGHT OFF  
54060 PRINT CHR$(1);"C";CHR$(48+FC); 'SET FOREGROUND COLOR 
54070 PRINT CHR$(1);"G"; 'PLOT ON  
54080 IF CMD$="SUB" THEN PRINT ")" ELSE PRINT ""; 
54090 RETURN  
54400 ' 
54510 ' 
54520 'THIS IS A SUBROUTINE TO HANDLE THE PLOTTING ENVIRONMENT  
54530 PRINT ";G"; 'PLOT ON  
54540 PRINT ";"; 'VECTOR SUBMODE 
54550 RETURN  
55000 ' 
55010 ' 
55020 'THIS SUBROUTINE COMPLEX FILLS AN OBJECT  
55030 PRINT ";U"; : PLOT XM#,YM# 'MOVE CURSOR TO COORD  
55040 PRINT ";J"; 
55050 PRINT ";M"; 'BACKGROUND LIGHT ON  
55060 PRINT ";C";CHR$(48+BC); 'SET BACKGROUND COLOR  
55070 PRINT ";N"; 'BACKGROUND LIGHT OFF  
55080 PRINT ";C";CHR$(48+FC); 'SET FOREGROUND COLOR  
55090 PRINT ";>"; CHR$(32); 'FILL OBJECT WITH SOLID COLOR  
55100 RETURN  
55500 ' 
55510 ' 
55520 'THIS SUBROUTINE CANCELS THE PLOTTING ENVIRONMENT  
55530 PRINT CHR$(21); 'CANCELS PLOT MODE  
55540 PRINT CHR$(28); 'CURSOR HOME  
55550 RETURN  
55560 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

EAGP (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 debug 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.