

AVON

CAMELOT

\$3.95

05068

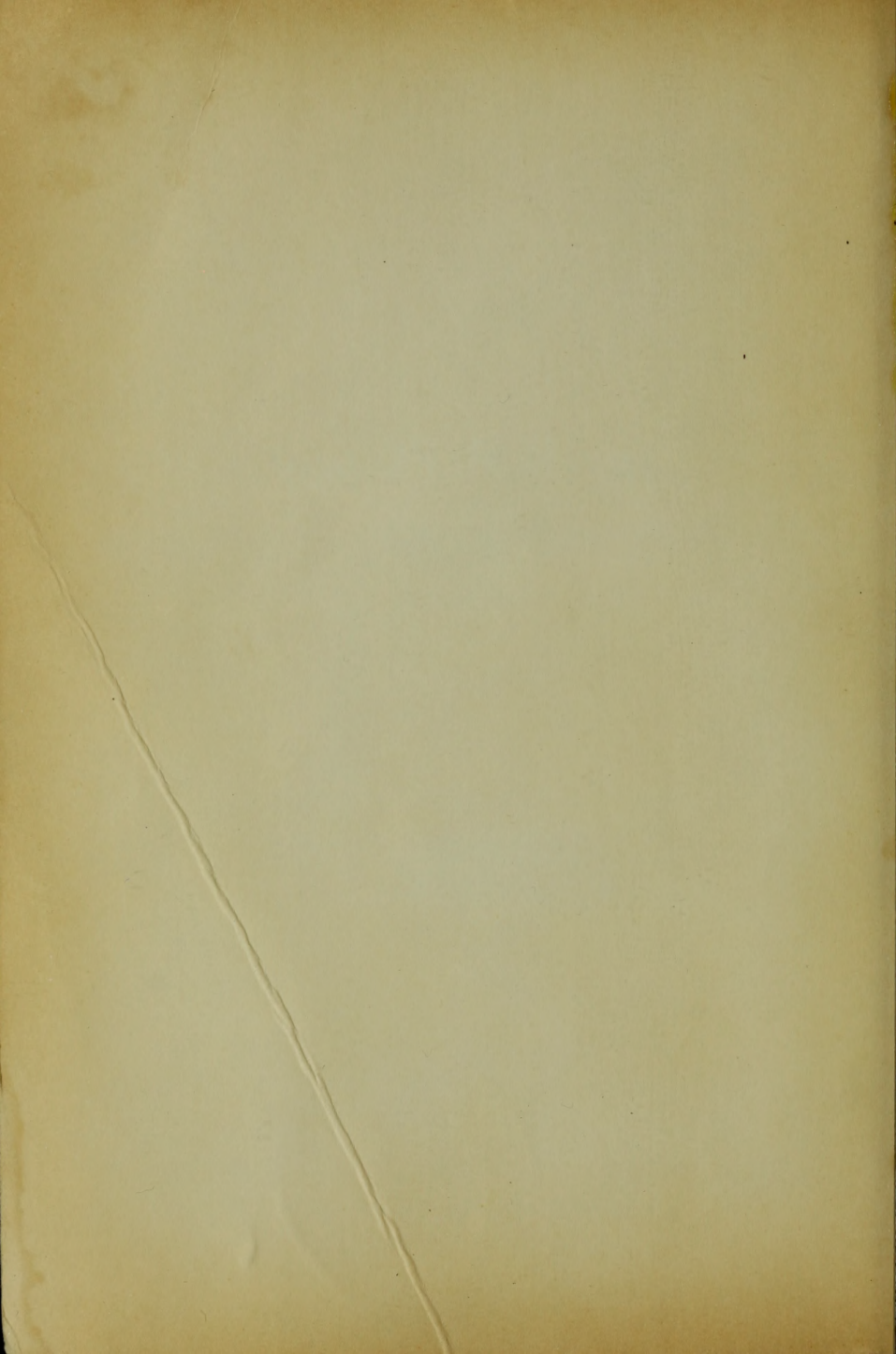
A guide to computer graphics

BASIC FUN WITH GRAPHICS

THE **IBM/PC**[®] WAY
COMPUTER



BY MARGARET ANN ZUANICH
AND SUSAN DRAKE LIPSCOMB



1 —

BASIC FUN WITH GRAPHICS

THE

IBM/PC[®]			
C	O	M	P
U	T	E	R

WAY

Other Avon Camelot Books by
Margaret Ann Zuanich and
Susan Drake Lipscomb

BASIC BEGINNINGS

BASIC FUN: COMPUTER GAMES, PUZZLES AND
PROBLEMS CHILDREN CAN WRITE

BASIC FUN WITH GRAPHICS: THE APPLE® COMPUTER WAY

BASIC FUN WITH GRAPHICS: THE ATARI® COMPUTER WAY

MARGARET ANN ZUANICH and SUSAN DRAKE LIPSCOMB have a unique combination of skills that contributed to the creation of BASIC FUN WITH GRAPHICS. Margaret Zuanich's experience in the computer field has included everything from programming to management consulting. She earned her Master's Degree in Business and is now involved in computer systems training. Susan Lipscomb holds a Master's Degree in Education and has spent fourteen years in the area of language and learning disabilities. They both live in Palo Alto, California.

Avon Books are available at special quantity discounts for bulk purchases for sales promotions, premiums, fund raising or educational use. Special books, or book excerpts, can also be created to fit specific needs.

For details write or telephone the office of the Director of Special Markets, Avon Books, Dept. FP, 1790 Broadway, New York, New York 10019, 212-399-1357.

BASIC FUN WITH GRAPHICS

THE

IBM/PC[®]
C O M P U T E R

WAY

**MARGARET ANN ZUANICH
AND SUSAN DRAKE LIPSCOMB**

AN AVON



CAMELOT BOOK

5th grade reading level has been determined by using the Fry Readability Scale.

BASIC FUN WITH GRAPHICS: THE IBM/PC®COMPUTER WAY is an original publication of Avon Books. This work has never before appeared in book form.

AVON BOOKS

A division of
The Hearst Corporation
1790 Broadway
New York, New York 10019

Copyright © 1983 by Margaret Ann Zuanich and Susan Drake Lipscomb

Front cover photograph shows an IBM Personal Computer,
Courtesy of International Business Machines Corp.

Cover photograph by Aram Gesar

Published by arrangement with the authors

Library of Congress Catalog Card Number: 83-17183

ISBN: 0-380-85068-0

All rights reserved, which includes the right to
reproduce this book or portions thereof in any form
whatsoever except as provided by the U. S. Copyright Law.
For information address Avon Books.

Library of Congress Cataloging in Publication Data

Zuanich, Margaret Ann.

BASIC fun with graphics.

(An Avon/Camelot book)

Summary: An introduction to graphics programming
including exercises showing how the graphics statements
on the IBM/PC computer work and simple BASIC programs
that demonstrate ways to use these statements.

1. Computer graphics—Juvenile literature. 2. IBM Personal Computer—
Programming—Juvenile literature.

3. Basic (Computer program language)—Juvenile literature.

[1. Computer graphics. 2. IBM Personal Computer—Pro-
gramming (Computers)] I. Lipscomb, Susan Drake.

II. Title. III. Title: B.A.S.I.C. fun with graphics.

T385.Z83 1983

001.64'43

83-17183

ISBN 0-380-85068-0

First Camelot Printing, November, 1983

CAMELOT TRADEMARK REG. U. S. PAT. OFF. AND IN
OTHER COUNTRIES, MARCA REGISTRADA, HECHO EN
U. S. A.

Printed in the U. S. A.

DON 10 9 8 7 6 5 4 3 2 1

TABLE OF CONTENTS

Introduction 7

How to Use *BASIC Fun With Graphics:*
The IBM/PC Way 9

1 PRINT 13

1. Warmups 14
2. House 16
3. Trees 18
4. Ocean Liner 20
5. Space Fleet 21
6. Bugs 23
7. Rocket Ship 25
8. Bicycle 28

2 LOCATE 29

1. Warmups 30
2. Face 32
3. Christmas Tree 34
4. Candy Jar 36
5. Tower 38
6. Suburb 39
7. School 40
8. Hide-and-Seek 42

3 LINE and PSET 45

1. Warmups 46
2. Christmas Lights 51
3. Jet Plane 53
4. House 54
5. Flag 55
6. 3-D Room 57
7. Train 58
8. Skier 59

4 DRAW and CIRCLE 61

1. Warmups 62
2. Fish 75
3. Car 76
4. Fishing Boat 77
5. Windmill 78
6. Snowflakes 79
7. Sailboat 81
8. Campground 82
9. Picture 84

5 PUT and GET 87

1. Warmups 88
2. Apple Tree 97
3. Walking Figure 99
4. Flying Bird 101
5. Jumping Fish 103
6. Steam Engine 105

6 COLOR and PAINT 109

1. Warmups 110
2. Helicopter 119
3. Moonlight Sail 123
4. Evening Campground 125
5. Orchard Scene 127
6. Painted Picture 129

7 FUN with GEOMETRY 131

1. Warmups 132
2. Apple Tree Harvest 139
3. Leaping Fish 142
4. Bird and Flower 144
5. Fishing Boat 147
6. Fancy Graphics 149
7. Ski Game 151

Glossary 155

INTRODUCTION

Are you ready for the fun and challenge of graphics programming? Do you want to use your computer to create your own pictures and designs, move objects around on the screen, or design your own video game? *BASIC Fun With Graphics: The IBM/PC Way* gets you started, using the graphics programming statements on your computer. Each chapter includes exercises that show how the graphics statements work and simple BASIC programs that demonstrate exciting and fun ways to use these statements.

BASIC Fun With Graphics: The IBM/PC Way presents a systematic yet entertaining approach to learning graphics programming. First you type in the exercises and programs exactly as they appear in the book. Then you experiment and change them using your own ideas. Finally, you create your own designs and pictures, using those in the book as a guide.

BASIC Fun With Graphics: The IBM/PC Way covers the beginning and intermediate graphics statements available in BASICA, as well as a chapter for "hotshots" that uses higher-level mathematical functions to rotate geometric figures and generate new and unique designs. It stops short of commands that directly address memory, like PEEK and POKE statements, and shape tables used in sophisticated graphics and animation. When you're ready for this level of programming, your computer's BASIC reference guide has a comprehensive discussion of how to proceed.

BASIC Fun With Graphics: The IBM/PC Way was written on an IBM/PC equipped with a color/graphics board in BASICA on IBM's DOS 1.1. The programs will run under DOS 1.0 and 2.0 but do not include the BASICA graphic statement extensions and the RANDOMIZE TIMER statement available on the 2.0 release. For these programming supple-

ments, refer to the IBM BASIC manual 1.1 version with 2.0 updates, or the 2.0 release. The programs will run on either a color or monochrome monitor. For visual appeal, COLOR statements are included in all the programs. To run on a monochrome or black and white display, omit the COLOR statement from Line 100 in each program.

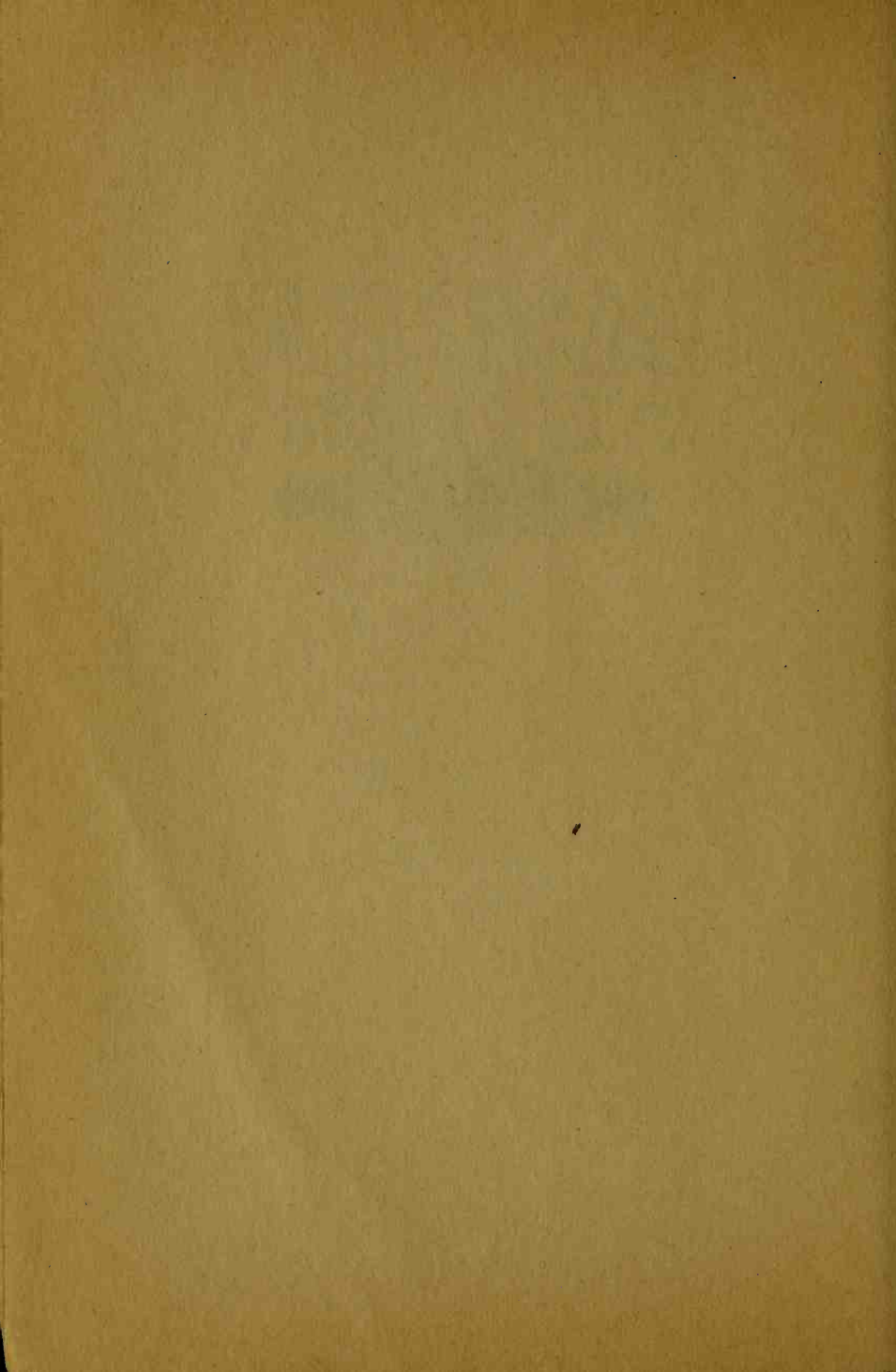
We found that experimenting with the graphics statements and creating designs and pictures on the screen was great fun. We hope you get as many "oohs," "aaahs," and chuckles as we have.

HOW TO USE BASIC FUN WITH GRAPHICS: The IBM/PC Way

1. Start up your IBM PC with DOS and type BASICA.
2. Type in each program, line by line, including the line numbers, exactly as shown in the Program Listing. Press the Return key after each line.
3. Once you have a program entered onto your computer, type RUN and press Return. The computer will RUN your program and display the results on the screen. It should look like the example in the book.
4. To check the programming statements, type LIST and press Return. The computer will list your program.
5. Now try making the changes suggested in the text, or experiment with other variations, using our program as a guide.
6. When you're ready to save your program, use the SAVE command. To retrieve a program from the disk, use the LOAD command.
7. Before you enter another program, type NEW to erase anything currently in memory.
8. The Warmup exercises at the beginning of most chapters contain directions and examples using each graphics statement. Read these and work through the exercises before you proceed to the programs.
9. The graphics programs are written in SCREEN 1, medium resolution graphics, as this mode gives you access to 16 colors. High-resolution mode, SCREEN 2, has only 2 colors and does not use the COLOR statement.

*BASIC FUN
WITH GRAPHICS*

THE **IBM/PC[®]** *WAY*
COMPUTER



1 PRINT

Use this program statement to tell the computer what to print on the screen or paper.

PROGRAMS

1. Warmups
2. House
3. Trees
4. Ocean Liner
5. Space Fleet
6. Bugs
7. Rocket Ship
8. Bicycle

WARMUPS

The format for the PRINT programming statement is:

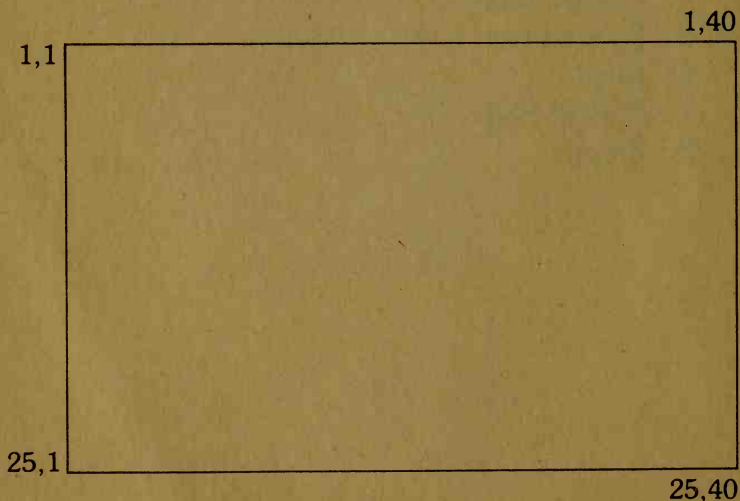
PRINT " "

You can use any combination of letters, numbers and, special characters in a PRINT statement.

Your computer has three display modes: Text, Medium Resolution, and High Resolution. This chapter shows you how to use the Text mode to draw shapes and objects.

In Text mode, you have 25 rows and 40 or 80 columns. The rows are numbered from top to bottom, the columns from left to right, on your screen. If you think of your screen as a piece of graph paper, then the top left square is 1,1 and the bottom right square is 25,40 or 25,80.

Screen Grid, Text Mode



Row 25 is used by the IBM "soft key" display. If you use the KEY OFF command, you can print on row 25. However, you will need to

clear the screen for subsequent program Runs to remove the line from the screen.

The WIDTH statement is used to set the number of columns. The format is:

WIDTH 40: specifies 40 columns

WIDTH 80: specifies 80 columns

The SCREEN command sets the display type. The format is:

SCREEN M, burst, apage, vpage

where: M: determines the display type, as follows:

0 Text mode

1 Medium Resolution mode

2 High Resolution mode

burst: enables color (0 or 1)

apage: active page (0 to 7)—Text mode only

vpage: visual page (0 to 7)—Text mode only

SCREEN can be used either in a program or as an immediate command. Any parameters can be omitted. In this book we have established the convention of setting only the display type, without additional parameters, as the first statement in each program.

HOUSE

Use PRINT to draw a house. Put this program in your computer by typing each line exactly as shown in the Program Listing.

Once you have this program up and running, try adding a chimney by using the IBM PC's edit keys (↑, ↓, ←, →, INS, DEL).

```

      /\
     /\
    /\
   /\
  /\
 /
I  I_ I  I_ I      I      I
I      I      I      I
I      I      I      I      I_ I      I
I      I I      I      I      I
I_ I      I I      I      I
□
  
```

PROGRAM LISTING

```

100 SCREEN 0:KEY OFF:COLOR 15,1:CLS
110 PRINT "      /\
120 PRINT "    /
130 PRINT "  /
140 PRINT " /
150 PRINT "I  I_ I  I_ I      I      I"
160 PRINT "I      I      I      I"
170 PRINT "I      I      I      I_ I      I"
180 PRINT "I      I I      I      I      I"
190 PRINT "I_ I      I I      I      I
  
```

NOTES

Line 100: SCREEN 0 puts the program run in Text mode. KEY OFF removes IBM BASIC's "Soft Key" display statement from row 25 of

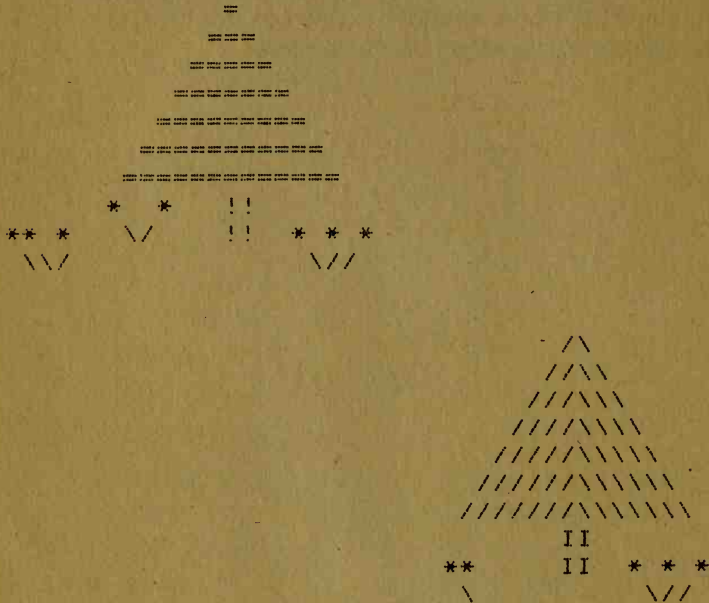
the screen. Omit the COLOR statement unless you have a color monitor. COLOR 15,1 sets the foreground color (the character) to white and the background color to blue. No border color is specified, so it defaults to the last one executed. COLOR is discussed in detail in Chapter 6. CLS clears the screen.

Lines 110-190: When the program runs, the PRINT statement tells the computer to print out everything between the two “ marks on each line, exactly as it is shown in the program.

TREES

Draw some trees and flowers. To make room for the trees, this program uses WIDTH 80.

See if you can add a bird or other trees.



PROGRAM LISTING

```

100 SCREEN 0:KEY OFF:COLOR 10,1,1:CLS
110 WIDTH 80
120 PRINT "
130 PRINT "
140 PRINT "
150 PRINT "
160 PRINT "
170 PRINT "
180 PRINT "
190 PRINT "
200 PRINT "
210 PRINT "
220 PRINT "

```

```

230 PRINT
240 PRINT "
250 PRINT "
260 PRINT "
270 PRINT "
280 PRINT "
290 PRINT "
300 PRINT "
310 PRINT "
320 PRINT "
330 PRINT "

```

```

      /\ "
     /\ /\ "
    /\ /\ /\ "
   /\ /\ /\ /\ "
  /\ /\ /\ /\ /\ "
 /\ /\ /\ /\ /\ /\ "
/\ /\ /\ /\ /\ /\ /\ "
      II "
    **   II   *** "
      \       \// "

```

NOTE

Line 110: WIDTH 80 changes the number of columns across the screen to 80.

OCEAN LINER

Type WIDTH 40 to get back to 40 columns. Draw an ocean liner with this program.

Now add some portholes or another deck to make your own ship.



PROGRAM LISTING

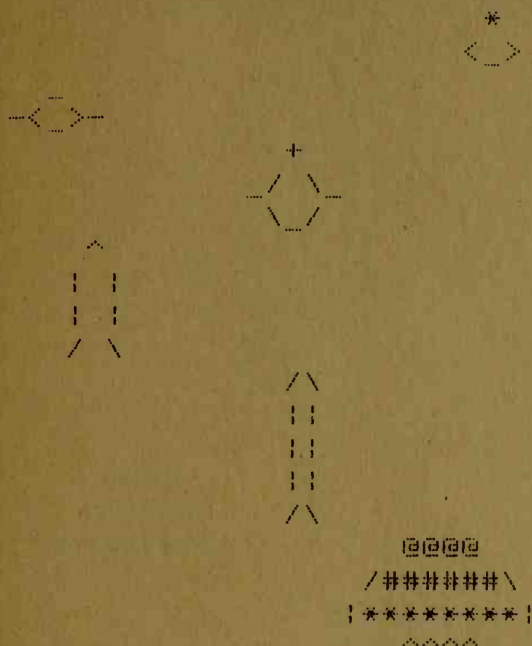
```

100 SCREEN 0:KEY OFF:COLOR 15,3,3:CLS
110 PRINT "
120 PRINT "
130 PRINT "
140 PRINT "
150 PRINT "
160 PRINT "

```

SPACE FLEET

Create a fleet of different kinds of spacecraft.



PROGRAM LISTING

```
100 SCREEN 0:KEY OFF:COLOR 14,1,1:CLS
110 PRINT "                                     *"
120 PRINT "                                     <_>"
130 PRINT "                                     "
140 PRINT "      _<_>_ "
150 PRINT "                                     + "
160 PRINT "      _/ \_ "
170 PRINT "      \_/ "
180 PRINT "      ^ "
190 PRINT "      | | "
200 PRINT "      | | "
210 PRINT "      / \ "
220 PRINT "                                     /\ "
230 PRINT "                                     || "
240 PRINT "                                     || "
250 PRINT "                                     || "
260 PRINT "                                     /\ "
270 PRINT "                                     @@@@ "
280 PRINT "                                     /#####\ "
290 PRINT "                                     |*****|"
300 PRINT "                                     ^^^^^ "
```

NOTE

You can use any of the special characters on the keyboard in a PRINT statement. Experiment with these characters and make some of your own spaceships.

BUGS

Here is a swarm of bugs.

Can you come up with some species of your own?

```

  \ /
  ( _ _ _ )
  3 3
  
```

```

  ! !
  /   \
  \   /
  
```

```

  +
  \ ( ) /
  / ( ) \
  / ~ \
  
```

```

  ^   ^   ^
  / \ ( ) / \
  \ /   \ /
  v     v
  
```

```

  \ _ _ /
  ( . . )
  // /\ \ \
  
```

```

  @ ~ ~ ~ ~ ~ ~ ~ ~
  3 3 3 3 3 3 3 3
  
```

PROGRAM LISTING

```

100 SCREEN 0:KEY OFF:COLOR 12,1,1:CLS
120 PRINT "          \/"
130 PRINT "          (  )"
140 PRINT "          '  '"
150 PRINT "          !!"
160 PRINT "          /  \"
170 PRINT "          \/ "
180 PRINT "          +"
190 PRINT "          \ ( ) /"
200 PRINT "          / ( ) \"
210 PRINT "          /~\"
225 PRINT "          ^  ^  ^"
230 PRINT "          /  \ ( ) /  \"
240 PRINT "          \  /  \  /"
250 PRINT "          v      v"
260 PRINT "          \___/"
270 PRINT "          ( . . )"
280 PRINT "          // /\\"
290 PRINT "          @~~~~~"
300 PRINT "          ~~~~~~"

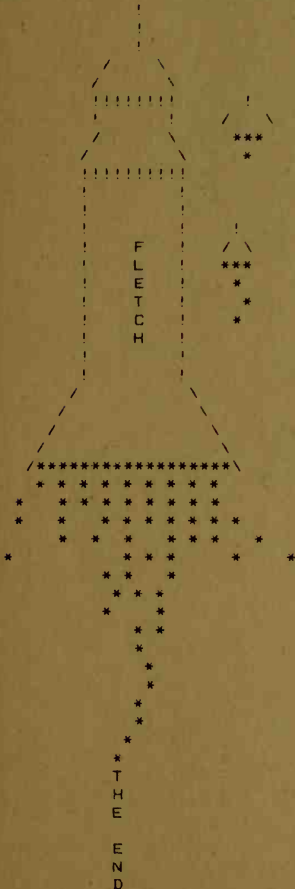
```

ROCKET SHIP

Type WIDTH 80 and you're ready to run this program. This rocket is drawn by the computer and then blasts off the screen, leaving a trail of fire.

You can do the same thing with your own spaceships.

RUN



PROGRAM LISTING

```

100 SCREEN 0:KEY OFF:COLOR 11,1,1:CLS
110 WIDTH 80
120 PRINT "          |"
130 PRINT "          |"
140 PRINT "          |"
150 PRINT "        /  \"
160 PRINT "       /    \"
170 PRINT "      | | | | |"
180 PRINT "      |         \"
190 PRINT "     /          \"
200 PRINT "    /            \"
210 PRINT "   | | | | | | |"
220 PRINT "   |             \"
230 PRINT "   |             \"
240 PRINT "   |             \"
250 PRINT "   |   F         \"
260 PRINT "   |   L         \"
270 PRINT "   |   E         \"
280 PRINT "   |   T         \"
290 PRINT "   |   C         \"
300 PRINT "   |   H         \"
310 PRINT "   |             \"
320 PRINT "   |             \"
330 PRINT "  /              \"
340 PRINT " /                \"
350 PRINT "/              \"
360 PRINT "/*****\"
370 PRINT "*****\"
380 PRINT "*****\"
390 PRINT "*****\"
400 PRINT "*****\"
410 PRINT "*****\"
420 PRINT "*****\"
430 PRINT "*****\"
440 PRINT "*****\"
450 PRINT "*****\"
460 PRINT "*****\"
470 PRINT "*****\"
480 PRINT "*****\"
490 PRINT "*****\"
500 PRINT "*****\"
510 PRINT "*****\"
520 PRINT "*****\"
530 PRINT "*****\"
540 PRINT "*****\"
550 PRINT "T\"
560 PRINT "H\"
570 PRINT "E\"
580 PRINT "\"
590 PRINT "E\"
600 PRINT "N\"
610 PRINT "D\"
620 FOR X=1 TO 40
630 PRINT
640 NEXT X

```

NOTES

Lines 620-640: This is a FOR . . . NEXT loop, which allows you to repeat part of your program as many times as you want. The FOR statement begins the loop and the NEXT statement ends it. In this program, it repeats Line 630, which PRINTs a blank line 40 times, making the rocket move up the screen so it appears to blast off.

BICYCLE

Type **WIDTH 40**. Now you can create a ten-speed bicycle. Can you design a racer or dirt bike of your own?

[illegible]

PROGRAM LISTING

```

100 SCREEN 0:KEY OFF:COLOR 15,4,4:CLS
110 PRINT "  I---"
120 PRINT "    \          _____"
130 PRINT "      ^          + &"
140 PRINT "    --+-- \_ /  ---"
150 PRINT "  -- I  -- \ /  -- I  --"
160 PRINT "  -- I  -- \ /  =  I  --"
170 PRINT "    --- + --- I /  + ---"
180 PRINT "    --- I  --- (X)  == I  --"
190 PRINT "    --- I  --- /  --- I  --"
200 PRINT "  -- I  -- =  -- I  --"

```

2

LOCATE

The LOCATE statement tells the computer in which row and column the characters are printed, indicated by the cursor.

PROGRAMS

1. Warmups
2. Face
3. Christmas Tree
4. Candy Jar
5. Tower
6. Suburb
7. School
8. Hide-and-Seek

WARMUPS

The format of the LOCATE statement is:

LOCATE row, column, N, start, stop

row: row position 1 through 25

column: column position:

1 through 40 in WIDTH 40 (in Text mode and Medium Resolution mode)

1 through 80 in WIDTH 80 (in Text mode and High Resolution mode)

N: indicates whether the cursor is visible or not:

0 is for off

1 is for on

start, stop: allow you to make the cursor any size you want. You indicate the starting and ending scan line. The scan lines are numbered from 0 at the top to 7 at the bottom of the character position.

All parameters of LOCATE are optional, and will assume the current value if omitted.

EXERCISE 1

Use LOCATE to put a character at various locations on your screen.

```
10 SCREEN 0,0,0:KEY OFF:CLS
20 LOCATE 12,20
30 PRINT "*"

```

This exercise puts an * at location row 12, column 20. Try placing the * at other locations on the screen.

EXERCISE 2

Use the WIDTH statement with LOCATE to center text.

```
10 SCREEN 0,0,0:KEY OFF:CLS
20 WIDTH 40
30 LOCATE 12,9
40 PRINT "THIS SHOULD BE CENTERED"
50 LOCATE 13,12
60 PRINT "IN 40 COLUMN TEXT"
```

THIS SHOULD BE CENTERED
IN 40 COLUMN TEXT

WIDTH 40 gives you 40 columns on the screen. LOCATE 12,9 positions the cursor on row 12, column 9. LOCATE 13,9 puts it on row 13, column 9.

Now try to center the same statement using WIDTH 80.

```
10 SCREEN 0,0,0:KEY OFF:CLS
20 WIDTH 80
30 LOCATE 12,20,0
40 PRINT "THIS SHOULD BE CENTERED IN 80 COLUMN TEXT"
```

THIS SHOULD BE CENTERED IN 80 COLUMN TEXT

FACE

Draw a girl's face and then add features. You can change her expression by altering Lines 210, 230, and 270.

```

  // // // \ \
  ||      ||
  ||  ^  ^  ||
  ||  O  O  ||
  ||      ||
  ||  ^  ||
  ||  -  ||
  || \____/ ||
  ||      ||

```

PROGRAM LISTING

```

100 SCREEN 0:KEY OFF:COLOR 15,0,0:CLS
110 PRINT "      // // // \ \
120 PRINT "      ||      ||"
130 PRINT "      ||      ||"
140 PRINT "      ||      ||"
150 PRINT "      ||      ||"
160 PRINT "      ||      ||"
170 PRINT "      ||      ||"
180 PRINT "      || \____/ ||"
190 PRINT "      ||      ||"
200 LOCATE 3,8
210 PRINT " ^  ^ "
220 LOCATE 4,8
230 PRINT " O  O "
240 LOCATE 6,9
250 PRINT " ^ "
260 LOCATE 7,9
270 PRINT " - "
280 LOCATE 15,1

```

NOTES

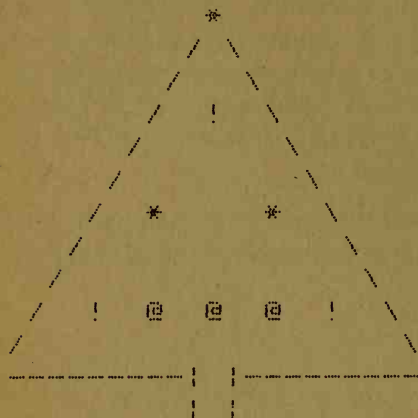
Line 100: SCREEN 0 tells the computer to start the program run in TEXT mode (SCREEN 0).

Lines 200, 220, 240, 260: LOCATE tells the computer to put the cursor at the row, column position you want. This statement must correspond to the text parameters of your screen mode (24 rows by 40 or 80 columns in Text mode, 24 rows by 40 columns in Medium Resolution mode, and 24 rows by 80 columns in High Resolution mode), or you will get an Error Message that says "Illegal Function Call" when you try to run your program.

Line 280: LOCATE 15,1 puts the cursor below the bottom of the picture, at row 15. Otherwise, the cursor displaces a line of the picture when it is printed on the printer.

CHRISTMAS TREE

Draw a Christmas tree, then go back and decorate it.
Add ornaments of your own to the program.



PROGRAM LISTING

```
100 SCREEN 0:KEY OFF:COLOR 14,1,1:CLS
110 PRINT
120 FOR L=1 TO 10
130 PRINT TAB(20-L);"/";TAB(20+L);"\ "
140 NEXT L
150 FOR L=1 TO 21
160 PRINT TAB(9+L);"-";
170 NEXT L
180 LOCATE 12,19
190 PRINT "| |"
200 LOCATE 13,19
210 PRINT "| |"
220 'DECORATE TREE
230 GOSUB 360
240 LOCATE 1,20
```

```

250 PRINT "*"
260 GOSUB 360
270 LOCATE 4,20
280 PRINT "!"
290 GOSUB 360
300 LOCATE 7,17
310 PRINT "*"      "*"
320 GOSUB 360
330 LOCATE 10,14
340 PRINT "! @ @ @ !"
350 GOTO 390
360 FOR W=1 TO 400:subroutine:wait loop
370 NEXT W
380 RETURN
390 LOCATE 20,1
400 END

```

NOTES

Line 130: The TAB statements allows you to tell the computer the column in which you want to start PRINTing. In this case, the first time through the loop, the computer will print / in column 19 (20-1) and \ in column 21 (20 + 1). The second time / will print in column 18 (20-2) and \ in column 22 (20 + 2), and so on.

Line 230: GOSUB tells the computer to leave the program and go to a subroutine, in this case Line 360.

Line 350: GOTO tells the computer to go to a designated line number, in this case Line 390.

Lines 360-380: Subroutines are small programs within a larger program that are used over and over. When the subroutine is finished, a RETURN statement tells the computer to go back to where it came from in the main program. Here the subroutine is a Wait Loop which simply makes the computer pause before it continues to decorate the tree.

Line 400: The END statement tells the computer that it is the last line in the program. It stops the Run.

CANDY JAR

This program draws a candy jar and fills it with jelly beans.
Design your own container and fill it with marbles.

```
  -----  
  |      |  
  |      |  
  |      |  
 /        \  
|0000000|  
|0000000|  
|0000000|  
|0000000|  
|_____|
```

PROGRAM LISTING

```
100 SCREEN 0:KEY OFF:COLOR 15,5,5:CLS  
110 LOCATE 7,1  
120 PRINT "  
130 PRINT "      |  |"  
140 PRINT "      |  |"  
150 PRINT "      /    \  
160 PRINT "      |      |"  
170 PRINT "      |      |"  
180 PRINT "      |      |"  
190 PRINT "      |      |"  
200 PRINT "      |_____|"  
210 FOR Y=14 TO 11 STEP -1  
220 FOR X=6 TO 11  
230 LOCATE Y,X  
240 PRINT "O";  
250 NEXT X  
260 GOSUB 290  
270 NEXT Y  
280 GOTO 320  
290 FOR T=1 TO 200
```



```
300 NEXT T
310 RETURN
320 LOCATE 20,1
330 END
```

NOTES

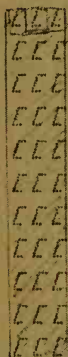
Lines 210-270: These lines form two nested FOR . . . NEXT loops, to fill the jar from the bottom up. The Y loop sets the rows and the X loop sets the columns to print "0".

Line 210: The loop of lines is incremented by the number in the STEP statement, in this case -1, so the program counts backward from 14 to 11. If there is no STEP statement, the loop is incremented by 1.

TOWER

Build a tower, using one of IBM's special ASCII characters.

Can you change the program to build a pyramid using a different character?



PROGRAM LISTING

```
100 SCREEN 0:KEY OFF:COLOR 13,4,4:CLS
110 FOR J=20 TO 10 STEP -1
120 FOR I=19 TO 21
130 LOCATE J,I
140 PRINT CHR$(219);
150 NEXT I
160 NEXT J
170 LOCATE 24,1
```

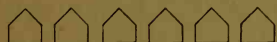
NOTE

Line 140: CHR\$(219) means the ASCII character numbered 219.
On the IBM PC this character is a ■.

SUBURB

“Suburb” draws identical houses in evenly spaced rows and columns on your screen.

Can you change the neighborhood pattern or the design of the houses?



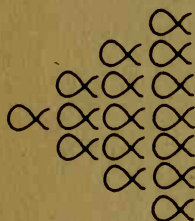
PROGRAM LISTING

```
100 SCREEN 0:KEY OFF:COLOR 14,3,3:CLS
110 FOR J=5 TO 10 STEP 2
120 FOR I=10 TO 20 STEP 2
130 LOCATE J,I
140 PRINT CHR$(127);
150 NEXT I
160 NEXT J
```

SCHOOL

Here's a school of fish the computer draws on the screen.

Can you make a swarm of arrows (IBM PC's ASCII character 26) heading in the opposite direction?



PROGRAM LISTING

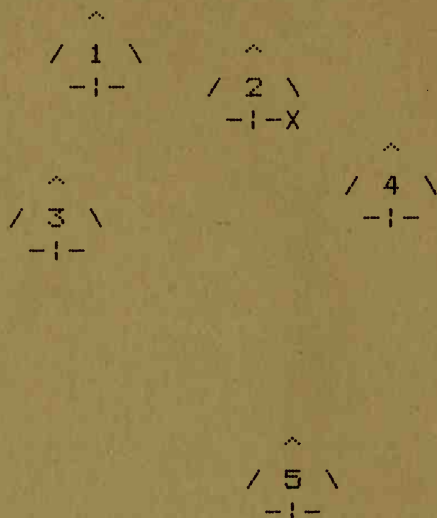
```
100 SCREEN 0:KEY OFF:COLOR 10,1,1:CLS
110 LOCATE 4,17
120 PRINT
130 FOR J=1 TO 4 'rows 1 thru 4
140 FOR I=1 TO J
150     X=21-I
160     LOCATE J,X
170     PRINT CHR$(224);
180 NEXT I
190 NEXT J
200 FOR J=5 TO 7 'rows 5 thru 7
210 FOR I=J TO 7
220     X=13+I
230     LOCATE J,X
240     PRINT CHR$(224);
250 NEXT I
260 NEXT J
270 LOCATE 20,1
```

NOTES

Lines 130 and 200: The REMARK (REM or ') statement includes nonexecutable statements in the body of the program, which can only be seen in the program listing, for the benefit of the user.

HIDE-AND-SEEK

Here's a hide-and-seek game you can play with the computer. Guess which tree X is hiding behind, and the computer will show you if you're right.



WHERE IS X HIDING?
BEHIND TREE 1,2,3,4 OR 5? 4
SORRY . . . BEHIND TREE 2!
YOU'RE STILL 'IT'!
WANT TO PLAY AGAIN (Y OR N)? N

PROGRAM LISTING

```
100 RANDOMIZE
110 SCREEN 0:KEY OFF:WIDTH 40:COLOR 15,1,3:CLS
120 A=INT(RND(1)*5+1) 'pick tree for X
130 FOR L=1 TO 5 'locate trees
140 X=INT(RND(1)*20+10)
150 Y=INT(RND(1)*15+1)
160 S=SCREEN(Y,X) 'look for empty space
```

```

170 IF S=32 THEN 180 ELSE 140
180 LOCATE Y,X      'print tree
190 PRINT " ^"
200 LOCATE (Y+1),(X-1)
210 PRINT "/" ; L ; "\"
220 LOCATE (Y+2),X
230 PRINT "-!-"
240 IF A=L THEN 270
250 NEXT L
260 GOTO 290
270 X1=X: Y1=Y
280 GOTO 250
290 LOCATE 18,1      'print questions
300 PRINT "WHERE IS X HIDING?"
310 PRINT "BEHIND TREE 1,2,3,4 OR 5";
320 INPUT N
330 IF N=A THEN 370
340 PRINT "SORRY . . . BEHIND TREE";A;"!"
350 PRINT "YOU'RE STILL 'IT'!"
360 GOTO 390
370 PRINT "YOU'RE RIGHT!!!"
380 PRINT "THERE HE IS!"
390 X1=(X1+3):Y1=(Y1+2)
400 LOCATE Y1,X1
410 PRINT "X"
420 LOCATE 22,1
430 PRINT "WANT TO PLAY AGAIN (Y OR N)";
440 INPUT N$
450 IF N$="Y" THEN 110
460 END

```

NOTES

Line 100: RANDOMIZE allows you to get a different series of random numbers each time you run a program using the RND function.

Lines 120, 140, 150: RND(1) or RND is a random-number function which generates random numbers from 0 to .9999. INT is an integer function, which changes decimal numbers to whole numbers by dropping the fraction part of the number. The statement $A = \text{INT}(\text{RND}(1) * 5 + 1)$ tells the computer: "Let A be a random number between 1 and 5."

Line 160: `S = SCREEN(Y,X)` tells the computer: "Let `S` = the number of the ASCII character located in row `Y`, column `X`."

Line 170: `IF S = 32 THEN 180 ELSE 140` is an `IF . . . THEN . . . ELSE` statement. In this program it tells the computer: "IF the ASCII character at location `Y,X` is number 32 (a blank space for the IBM PC), THEN goto 180 and start printing the tree. Otherwise, goto 140 and pick another random location." This statement keeps two trees from being printed in exactly the same location.

Lines 320, 440: The `INPUT` programming statement allows you to enter different numbers or words into your program each time it runs.

Lines 330, 450: The `IF . . . THEN` programming statement makes the computer decide which line number to execute next. IF the condition is true, THEN the program goes to the line number indicated. Otherwise, it goes to the following line number.

3 LINE and PSET

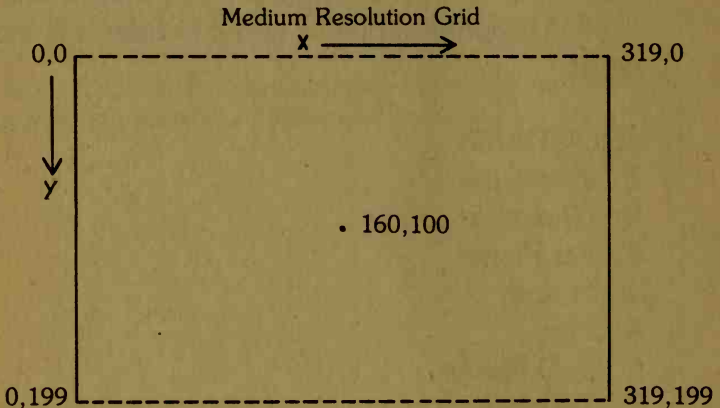
The LINE and PSET graphics commands are used in graphics modes 1 and 2 to draw a line or a point.

PROGRAMS

1. Warmups
2. Christmas Lights
3. Jet Plane
4. House
5. Flag
6. 3-D Room
7. Train
8. Skier

WARMUPS

Now you are ready to explore the graphics capabilities of your computer. The programs and exercises included in the rest of the chapters use the Medium Resolution graphics mode. This mode uses a grid 320 columns across and 200 rows down, with 16 color options. If you think of the screen as an X,Y axis, then X corresponds to the columns and Y to the rows. The top left point on the screen is (0,0) and the bottom right is (319,199). X increases as you move to the right across the screen and Y increases as you move down.



The command `SCREEN 1` puts your computer into the Medium Resolution mode. Once you type this in, you will stay in `SCREEN 1` until you do a System Reset or type in `SCREEN 0` or `SCREEN 2`.

LINE

The `LINE` graphics statement draws a line or a box on the screen. The formats for the statement are:

a) `LINE (X1,Y1)(X2,Y2),C,Box option`

`X1,Y1:` X,Y coordinates of the starting point of the line.

`X2,Y2:` X,Y coordinates of the ending point of the line.

C: the color number 0 through 3.

Box Option, where:

B: draws a rectangle with X1,Y1 and X2,Y2 at opposite corners.

BF: draws a rectangle and fills it in with color C.

b) LINE—(X2,Y2),C,Box Option

This uses the last point drawn as the starting point of the line and X2,Y2 as the ending point.

LINE is used the Medium Resolution graphics mode (SCREEN 1) and High Resolution graphics mode (SCREEN 2).

PSET/PRESET

The PSET/PRESET graphics statements plot a point on the screen.

The formats are:

PSET (X1,Y1),C

PRESET (X1,Y1),C

X1,Y1: X,Y coordinates of the point on the screen.

C: color number 0 through 3.

Color is optional in both statements. 0 uses the background color and 1, 2, and 3 use colors from the palette designated by the COLOR statement. If the color option is omitted from PSET, the foreground color is used. If color is omitted from PRESET, the background color is used. Therefore, PRESET can be used to erase a point plotted by PSET.

PSET is most frequently used to determine the point on the screen where a shape begins. Then LINE (or DRAW) is used to make the shape.

EXERCISE 1

Use PSET to place a point in the middle of the screen.

```
SCREEN 1
CLS
KEY OFF
PSET (160,100),2
```

This example uses color 2 from Palette 0 for the point. If color is omitted, PSET uses the foreground color 3.

Now erase this point using PRESET.

```
PRESET (160,100)
```

EXERCISE 2

Draw a line, using PSET.

```
10 SCREEN 1:CLS
20 FOR X=10 TO 50
30 PSET(X,50)
40 NEXT X
```

Since we did not use the color option, PSET used the foreground color 3. We can erase the line with PRESET by adding these lines:

```
50 FOR W=1 TO 600:NEXT W 'wait loop
60 FOR X=10 TO 50 'erase line
70 PRESET(X,50)
80 NEXT X
```

Notice that you can use a numeric variable as the X or Y coordinate in PSET/PRESET.

EXERCISE 3

Use the LINE statement to draw a line on the screen.

```
NEW
SCREEN 1
KEY OFF
CLS
LINE (140,100)-(180,140),2
```



Now draw a line from the end of the first line to (220,100)

```
LINE-(220,100),2
```



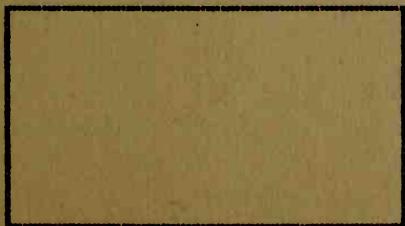
This format of the LINE statement uses the last point drawn on the screen as the starting point. This allows you to draw a continuous shape without specifying both X,Y coordinates.

EXERCISE 4

Use the LINE statement to draw a box.

```
CLS
```

```
LINE (140,60)-(220,100),2,B
```



Using the box option with LINE allows you to draw a box by specifying only the two opposite corners.

Experiment with different X,Y coordinates for the corners and see how the box rotates.

Now draw a box and fill it in with the foreground color.

```
CLS
```

```
LINE (100,100)-(160,120),,BF
```



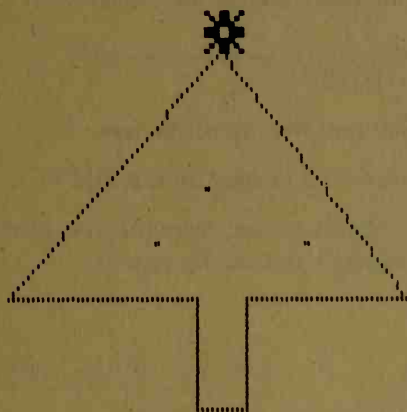
Since no color was specified, the computer used the foreground color.

Draw other boxes and experiment with the other colors. What happens when the boxes overlap? Try making a design with your boxes.

CHRISTMAS LIGHTS

Use this program to draw a small Christmas tree and put on some lights.

Add some more lights to the tree once you get this program up and running.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,0:CLS
110 LINE (80,100)-(123,55),1
120 LINE -(160,100),1
130 LINE (80,100)-(118,100),1
140 LINE -(118,120),1
150 LINE -(128,120),1
160 LINE -(128,100),1
170 LINE -(160,100),1
180 PSET (120,80),2
190 PSET (110,90),2
200 PSET (140,90),2
210 LOCATE 7,16
220 PRINT CHR$(15)
230 LOCATE 20,1
```

NOTES

Line 100: SCREEN 1 puts the computer in Medium Resolution graphics mode.

Line 110: The LINE statement draws a line from (80,100) to (123,55).

Line 120: This LINE statement continues from the last point drawn to (160,100).

Line 180: PSET puts a point at (120,80).

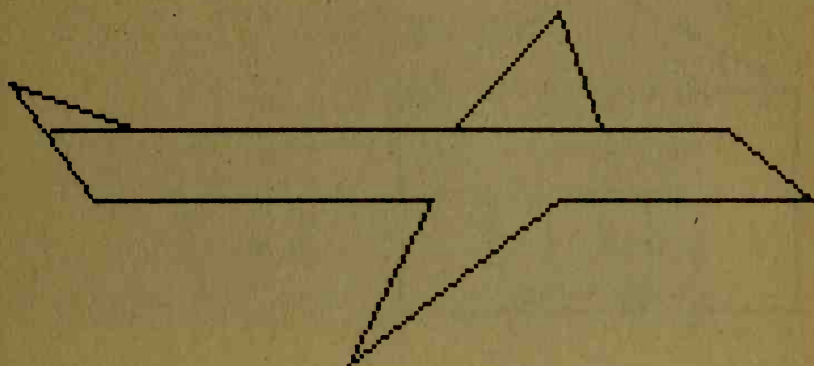
Line 210: LOCATE the row, column at the top of the tree.

Line 220: PRINT the ASCII character 15 (a star, on the IBM PC).

Line 230: Here LOCATE moves the cursor away from the tree so no lines will be displaced when the image is run on the printer.

JET PLANE

The computer can draw a Jet Liner with this program.
Can you improve upon the design?



PROGRAM LISTING

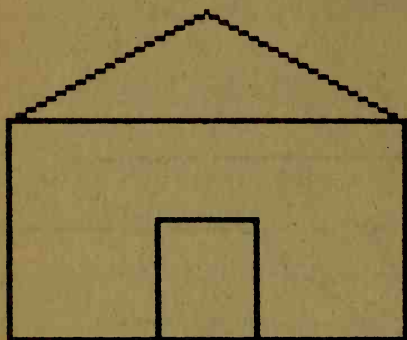
```
100 SCREEN 1:KEY OFF:COLOR 13,1:CLS
110 LINE (80,100)-(240,100),3 'TOP
120 LINE -(260,115),3 'NOSE
130 LINE -(200,115),3 'BOTTOM
140 LINE -(150,150),3 'R.WING
150 LINE -(170,115),3
160 LINE -(90,115),3
170 LINE -(70,90),3 'TAIL
180 LINE -(100,100),3
190 LINE (175,100)-(200,75),3
200 LINE -(210,100),3
```

NOTE

Lines 120-180: Notice that these LINE statements all use the prior point as the starting point.

HOUSE

Here is a small house. See if you can add windows or a chimney.



PROGRAM LISTING

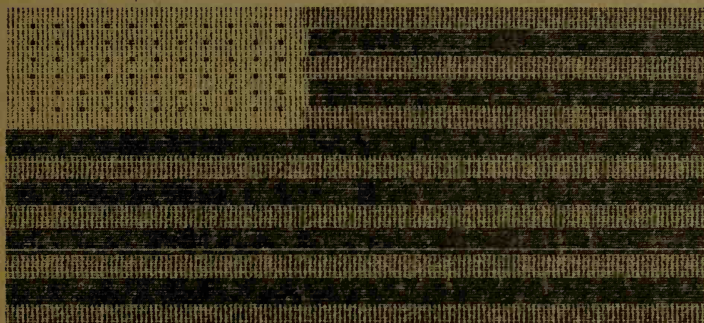
```
100 SCREEN 1:KEY OFF:COLOR 9,1:CLS
110 LINE (80,100)-(160,140),3,B
120 LINE (80,100)-(120,80),3
130 LINE-(160,100),3
140 LINE (110,118)-(130,140),3,B
```

NOTE

Lines 110, 140: The Box Option of the LINE statement draws a rectangle, using the coordinates specified as opposite corners.

FLAG

This program draws a flag with 66 stars and 13 stripes.
Can you change it to draw the American flag?



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,1:CLS
110 LINE (100,100)-(240,157),3,BF
120 LINE (100,100)-(160,121),1,BF
130 LINE (160,100)-(240,103),2,BF
140 LINE (160,109)-(240,112),2,BF
150 LINE (160,118)-(240,121),2,BF
160 LINE (100,127)-(240,130),2,BF
170 LINE (100,136)-(240,139),2,BF
180 LINE (100,145)-(240,148),2,BF
190 LINE (100,154)-(240,157),2,BF
200 Y=0
210 FOR X=5 TO 55 STEP 5
220 PSET (100+X,103+Y),3
230 NEXT X
240 Y=Y+3
250 IF Y<18 THEN 210 ELSE 270
260 GOTO 210
270 END
```

NOTES

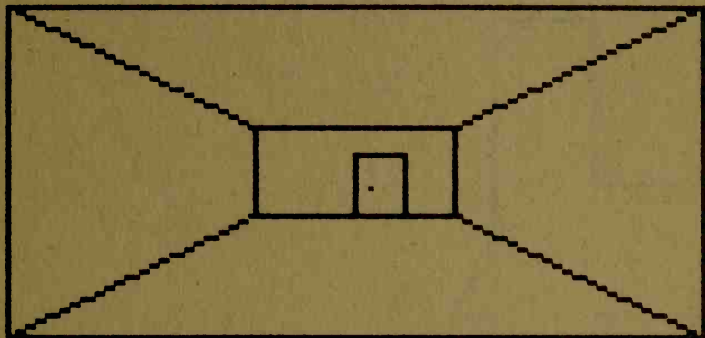
Lines 110-190: The LINE,C,BF statement tells the computer to fill in the box it draws with the color C.

Line 250: This IF . . . THEN . . . ELSE statement tells the computer, "IF the row is less than 18, THEN go to Line 210 and print another row of stars. Otherwise, go to Line 270, the END of the program."

3-D ROOM

The computer draws a 3-dimensional room with a door in the far wall.

Change the program so the far wall seems even farther away.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,1:CLS
110 LINE (80,100)-(220,160),3,B
120 LINE (130,122)-(170,138),3,B
130 LINE (80,100)-(130,122),3
140 LINE (80,160)-(130,138),3
150 LINE (220,100)-(170,122),3
160 LINE (220,160)-(170,138),3
170 LINE (150,127)-(160,138),3,B
180 PSET (153,133),2
```

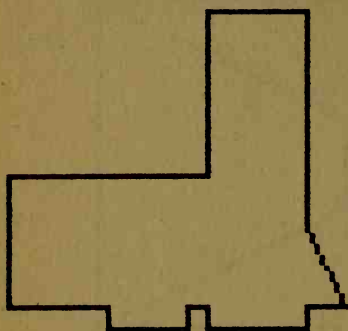
NOTE

Since this is not a continuous shape, each LINE statement requires both the beginning and ending coordinates.

TRAIN

Here is a steam engine ready to chug down the tracks in a later chapter.

Try designing a modern locomotive or a race car.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,1:CLS
110 LINE (0,46)-(40,46),3 'top
120 LINE-(40,16),3 'stack
130 LINE-(60,16),3
140 LINE-(60,56),3
150 LINE-(68,70),3 'fender
160 LINE-(60,70),3
170 LINE-(60,74),3
180 LINE-(40,74),3 'wheel
190 LINE-(40,70),3
200 LINE-(36,70),3
210 LINE-(36,74),3
220 LINE-(20,74),3
230 LINE-(20,70),3
240 LINE-(0,70),3 'bottom
250 LINE-(0,46),3 'back
```

SKIER

This skier will take off down the slopes in a later chapter.
See if you can design a person on a sled to substitute for the skier.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,1:CLS
110 CIRCLE(150,63),3,3 'head
120 LINE(150,65)-(150,78),3 'torso
130 LINE-(145,100),3 'l. leg
140 PSET (150,78),3
150 LINE-(155,100),3 'r. leg
160 PSET(150,68),3
170 LINE-(146,68),3 'l. arm
180 LINE-(143,75),3
190 LINE-(135,70),3
200 PSET(136,68),3
210 LINE-(125,100),3 'l. pole
220 PSET(150,68),3
230 LINE-(154,68),3 'r. arm
240 LINE-(157,75),3
250 LINE-(165,70),3
260 PSET(166,68),3
270 LINE-(155,100),3 'r. pole
280 PSET(140,96),3
```

```
290 LINE-(157,110),3 'l. ski  
300 PSET(150,96),3  
310 LINE-(167,110),3 'r. ski
```

NOTE

Line 110: This CIRCLE programming statement tells the computer to draw a circle with a midpoint of X coordinate 150, Y coordinate 63, with a radius of 3, in color number 2. For a more detailed explanation of the CIRCLE statement, see Chapter 4.

4 DRAW and CIRCLE

DRAW and CIRCLE are graphics statements used in graphics modes 1 and 2. DRAW provides an abbreviated graphics language to create shapes and pictures. CIRCLE draws an ellipse on the screen.

PROGRAMS

1. Warmups
2. Fish
3. Car
4. Fishing Boat
5. Windmill
6. Snowflakes
7. Sailboat
8. Campground
9. Picture

WARMUPS

The graphics statements in this chapter allow you to do many complex graphics operations with one command.

DRAW

The DRAW statement contains single-letter commands that are used to draw a figure.

The commands are:

Un: move up.

Dn: move down.

Ln: move left.

Rn: move right.

En: move diagonally up and right.

Fn: move diagonally down and right.

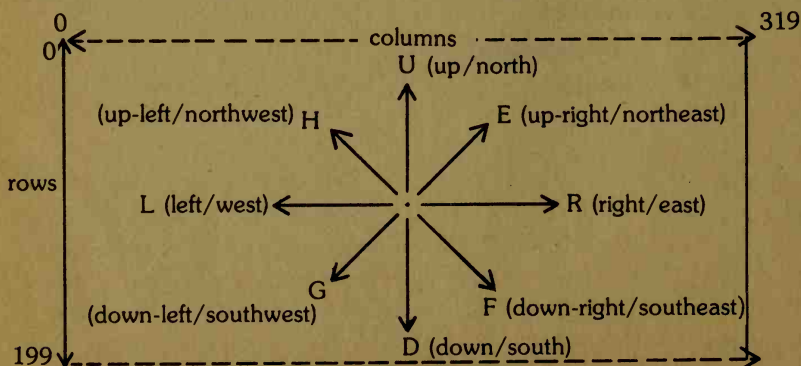
Gn: move diagonally down and left.

Hn: move diagonally up and left.

(n in each of the above commands indicates the distance to move.) The diagonal commands E, F, G, and H actually move 1.4 times n.

MX1,Y1: move to the X,Y coordinates specified by X1,Y1.

Think of your screen as a map with these geometric directions:



The following commands can precede the movement commands:

B: move but don't plot.

N: move and then return to the starting position when finished.

The other commands available are:

An: set angle n, where n ranges from 0 to 3, and

0 = 0 degrees

1 = 90 degrees

2 = 180 degrees

3 = 270 degrees.

This rotates the figure being drawn by the degrees set in n.

Cn: set color n, where n ranges from 0 to 3.

Sn: set scale factor. N may range from 1 to 255, and $n/4$ is the scale factor. The scale factor is then multiplied by the distances used in the Move commands.

XA\$: execute substring. A\$ is set elsewhere in the program to a string of DRAW commands. The X command allows you to execute A\$ from within a character string of DRAW commands.

All the DRAW commands are contained in a character string. The format is:

DRAW "draw commands"

or

DRAW A\$, where A\$ is set to the DRAW commands elsewhere in the program.

The advantage of LINE and PSET statements is that they can be set equal to numeric variables in counters and in RND functions.

The advantage of the DRAW statement's graphic commands is their ability to store your figures in strings. Remember, if you use the Move command, which specifies X,Y coordinates in the DRAW statement, it will affect your flexibility in using Angle and Scale com-

mands. The U, D, E, F, G, and H commands are relative to your starting position, whereas the Move commands are absolute.

CIRCLE

The CIRCLE statement draws a circle or part of a circle.
The format is:

CIRCLE (X1,Y1),R,C,S,E,A where

X1,Y1 X,Y: coordinates for the center of the circle

R: radius of circle

C: color to use

S: radians to begin drawing

E: radians to end drawing

A: Aspect or ratio of X axis to Y axis

If S and E are omitted, an entire circle is drawn. To draw an arc, set S and E to a value between $-2 * \pi$ and $2 * \pi$, where $\pi = 3.14159$. For example, to draw an arc between 0 and π , set $S=0$, $E=3.14159$.

The parameter, A, determines the shape of the ellipse. An aspect of 5/6 in Medium Resolution gives a circle, whereas an aspect of 3/6 will make an ellipse.

The parameters X1,Y1 and R are required, the others are optional.

EXERCISE 1

Try these first exercises in the immediate mode, without line numbers. First type in SCREEN 1, then CLS, and then KEY OFF. This will get you into Medium Resolution, clear the screen, and remove the soft key statements.

Draw a line right of the screen midpoint of length 50.

DRAW"R50"



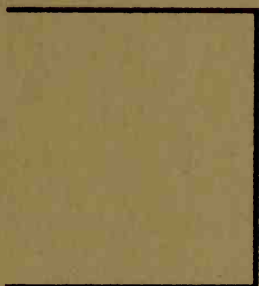
Draw a line down 50.

DRAW"D50"



Now draw 50 units left.

DRAW"L50"



And finish the rectangle with 50 units up.

DRAW"U50"



Now explore the DRAW statement by experimenting with different directions and lengths. Notice that the first DRAW statement starts at the screen midpoint. Each DRAW statement following starts at the last point drawn.

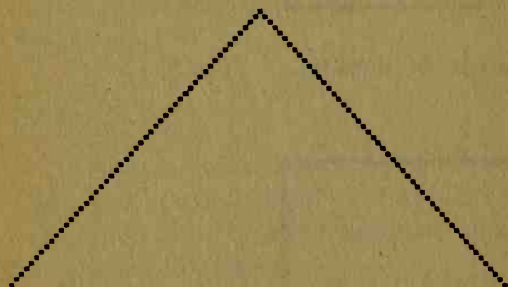
EXERCISE 2

Now experiment with the DRAW commands on the diagonal.

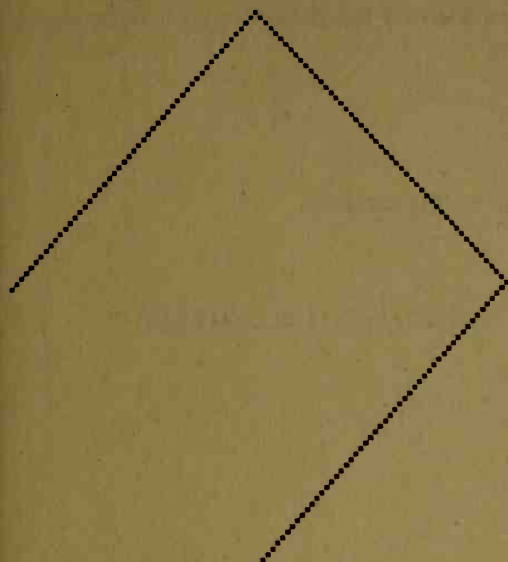
DRAW"E50"



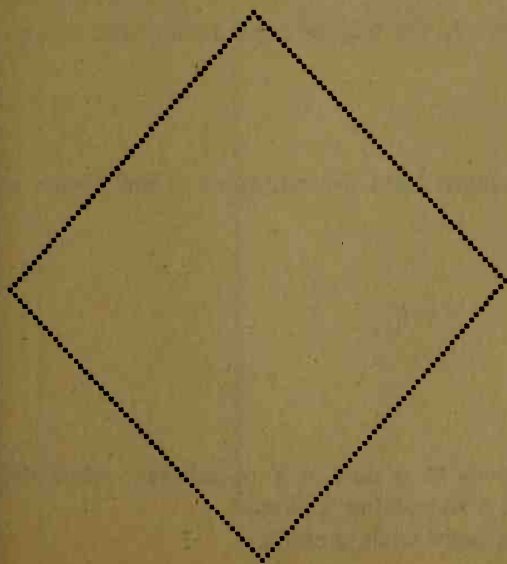
DRAW"F50"



DRAW"G50"



DRAW"H50"



You have now made a diamond. Try other combinations of these commands until you have learned the direction each letter stands for.

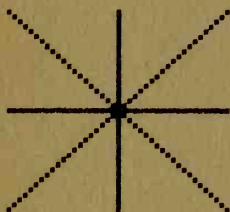
EXERCISE 3

Make your own asterisk on the screen.

```
DRAW"NU20"
```

```
DRAW"NE20"
```

```
DRAW"NR20;NF20;ND20;NG20;NL20;NH20"
```



The N command returns to the original position after the move.

EXERCISE 4

Move the starting position from the midpoint of the screen to (20,100).

```
DRAW"BM20,100"
```

```
DRAW"D10"
```



The M command moves to a new X,Y coordinate, while the B command is used with it so nothing is plotted.

Now move the starting point while plotting.

DRAW"BM20, 100;M20, 200"



Now draw a line parallel to this one.

DRAW"BM60, 100;M60, 200"



EXERCISE 5

Now use the DRAW statement in a simple program. Where will BOX\$ start?

```
10 SCREEN 1:CLS:KEY OFF
20 BOX$="R20;U10;L20;D10"
30 DRAW BOX$
```



Notice that the DRAW statement can use a character string defined elsewhere in the program. This allows you to save the commands used to draw a shape or figure in a string variable and then use the shape over and over, alone or in combination with others.

Rotate the rectangle by adding this line to your program and running it.

```
25 DRAW "A1"
```



and the angle rotates 90 degrees. Now substitute this line and run it.

```
25 DRAW "A2"
```



This rotates the angle 180 degrees.

To rotate the angle 270 degrees substitute this line and run the program.

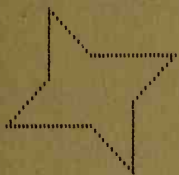
```
25 DRAW "A3"
```



DRAW "An" sets the angle n to rotate. When $n = 1$, it rotates 90 degrees, when $n = 2$, it rotates 180 degrees and when $n = 3$, it rotates 270 degrees. Type DRAW "A0" to get back to 0 degrees angle, the graphics mode's normal orientation.

EXERCISE 6

Type NEW and you're ready to draw a star in color.



```
10 SCREEN 1:CLS:KEY OFF
20 STAR$="R20;G10;D15;H10;L20;E10;U15;F10"
30 DRAW "C1"
40 DRAW STAR$
```

The DRAW "Cn" command sets the color for the DRAW statement. N ranges from 0 to 3.

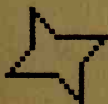
Now experiment with different values for n in line 30. See what colors you get.

Notice that if you put the C command in a separate statement, it is much easier to change the colors you use.

EXERCISE 7

Reduce the size of the star using the scale command, DRAW "Sn". Replace Line 30 in Exercise 6 with:

```
30 DRAW "S2"
```

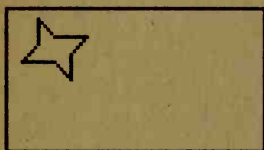


This star is $\frac{1}{2}$ the size of the one in the previous exercise.

The scale command allows you to easily change the size of the shapes you draw. Try different values for *n* in Line 30 and see how big your star can be.

EXERCISE 8

Type NEW and you're ready to experiment with the X command.



```
10 SCREEN 1,0:CLS:KEY OFF
20 STAR$="S2;R20;G10;D15;H10;L20;E10;U15;F10"
30 SHAPE$="S8;R40;D20;L40;U20"
40 DRAW SHAPE$
50 DRAW"BM175,108 X STAR$;"
```

The X command allows you to execute a second string within a string. In this example, Line 50, DRAW"BM175,108X STAR\$;", the BM command moves to the X,Y coordinates of (175,108) without plotting. Then the X STAR\$ command draws the star.

In all of these DRAW commands, the argument *n* can be a constant, like DRAW"H20", or set equal to a numeric variable, like DRAW"H=X;". A semicolon is required if you use a variable this way, and is required also for the X command: DRAW "BD90 X STAR\$;". Otherwise, the semicolon is optional. We use it in our DRAW statements to make the statements easier to read and copy.

EXERCISE 9

Use the CIRCLE statement to draw a circle in the middle of the screen.

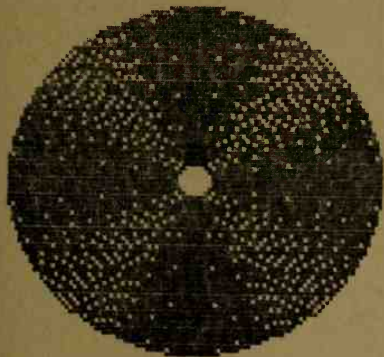
```
NEW  
SCREEN 1  
CLS  
CIRCLE (160, 100), 5
```



This circle has a radius of 5. Use the CIRCLE statement to make other circles on the screen.

EXERCISE 10

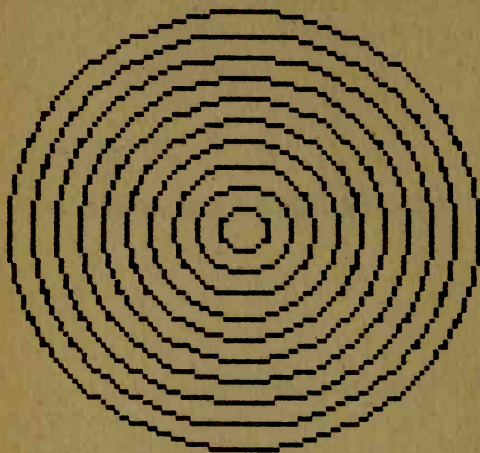
Use the CIRCLE statement to make a design.



```
10 SCREEN 1:CLS:KEY OFF  
20 FOR X=5 TO 50  
30 CIRCLE (160, 100), X  
40 NEXT X
```

Change Line 20 to make concentric circles.


```
20 FOR X=5 TO 50 STEP 5
```



EXERCISE 11

Now draw part of a circle.

```
NEW
```

```
SCREEN 1
```

```
CLS
```

```
CIRCLE (160, 100), 10, , 0, 1.57
```



Experiment with the values for S (start) and E (end) and see what portion of a circle you draw.

Notice that the color parameter is omitted. However, a, is required to keep the rest of the parameters in the proper order.

To make an ellipse, change the aspect:

```
CIRCLE (160, 100), 10, , , 3/6
```



Try other values and see what your ellipse looks like.

FISH

Use the DRAW statement to make a fish.

Can you change the size of this fish and add others to the picture?



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,1:CLS
110 DRAW"C3;BM65,150" 'position fish
120 DRAW"M96,140;M110,150;M96,158;M62,145;M65,150"
130 PSET(103,148),2
```

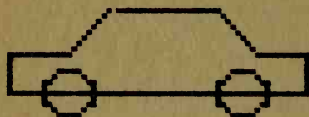
NOTES

Line 110: DRAW"BM65,150" tells the computer to move from the screen midpoint to the X,Y coordinates of (65,150) but not plot any points.

Line 120: This DRAW statement tells the computer to move, drawing a line, to each of the coordinates given.

CAR

Here is a car ready to drive across the screen in a later chapter.
Try to design your favorite sports car or a racing model.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,1:CLS
110 DRAW "BM100,60" 'position car
120 DRAW "C3;R20;F8;R10;D7" 'top
130 DRAW "C3;L60;U7;R12;E8" 'bottom
140 CIRCLE (90,75),5,3 'l.tire
150 CIRCLE (125,75),5,3 'r.tire
```

NOTES

Line 110: This DRAW statement positions the starting point of the car at (100,60).

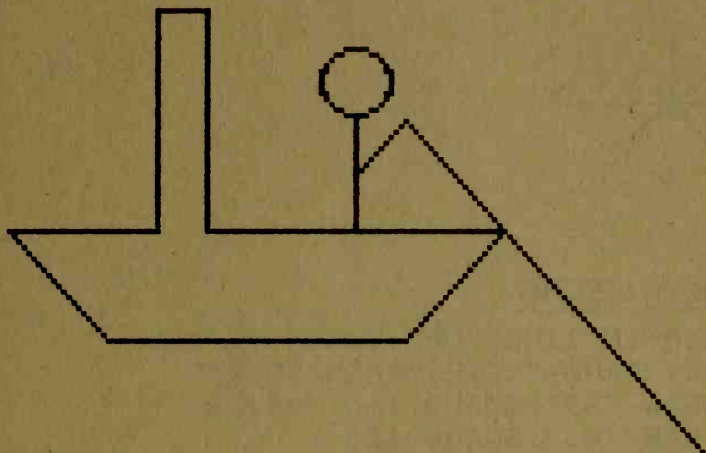
Lines 120, 130: These DRAW statements make the car. The DRAW commands are separated by semicolons to make the line easier to read. The semicolons are optional in the statement.

Lines 140, 150: The CIRCLE statements draw the tires of the car.

FISHING BOAT

This fisherman is hoping a fish will come along.

See if you can change the program to make his boat look like a cabin cruiser, or add some fish to the picture.



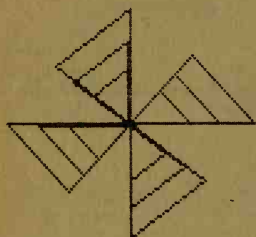
PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,1:CLS
110 DRAW "BM100,140" 'position boat
120 DRAW "C3;R30;U40;R10;D40;R30;U20;D10;E10;F60" 'top
130 CIRCLE (170,113),7,3 'man's head
140 DRAW "BM170,140" 'position stern
150 DRAW"C3;R30G20L60H20" 'btm. of boat
```

WINDMILL

The computer draws the arms of a windmill.

Add to the program to make it a pinwheel or propeller.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,0:CLS
110 LET ARM$="C3;E25;F25;L50;"
120 DRAW "S2":GOSUB 160 'sizes
130 DRAW "S3":GOSUB 160
140 DRAW "S4":GOSUB 160
150 GOTO 210
160 DRAW "A0 X ARM$;" 'angles
170 DRAW "A1 X ARM$;"
180 DRAW "A2 X ARM$;"
190 DRAW "A3 X ARM$;"
200 RETURN
210 END
```

NOTES

Line 110: This line sets ARM\$ equal to the commands needed to draw an arm.

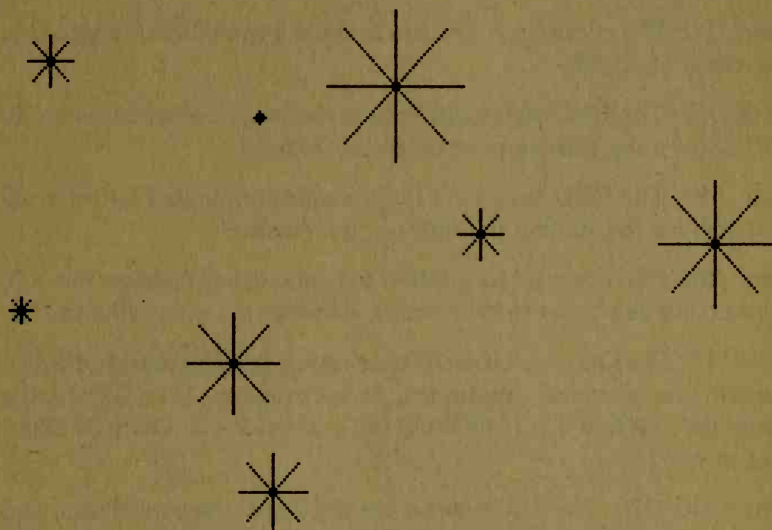
Lines 120-140: The DRAW "Sn" command sets the size of the arm.

Lines 160-190: The DRAW "An" command sets the angle of the arm. The X command is used to call the string variable ARM\$.

SNOWFLAKES

Fill the screen with different sized snowflakes.

Change the program to print out even more random snowflakes on your screen.



PROGRAM LISTING

```
100 RANDOMIZE
110 SCREEN 1:KEY OFF:COLOR 9,1:CLS
120 LET FLAKE$="C3;NU4;NE3;NR4;NF3;ND4;NG3;NL4;NH3"
130 FOR T=1 TO 8 'rnd positions for snow
140 X=INT(RND(1)*280+20)
150 Y=INT(RND(1)*160+20)
160 PSET(X,Y) 'position flake
170 ON T GOSUB 200,210,220,230,240,250,260,270
180 NEXT T
190 GOTO 280
200 DRAW "S2 X FLAKE$;":RETURN 'sizes
210 DRAW "S4 X FLAKE$;":RETURN
220 DRAW "S8 X FLAKE$;":RETURN
230 DRAW "S12 X FLAKE$;":RETURN
```

```
240 DRAW "S16 X FLAKE$;":RETURN
250 DRAW "S20 X FLAKE$;":RETURN
260 DRAW "S24 X FLAKE$;":RETURN
270 DRAW "S8 X FLAKE$;":RETURN
280 END
```

NOTES

Line 120: The commands needed to draw a snowflake are stored in the string FLAKE\$.

Line 140: The RND statement finds a random number between 20 and 280 for the starting position of the X axis.

Line 150: The RND statement finds a random number between 20 and 160 for the starting position on the Y axis.

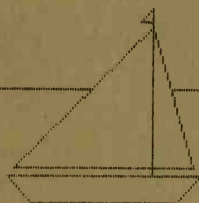
Line 160: PSET is used to position the snowflakes because the X,Y coordinates can be set to the numeric variables in Lines 140 and 150.

Line 170: The ON . . . GOSUB statement is used to branch the program to one of several subroutines. In this program, Line 170 tells the computer: "When T = 1, GOSUB 200; when T = 2, GOSUB 300," and so on.

Lines 200-270: The S command sets the size of the snowflakes, and the X command executes the DRAW commands stored in FLAKE\$.

SAILBOAT

Draw a picture on your screen of a sailboat.
Can you add some sea gulls to the picture?



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,0:CLS
110 LET BOW$="C1;BM160,100;R80;G10;L60;H10"
120 LET MAIN$="C1;BM220,100;U55;M163,97;M220,97"
130 LET JIB$="C1;BM220,45;M237,97;M220,97"
140 LET FLAG$="C1;BM220,45;U7;G5;R5"
150 DRAW BOW$:DRAW MAIN$:DRAW JIB$:DRAW FLAG$
190 DRAW "C1;BM0,68;R195;BM228,68;R92" 'horizon
210 CIRCLE (40,20),15,1 'moon
```

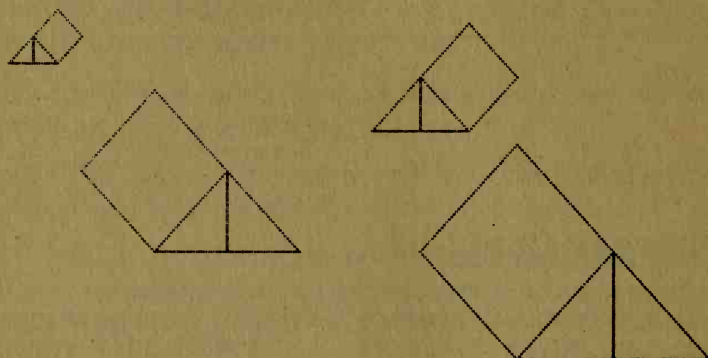
NOTE

Lines 110-140: Each part of the sailboat is stored in a string, to be used by the DRAW statements in Line 150.

CAMPGROUND

Here's a campground scene.

Try adding some figures or trees to the picture.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,0:CLS
110 LET FRONT$="C2;NG20;ND20;F20;L40"
120 LET SIDES$="C2;E20;F20;G20;H20"
130 LET A$="A0" 'angle of sides
140 LET A2$="A2"
150 FOR X=1 TO 4
160 ON X GOSUB 190,230,270,310
170 NEXT X
180 GOTO 430
190 LET START$="BM40,70" 'position #1
200 DRAW "S2":GOSUB 350
220 RETURN
230 LET START$="BM120,120"'position #2
```



```

240 DRAW "S6":GOSUB 390
260 RETURN
270 LET START$="BM200,85" 'position #3
280 DRAW "S4":GOSUB 350
300 RETURN
310 LET START$="BM280,150"'position #4
320 DRAW "S8":GOSUB 390
340 RETURN
350 'tent routine for #1 & #3
360 DRAW START$:DRAW FRONT$:DRAW START$
370 DRAW A$:DRAW SIDES$
380 RETURN
390 'tent routine for #2 & #4
400 DRAW START$:DRAW FRONT$:DRAW START$
410 DRAW A2$:DRAW SIDES$:DRAW A$
420 RETURN
430 LINE(0,51)-(319,51),2
450 CIRCLE (260,20),10,2 'moon

```

NOTES

Line 110, 120: The commands used to draw the front and side of the tents are stored in strings. Then these strings can be used in other parts of the program.

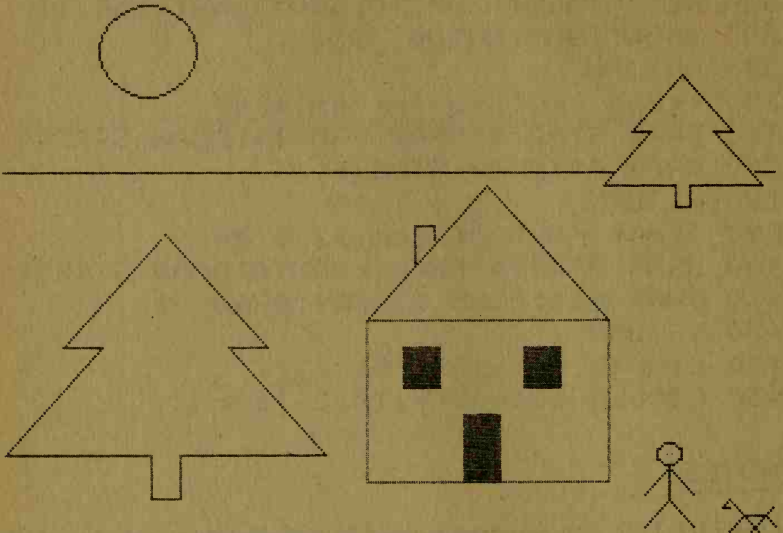
Line 130, 140: These lines set a string equal to two different angles. Later in the program these angles are used to rotate the tent.

Lines 190, 230, 270, 310: The starting location for each tent is stored in START\$. Then the tent routines use this string to find the starting positions to draw the tents.

PICTURE

Draw this picture with your computer.

What additions can you make to the scene?



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,1:CLS
110 LET TREE$="C3;G20;R8;G20;R30;D8;R6;U8;R30;H20;R8;H21"
120 DRAW "BM65,90" 'position lrg. tree
130 DRAW "S8 X TREE$;"
140 DRAW "BM280,30" 'position sm. tree
150 DRAW "S4 X TREE$;"
160 LINE (150,120)-(250,180),2,B 'house
170 DRAW "C3;BM150,120;E50;F50" 'roo
190 LINE (165,130)-(180,145),3,BF 'l.win
200 LINE (190,155)-(205,180),3,BF 'door
210 LINE (215,130)-(230,145),3,BF 'r.win
220 DRAW "C3;BM170,100;U15;R8;D5" 'chimney
240 CIRCLE (275,170),5,3 'man's head
260 PSET(274,170),1
270 PSET(276,170),1
280 DRAW "C3;BM275,175;NF10;NG10;D12;NF10;NG10" 'man
290 DRAW "BM300,190" 'dog's position
300 DRAW "C3;L3;E3;F6;NG6;NF6;R9;NG6;NF6;E3" 'dog
310 DRAW"C3;BM0,65;R253;BM311,65;R8" 'horizon
320 CIRCLE (60,20),20,3 'sun
```

NOTE

This program demonstrates the most useful features of the DRAW and LINE programming statements by combining their use to draw a picture.

5 PUT and GET

The PUT and GET graphics statements make it easy to move objects around on the screen.

PROGRAMS

1. Warmups
2. Apple Tree
3. Walking Figure
4. Flying Bird
5. Jumping Fish
6. Steam Engine

WARMUPS

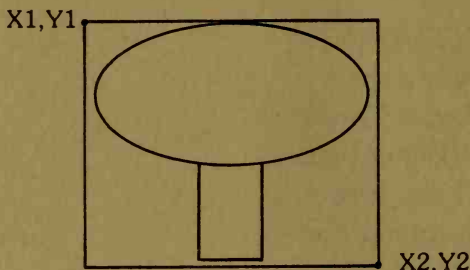
GET

The GET command stores a shape already on the screen into an array. The format is:

GET(X1,Y1) – (X2,Y2),Array

X1,Y1: X,Y coordinates of the top left corner of the rectangle which contains the shape.

X2,Y2: X,Y coordinates of the bottom right corner of the rectangle.



Array: The name of the array used to store the shape.
The size of the array for Medium Resolution (SCREEN 1) is calculated as follows:

$$\text{Size} = (4 + \text{INT}((2 * X + 7)/8) * Y)/4$$

where $X = X2 - X1 + 1$

$Y = Y2 - Y1 + 1$

For example, if we want to store a rectangle, using GET(88,140)-(112,170),T

Then $X1 = 88$ $Y1 = 140$ $X2 = 112$ $Y2 = 170$

And $X = 112 - 88 + 1 = 25$ $Y = 170 - 140 + 1 = 31$

$\text{Size} = (4 + \text{INT}((2 * 25 + 7)/8) * 31)/4 = 55$

If you try to use an array size too small for the rectangle you will get the error message "ILLEGAL FUNCTION CALL."

PUT

The PUT statement displays on the screen the shape stored in the array by GET. Once a shape is in the array, it can be PUT at any location on the screen. The format is:

PUT(X1,Y1), Array,Action

X1,Y1: X,Y coordinates of the screen location to put the left corner of the rectangle.

Array: The numeric array containing the shape

Action: May be one of:

PSET Displays shape on screen.

PRESET Same as PSET but produces an inverse image.

XOR Displays shape on the screen and also will invert points on the screen that match the shape.

Used to erase a prior image in animation.

AND Displays a shape only if one exists under it.

Used to overlap images on the screen.

OR Superimposes a shape over an existing one.

If no action is specified, XOR is the default action.

Animation is done using either XOR or PSET.

To use XOR, just PUT a shape twice in the same X,Y coordinates. The first time draws the shape, and the second time erases it.

To use PSET, make the rectangle large enough to contain the shape at both the old location and the new one. PSET is faster than XOR and has less flicker, but will erase any objects in the background. XOR preserves the background.

EXERCISE 1

Moving a shape across the screen requires 3 steps:

1. Draw the shape.
2. Erase the shape.
3. Draw it in a new location.

The faster the computer can do steps 1, 2, and 3, the faster the shape will move. If there is a delay between steps 2 and 3, the shape will flicker as it moves.

Any of the graphics statements can be used to draw the shape. Then to erase it, use the same statements, but set the color variable to 0, the background color.

Use the CIRCLE statement to draw a circle and then move it across the screen.



```
100 SCREEN 1: CLS: KEY OFF
110 FOR X=5 TO 250 STEP 10
120 CIRCLE(X,50),5,3 'Draw circle
130 CIRCLE(X,50),5,0 'Erase circle
140 NEXT X
```

Try changing the size of the step in Line 110 and see how the movement of the circle changes.

Change the program to move the circle up the screen.

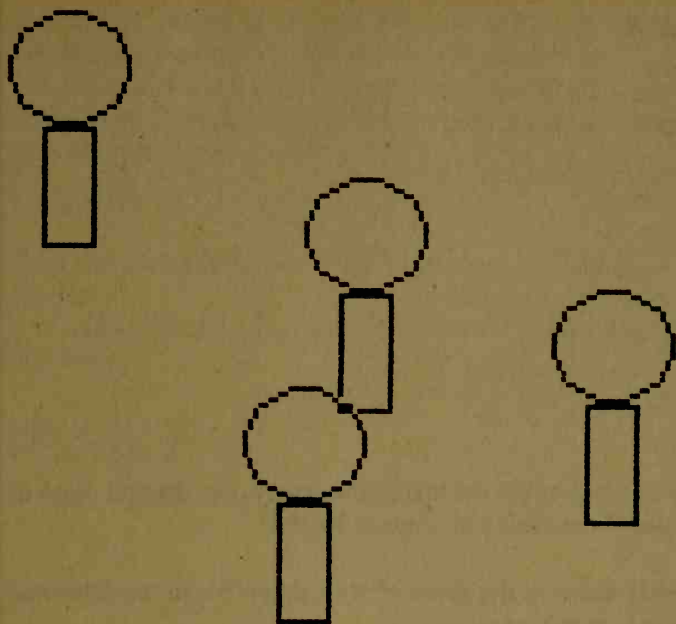


```
100 SCREEN 1: CLS: KEY OFF
110 FOR Y=200 TO 10 STEP -10
120 CIRCLE(50,Y),5,3 'Draw circle
130 CIRCLE(50,Y),5,0 'Erase circle
140 NEXT Y
```

Put in a Wait Loop after Line 110. Does this change how the circle moves? Put in a Wait Loop after Line 120—now what happens?

EXERCISE 2

Now use GET and PUT to move objects on the screen. Draw a tree and then use it to make a forest.



```
100 SCREEN 1: CLS: KEY OFF
110 DIM T(80)
120 CIRCLE(100,100),12 'Draw tree
130 LINE(95,111)-(105,132),,B
140 GET(88,88)-(112,132),T'Save tree
150 PUT(100,50),T 'Draw tree
160 PUT(40,20),T
170 PUT(150,70),T
```

Notice that the GET statement does not erase the original shape from the screen. This exercise PUTs the original tree at three additional locations. Add more PUT statements to this exercise to fill up the forest with trees.

EXERCISE 3



```
100 SCREEN 1: CLS: KEY OFF
110 DIM C(96)
120 Y=10
130 PSET (100,60)           'Draw car
140 DRAW "R20;F8;R10;D7
150 DRAW "L60;U7;R12;E8
160 CIRCLE (90,75),5
170 CIRCLE (125,75),5
180 GET(72,60)-(142,80),C
190 FOR X=10 TO 200 STEP 10 'Move car
200 PUT(X,Y),C
210 NEXT X
```

This exercise moves the car across the screen, but doesn't erase as the car moves. There are two ways to fix this:

1. Use PUT twice at the same X,Y location. Make the following changes to this exercise:

```
200 PUT(X,Y),C,XOR
205 PUT(X,Y),C,XOR
```

This method restores any objects in the background.

2. Use PUT with PSET and increase the size of the rectangle to contain both the old shape and the new shape.

Make the following changes to the exercise:

```
110 DIM C(124)
180 GET(52,60)-(142,80),C
200 PUT(X,Y),C,PSET
DELETE 205
```

PSET erases the old shape as the car moves across the screen. However, it also erases any shapes in the background.

Now experiment with this program to see how PUT and GET work. Try increasing the size of the step increment on LINE 190 and see if the prior image is still erased. If you place the car in a new loca-

tion too far ahead of the old one, the rectangle will not be large enough to contain both the old and the new shape.

There is still one thing to fix in this exercise. The first car we drew to put in array C stays on the screen. In order to erase this car make the following change:

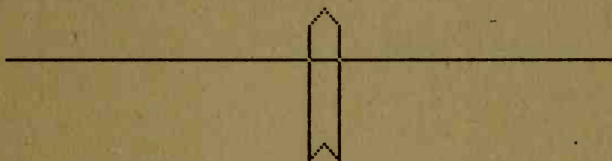
```
185 PUT (52,60),C,XOR
```

Now we have a program that draws a car and moves it across the screen.

You can use this same type of program to move any object across the screen.

EXERCISE 4

Make a rocket blast off the screen.



```
100 SCREEN 1: KEY OFF: CLS
110 DIM R(40)
120 PSET (10,20)
130 DRAW"B5:D40;E5:F5:U40:H5" 'make rocket
140 GET (5,15)-(15,65),R
150 PUT (5,15),R,XOR
160 X=150
170 LINE (50,50)-(250,50) 'draw line
180 FOR Y=90 TO 10 STEP -10
190 PUT (X,Y),R,XOR 'draw rocket
200 PUT (X,Y),R,XOR 'erase rocket
210 NEXT Y
```

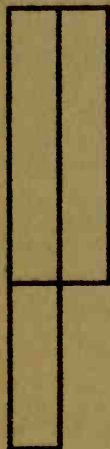
Notice that the line at the top of the screen is restored after the rocket goes through. Using PUT with XOR preserves any shapes in the background.

Line 150 erases the original rocket we drew in order to store the shape in array R.

Now try some programs of your own and move objects across or up and down the screen.

EXERCISE 5

AND and OR allow you to control the image made when two objects overlap on the screen. Using PUT with AND, displays only the points the two shapes have in common. Using PUT with OR displays all points made by both shapes. Using a combination of PUT's with AND and OR allows you to superimpose one image on another. Use these boxes to practice with AND and OR.



```
100 CLS: SCREEN 1: KEY OFF
110 DIM B(80),C(61)
120 LINE(10,50)-(30,100),,B 'Box B
130 GET(10,50)-(30,100),B
140 CLS
150 LINE(10,50)-(20,130),,B 'Box C
160 GET(10,50)-(20,130),C
170 CLS
180 PUT(40,10),B 'Display Box B
190 PUT(40,10),C,OR 'Display Box C
```

This exercise displays all the points on both boxes.

To display only the points of box C in common with box B, make the following change:

190 PUT (40, 10), C, AND



To overlap box B on box C, make the following changes:

190 PUT (40, 10), C, OR

200 PUT (40, 10), B, AND "overlaps B on C



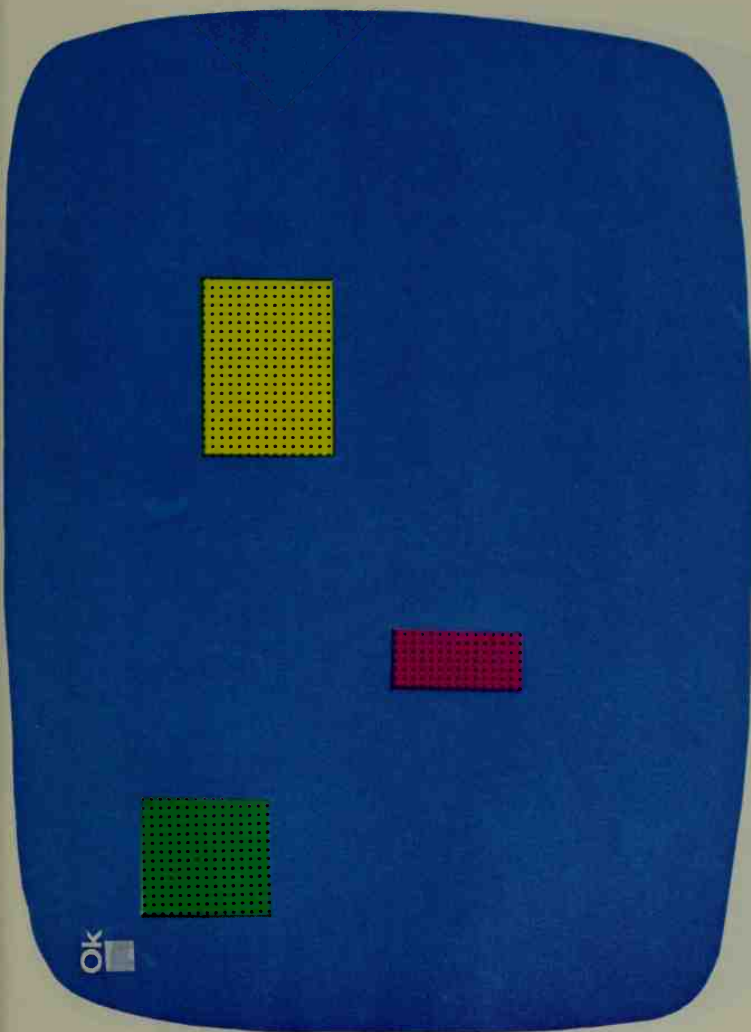
To overlap box C onto box B, make these changes:

```
180 PUT(40,10),C  
190 PUT(40,10),B,OR  
200 PUT(40,10),C,AND
```

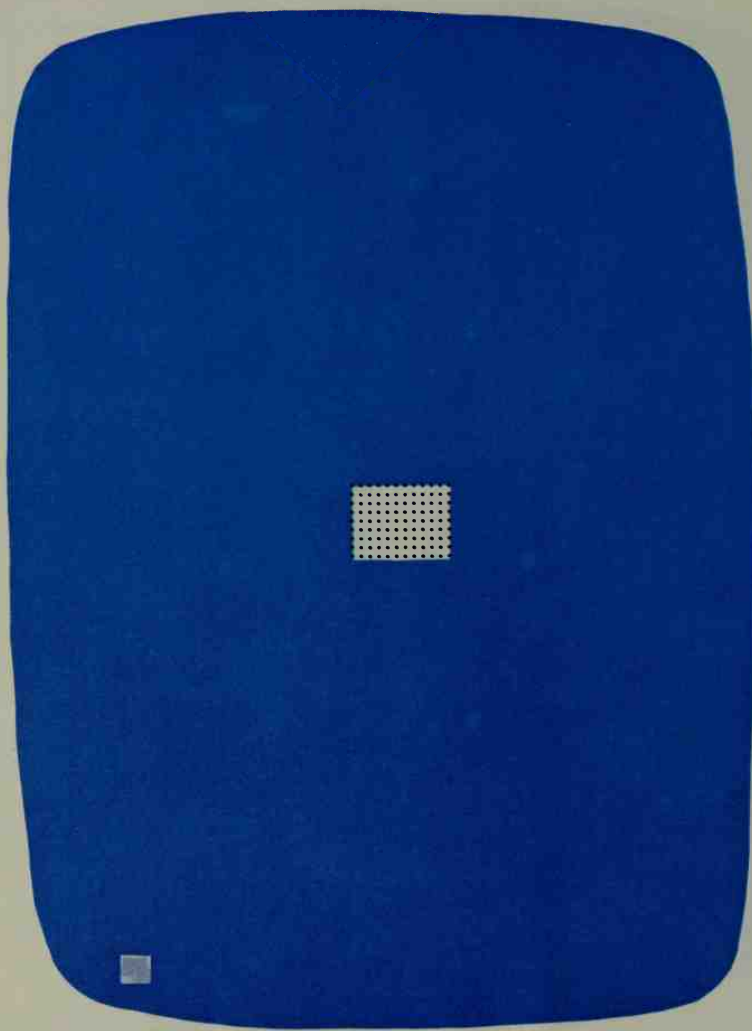


Now we are ready to try some programs that use what you have just learned.

GRAPHICS FROM CHAPTER 6



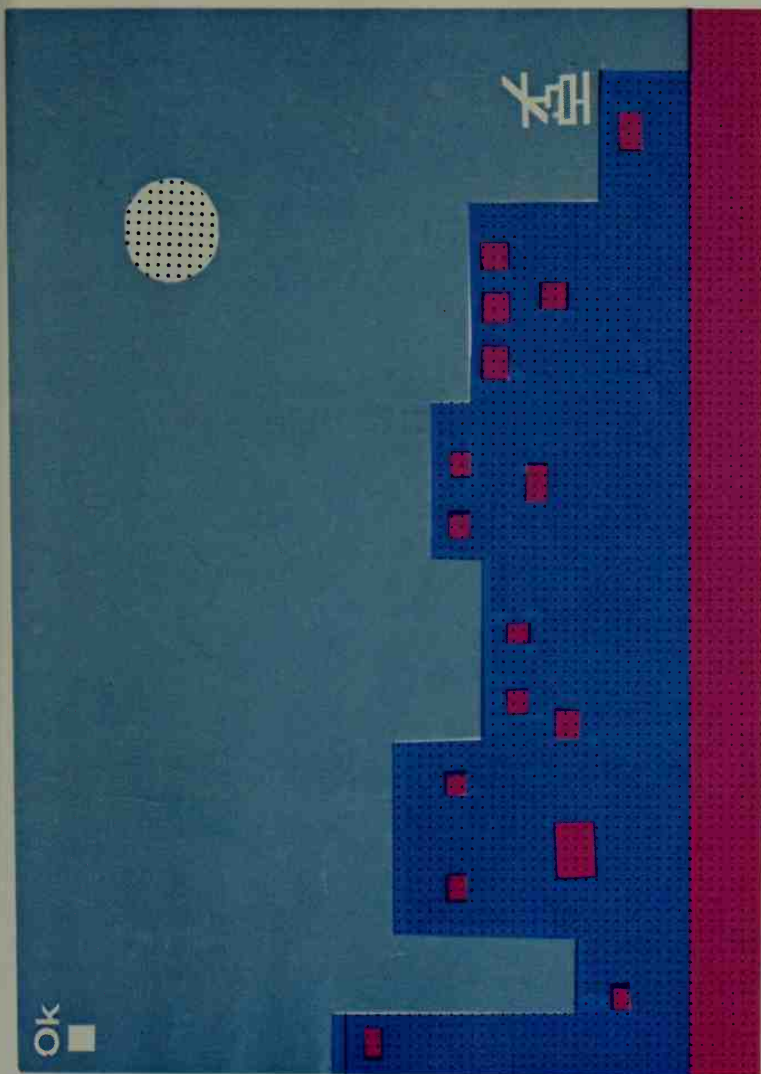
EXERCISE 4 from page 113



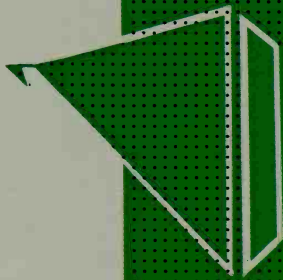
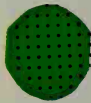
EXERCISE 9 from page 116

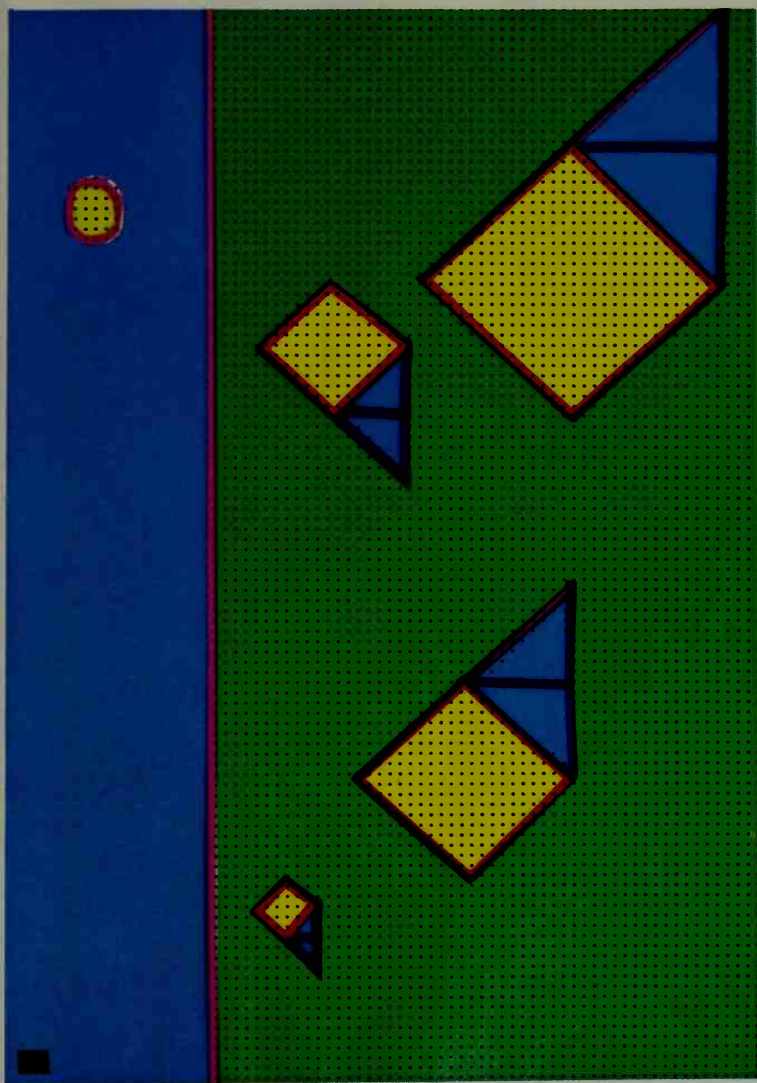






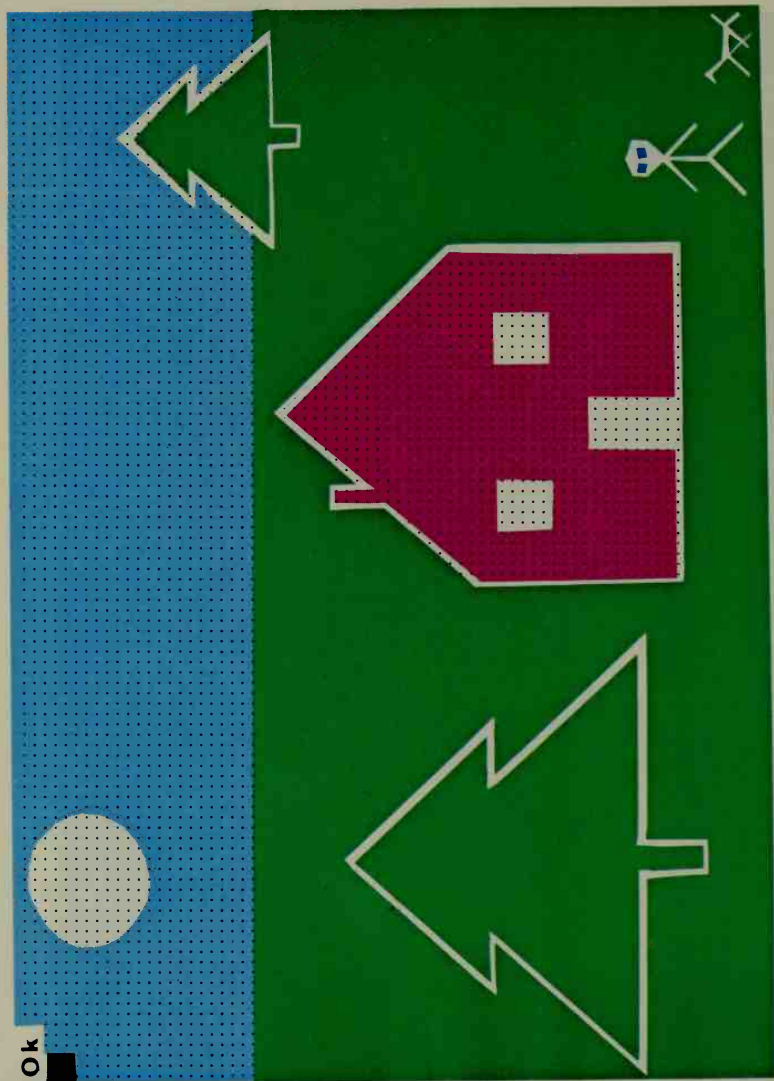
HELICOPTER *from page 119*





EVENING CAMPGROUND from page 125

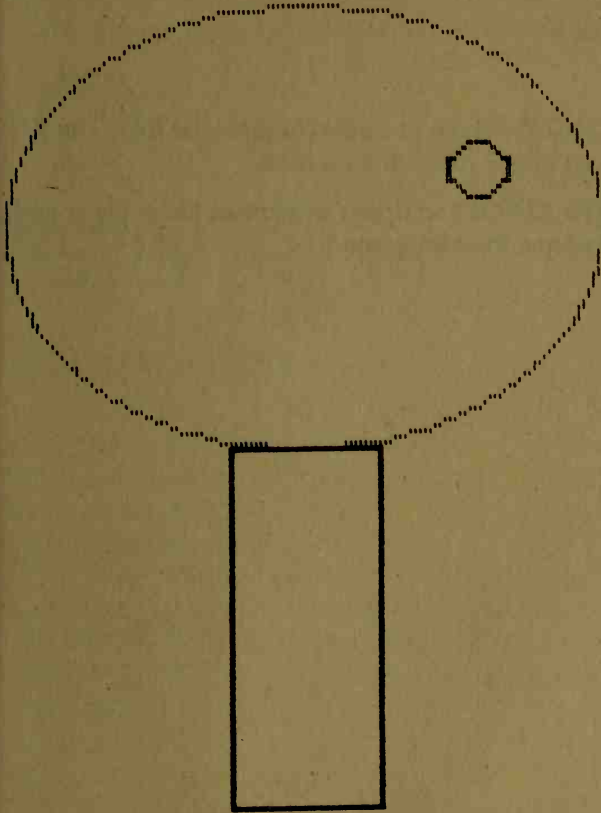
Ok



APPLE TREE

Run this program and watch what happens to the apple.

Try putting more apples on the tree, or make other objects fall out of the tree, such as a branch or acorn.



PROGRAM LISTING

```
100 SCREEN 1: KEY OFF: COLOR 8,0: CLS
110 DIM A(26)
120 CIRCLE(145,60),60,1,,,2/3
```

```
130 LINE(130,100)-(160,165),,B 'Trunk of tree
140 CIRCLE(180,50),6,2 'Draw apple
150 GET(174,45)-(186,55),A
160 I=45
170 PUT(174,I),A,XOR 'Erase apple
180 I=I+1
190 PUT(174,I),A,XOR 'Draw apple
200 IF I<150 THEN 170
```

NOTES

Line 130: The CIRCLE statement is used to draw the tree. The 2/3 makes the figure an ellipse rather than a circle.

Lines 170,190: The XOR action draws and erases the apple at each location without erasing the background.

WALKING FIGURE

Make a simple cartoon character walk across the screen.

To make your own figure, just draw your character in 2 or more walking positions and then alternate the positions as you move your figure across the screen.

Try making an animal walk or run across the screen.



PROGRAM LISTING

```
100 SCREEN 1: KEY OFF: COLOR 9,1: CLS
110 DIM F(200),G(200)
120 C=3:X=30:Y=50
130 GOSUB 210
140 GOSUB 310
150 FOR X=30 TO 250 STEP 15
160 PUT(X,50),F,PSET ' Draw Fig 1
170 FOR I=1 TO 50: NEXT I
180 PUT(X,50),G,PSET ' Draw Fig 2
190 NEXT X
200 STOP
210 CIRCLE(X,Y),10,C 'FIG 1
220 LINE(X-5,Y+8)-(X-15,Y+20),C
230 LINE-(X+15,Y+20),C
240 LINE-(X+5,Y+8),C
250 LINE(X-5,Y+20)-(X,Y+25),C,B
260 LINE(X,Y+20)-(X+5,Y+25),C,B
270 LINE(X-5,Y+25)-(X+10,Y+30),C,B
```

```

280 GET (0 , 40) - (70, 80) , F
290 CLS
300 RETURN
310 CIRCLE (X, Y) , 10, C 'FIG 2
320 LINE (X-5, Y+8) - (X-15, Y+20) , C
330 LINE - (X+15, Y+20) , C
340 LINE - (X+5, Y+8) , C
350 LINE (X-5, Y+20) - (X-10, Y+25) , C, B
360 LINE (X-10, Y+25) - (X, Y+30) , C, B
370 LINE (X+10, Y+20) - (X+15, Y+25) , C, B
380 LINE (X+10, Y+25) - (X+20, Y+30) , C, B
390 GET (0 , 40) - (50, 95) , G
400 CLS
410 RETURN

```

NOTES

Lines 130, 140: The drawings of the figure are placed in subroutines starting on Lines 210 and 310. This helps the programmer separate the action part of the program from the initial drawings.

Line 180: Since there is no background in this picture, the PSET action can be used to draw and erase the figure.

Line 200: The STOP statement terminates the program execution and returns the computer to the command level.

Lines 290, 400: The CLS statement can be used to erase the image after it is stored in the array, since there are no other objects on the screen.

FLYING BIRD

Make a bird fly across the screen. This program uses three positions of the bird's wings in order to simulate motion.

Try making a bird or a butterfly of your own.



PROGRAM LISTING

```
100 SCREEN 1: KEY OFF: COLOR 9,1: CLS
110 DIM B(140),C(140),D(140)
120 Y=50
130 GOSUB 200 'GET BIRD
140 FOR X=1 TO 200 STEP 4: MAKE BIRD MOVE
150 PUT(X,Y),B,PSET
160 PUT(X,Y),C,PSET
170 PUT(X,Y),D,PSET
180 NEXT X
190 STOP
200 LINE (15,40)-(40,30),3: Bird fig 1
210 LINE -(50,50),3
220 LINE -(60,30),3
230 LINE -(85,40),3
240 GET(4,28)-(88,52),B
250 CLS
260 LINE (10,35)-(40,35),3: Bird fig 2
270 LINE -(50,50),3
280 LINE -(60,35),3
290 LINE -(85,35),3
300 GET(4,28)-(88,52),C
310 CLS
320 LINE (10,30)-(50,50),3: Bird fig 3
```

```
330 LINE -(85,30),3
340 GET(4,28)-(88,52),D
350 CLS
360 RETURN
```

NOTES

Lines 150, 160, 170: Each PUT statement displays a different position of the bird's wings.

Lines 240, 300, 340: The GET statement works faster if the X coordinate is evenly divisible by 4.

JUMPING FISH

This fish swims across the screen and also jumps out of the water until you hit the CTLBREAK key.

Change the speed he swims by changing the step increments on Lines 150 and 250. But watch out—if you move the fish too far each time, the old image may not be erased.



PROGRAM LISTING

```
100 SCREEN 1: KEY OFF: COLOR 9,1: CLS
110 DIM F(260),S(260)
120 GOSUB 300
130 LINE(0,50)-(280,50) ' Draw water line
140 YC=80
150 FOR I=1 TO 100 STEP 10 ' Move fish
160 PUT(I,YC),F,PSET
170 NEXT I
180 FOR Y=YC TO YC-75 STEP -5 ' Jump up
190 PUT(I,Y),F,PSET
200 NEXT Y
210 FOR Y=YC-75 TO YC STEP 5 ' Jump down
220 PUT (I,Y),F,PSET
230 NEXT Y
240 LINE(0,50)-(280,50) ' Redraw water
250 FOR J=I TO 200 STEP 10 ' Move fish
260 PUT(J,YC),F,PSET
270 NEXT J
280 PUT(J,YC),S,PSET ' Erase last fish
290 GOTO 150
300 PSET(65,150),3
310 DRAW"C3;M96,140;M110,150;M96,158;M62,145;M65,150"
320 PSET(103,148),2
330 GET(40,120)-(120,168),F
370 CLS
380 GET(40,120)-(120,168),S
390 RETURN
```

NOTES

Lines 150-170: Move the fish across the screen.

Lines 180-200: Make the fish jump up.

Lines 210-230: Make the fish jump down.

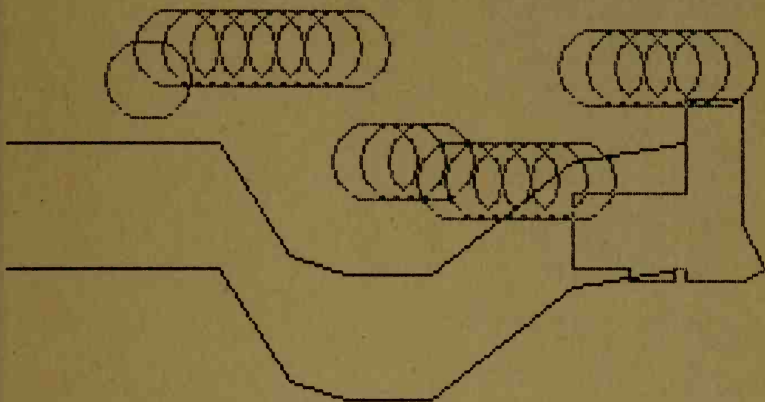
Lines 250-270: Move the fish the rest of the way across the screen.

Line 280: Erases the last drawing of the fish at the end of the screen. Since the next location will be at the left of the screen, this one does not get erased by a PUT statement.

Line 380: Stores a blank rectangle in the array S. This is another useful way to erase an image.

STEAM ENGINE

This train moves along a track, leaving a trail of smoke.
Can you draw a car and move it along a road?



PROGRAM LISTING

```
100 SCREEN 1: KEY OFF: COLOR 9,1: CLS
110 DIM A(400),XT(8),YT(7)
120 X=30: Y=50: C=3
130 GOSUB 320 'Draw train
140 GOSUB 590 'Get location of tracks
150 GOSUB 500 'Draw tracks
160 X=XT(1) 'Move train
170 Y=YT(1)-10
180 K=1
190 XR=X+50
200 YR=Y-10
210 PUT(X,Y),A 'Draw train
220 CIRCLE(XR,YR),15,2 'Smoke
230 XS=X
240 YS=Y
250 X=X+10
260 IF X<=XT(K) THEN 300
270 K=K+1
280 IF K>6 THEN END
290 Y=YT(K)-20
300 PUT(XS,YS),A 'erase train
310 GOTO 190
320 C=2
```



```

330 LINE-(X,Y)-(X+40,Y),C 'Draw train
340 LINE-(X+40,Y-30),C
350 LINE-(X+60,Y-30),C
360 LINE-(X+60,Y+10),C
370 LINE-(X+68,Y+24),C
380 LINE-(X+60,Y+28),C
390 LINE-(X+40,Y+28),C
400 LINE-(X+40,Y+24),C
410 LINE-(X+36,Y+24),C
420 LINE-(X+36,Y+28),C
430 LINE-(X+20,Y+28),C
440 LINE-(X+20,Y+24),C
450 LINE-(X,Y+24),C
460 LINE-(X,Y),C
470 GET(X,Y-30)-(X+70,Y+50),A
480 PUT(X,Y-30),A,XOR
490 RETURN
500 PSET(XT(1),YT(1))
510 FOR I=2 TO 7
520 LINE-(XT(I),YT(I))
530 NEXT I
540 PSET(XT(1),YT(1)+40)
550 FOR I=2 TO 7
560 LINE-(XT(I),YT(I)+40)
570 NEXT I
580 RETURN
590 FOR I=1 TO 7 'location of tracks
600 READ XT(I),YT(I)
610 NEXT I
620 RETURN
630 DATA 0,44,75,44,100,80,120,86,150,86,200,50,240,44

```

NOTES

Lines 190-310: Move the train along the track. Since the track is not straight, the X,Y coordinates for the train are in arrays XT and YT. Line 250 increments the X coordinate 10 spaces at a time. Line 290 gets the Y coordinate from YT. Line 300 erases the train at the old location.

Lines 500-580: This subroutine draws the tracks, using the X,Y coordinates in arrays XT,YT. The LINE statement allows the use of arrays for the X,Y coordinate.

Lines 590-620: READ the X,Y coordinates of the tracks from the

DATA statement on Line 630. This information is then used to draw the tracks and guide the train.

6 COLOR AND PAINT

The COLOR statement sets the color options available in Text mode and Medium Resolution graphics mode only.

PAINT fills in an area of the screen with a color in graphics modes only (SCREEN 1 or SCREEN 2).

PROGRAMS

1. Warmups
2. Helicopter
3. Moonlight Sail
4. Evening Campground
5. Orchard Scene
6. Painted Picture

Note: If you have a monochrome or black-and-white monitor, you can still use some of the COLOR statement options. Refer to your system's BASIC reference manual for a description of these.

WARMUPS

COLOR

Although the IBM PC has 16 colors available, your options are limited by the screen mode and the visibility of your choices on the screen.

In Text mode, the format of the COLOR statement is:

COLOR, F, B, E where:

F: foreground character color number:

Dark Shades	Light Shades
0 black	8 gray
1 blue	9 light blue
2 green	10 light green
3 cyan (bright blue)	11 light cyan
4 red	12 light red
5 magenta	13 light magenta
6 brown	14 yellow
7 white	15 bright white

16-31 produce blinking characters in corresponding shades (example: 16 = blinking black, 17 = blinking blue, etc.)

B: background color. Only 0-7 can be used.

E: border of the screen. Colors 0-16 are available.

Colors and intensity may vary depending on your monitor.

The foreground color may equal the background color. This has the effect of making a character invisible. Changing the foreground or background color will make the character visible again. This method can be used to simulate character animation.

Any parameter may be omitted from the COLOR statement. Omitted parameters assume the old value of the previous COLOR statement executed.

In Text mode it is necessary to follow the COLOR statement with CLS to "wash" your screen with the background color you've

selected. In order to get the desired effect, we've established this program initialization convention for our programs' first statement:

```
100 SCREEN mode:KEY OFF:COLOR statement:CLS
```

If you use a WIDTH statement in your program, insert it before the COLOR statement, since it resets the screen mode.

EXERCISE 1

Print a message in yellow on a blue screen with a green border.

```
HOW DO YOU LIKE THIS?
```

```
10 SCREEN 0:KEY OFF:COLOR 14,1,10:CLS
20 LOCATE 12,10
30 PRINT"HOW DO YOU LIKE THIS?"
```

Notice that only the background color is within the parameters of the Text mode (24 rows, 40 or 80 columns). The border is "outside" the Text screen.

Change the COLOR statement to experiment with other possibilities.

EXERCISE 2

Now watch your message disappear.

```
10 SCREEN 0:KEY OFF:COLOR 1,1,10:CLS
20 LOCATE 12,10
30 PRINT"HOW DO YOU LIKE THIS?"
```

In Medium Resolution graphics mode, the COLOR statement sets the actual colors used by PSET, PRESET, LINE, CIRCLE, PAINT, and DRAW. The format of the COLOR statement in SCREEN 1 is:

COLOR B, P:

B: background color number in the range of 0-15.

P: selects palette of colors, 1 or 0, which can be used by the graphics statements (LINE, DRAW, CIRCLE, etc.). The palette and color correspondence is:

Palette 0	Palette 1	Color Number
green	cyan	1
red	magenta	2
brown	white	3

For example, if you want to draw a green fish in blue water, you should select a background color of blue (we'll choose 9), with the palette which has green: Palette 0. Your color statement is: COLOR 9,0, and your DRAW statements contain the color command C1 to outline the fish.

The color options 1-3 in each of the graphics statements select the color from the palette you've chosen.

The color selected for background in the COLOR statement can be the same as any of the palette colors.

EXERCISE 3

Draw a red circle against a blue background.



```
10 SCREEN 1:KEY OFF:COLOR 9,0:CLS
20 CIRCLE(160,100),25,2
```

Experiment with the COLOR statement and the color option of the CIRCLE statement to see which color combinations you like best.

What happens if you outline your figure in the same color as the background?

```
10 SCREEN 1:KEY OFF:COLOR 12,0:CLS
20 CIRCLE(160,100),25,2
```

The COLOR statement chose red (12) as the background and Palette 0. The CIRCLE statement used red (2) from palette 0, so the circle is invisible.

EXERCISE 4

Now see what happens if you use all the colors from the palette in your graphics statements.



```
10 SCREEN 1:KEY OFF:COLOR 9,0:CLS
20 LINE(20,20)-(60,60),1,BF
30 LINE(100,100)-(120,140),2,BF
40 LINE(180,40)-(240,80),3,BF
50 CIRCLE(240,150),20,0
```

Why doesn't the circle appear on the screen? Change CIRCLE's color variable to 3 and Run the program again. Now you should see it.



```
10 SCREEN 1:KEY OFF:COLOR 9,0:CLS
20 LINE (20,20)-(60,60),1,BF
30 LINE (100,100)-(120,140),2,BF
40 LINE (180,40)-(240,80),3,BF
50 CIRCLE (240,150),20,3
```

PAINT

PAINT fills in an area of the screen with a color, allowing scenes to be painted with very few statements. It can be used in Medium Resolution or High Resolution graphics modes. The format is:

PAINT (X1,Y1), P, B where:

X1, Y1: are coordinates of a point within the area to be painted.

P: color to PAINT with, ranging from 0-3.

B: color of the edges of the figure to be filled in, ranging from 0-3.

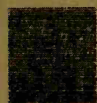
The figure to be filled in usually is drawn with a color boundary. If boundary is not specified, then the foreground color is used (3 in Medium Resolution, 1 in High Resolution).

The figure to be filled in must be totally enclosed by a color boundary. Otherwise, the PAINT statement will PAINT the entire screen.

Since there are only two colors in High Resolution, it doesn't make sense for the PAINT color to be different from the boundary color. If the boundary color is omitted, it will default to equal the PAINT color.

EXERCISE 5

If you don't specify a color to draw the box, the computer defaults to color 3, the foreground (in this case, white). You can PAINT the box, omitting the color options and defaulting to color 3, as long as the figure was drawn with the same color number.



```
10 SCREEN 1:KEY OFF:COLOR 9,1:CLS
20 LINE(150,90)-(170,110),,B
30 PAINT(151,91)
```

EXERCISE 6

```
10 SCREEN 1:KEY OFF:COLOR 9,1:CLS
20 LINE(150,90)-(170,110),2,B
30 PAINT(151,91)
```

In this case, the box was drawn with color 2 (magenta). Since this color number is different from PAINT's default color number 3 (white), the entire screen PAINTed white. No boundaries were specified.

EXERCISE 7

Here the box is drawn with color 3, so PAINT fills only the box with white.



```
10 SCREEN 1:KEY OFF:COLOR 9,1:CLS
20 LINE(150,90)-(170,110),3,B
30 PAINT(151,91)
```

EXERCISE 8

Another way to fill the box with white (color 3), is to draw the box in a different color number (here, 2) and PAINT to that boundary number.



```
10 SCREEN 1:KEY OFF:COLOR 9,1:CLS
20 LINE(150,90)-(170,110),2,B
30 PAINT(151,91),,2
```

EXERCISE 9

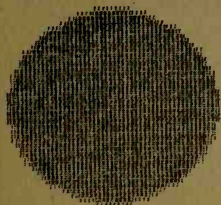
You can PAINT any figure in the same color it is drawn and omit the boundary option. This program fills a green circle with green.



```
10 SCREEN 1:KEY OFF:COLOR 9,0:CLS
20 CIRCLE(160,100),25,1
30 PAINT(158,98),1
```

EXERCISE 10

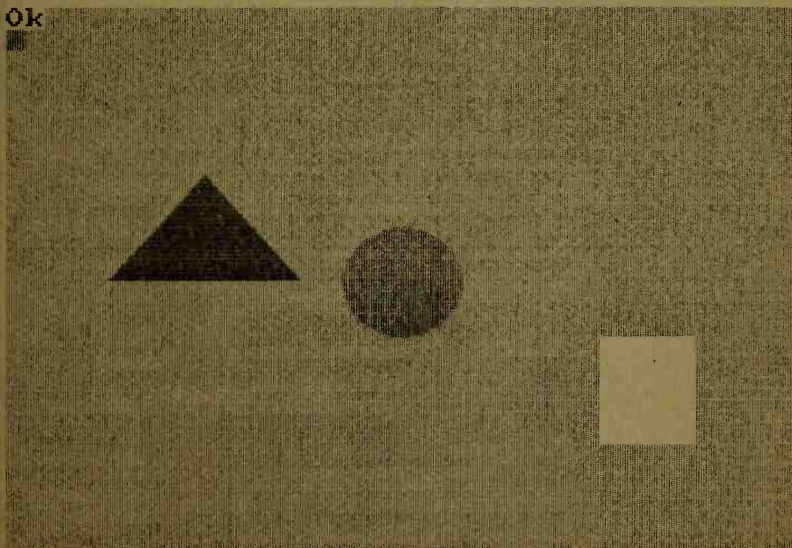
If you want to fill the figure with a different color than the one it is drawn in, specify the boundary in the PAINT statement.



```
10 SCREEN 1:KEY OFF:COLOR 9,0:CLS
20 CIRCLE(160,100),25,1
30 PAINT(158,98),2,1
```

EXERCISE 11

Ultimately, you have 4 color options with which to PAINT, using any given COLOR statement in Medium Resolution graphics mode.



```
10 SCREEN 1:KEY OFF:COLOR 9,0:CLS
20 CIRCLE(160,100),25,1
30 PAINT(158,98),2,1
40 LINE(40,100)-(80,60),1
50 LINE-(120,100),1
60 LINE-(40,100),1
70 PAINT(60,95),3,1
80 LINE(240,120)-(280,160),1,B
90 PAINT(250,130),0,1
100 PAINT(0,0),1,1
```

In this exercise:

Line 10: Background color 9 (blue), and Palette 0 are used.

Line 20: A circle is drawn in color number 1 (green) from Palette 0.

Line 30: PAINT fills the circle with color 2 (red) to the boundary.

Lines 40-60: Draw a triangle with color 1.

Line 70: Fills it with color 3.

Line 80: Draws a box in color 1.

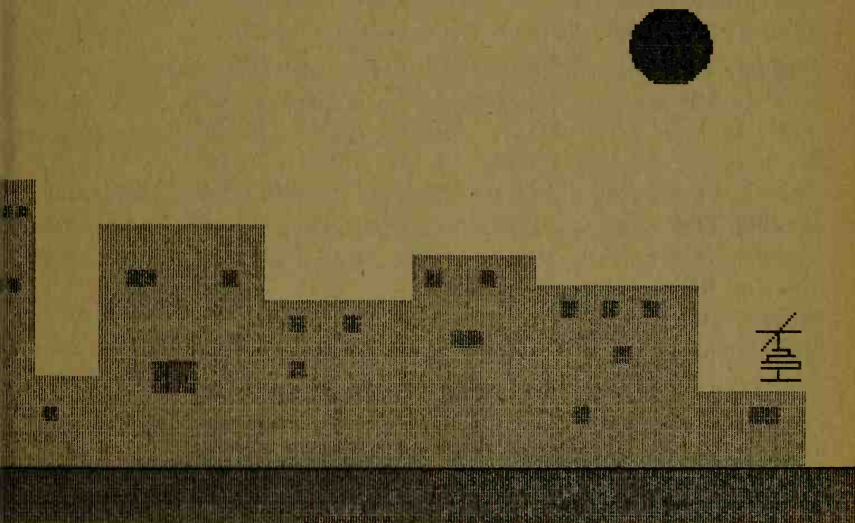
Line 90: Fills it with the background color, blue (0).

Line 100: Fills the entire screen with color 1, to the boundaries of any figures drawn in color 1.

HELICOPTER

Watch your computer draw a helicopter flying above a city skyline and landing on a building, in this program.

Once you get this program up and running, experiment with the COLOR and PAINT statements to see different color effects.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 8,1:CLS
110 DIM A(200)
120 LET Y=180 'skyline
130 LINE(0,85)-(16,Y),1,BF
140 LINE(16,150)-(40,Y),1,BF
150 LINE(40,100)-(100,Y),1,BF
160 LINE(100,125)-(155,Y),1,BF
```



```

170 LINE(155,110)-(200,Y),1,BF
180 LINE(200,120)-(260,Y),1,BF
190 LINE(260,155)-(300,Y),1,BF
200 LINE(4,94)-(7,97),2,BF'windows
210 LINE(10,94)-(13,97),2,BF'bld.#1
220 LINE(7,118)-(10,122),2,BF
230 LINE(20,160)-(24,164),2,BF '#2
240 LINE(50,115)-(60,120),2,BF '#3
250 LINE(85,115)-(90,120),2,BF
260 LINE(60,145)-(75,155),2,BF
270 LINE(110,130)-(115,135),2,BF '#4
280 LINE(130,130)-(135,135),2,BF
290 LINE(110,145)-(115,150),2,BF
300 LINE(160,115)-(165,120),2,BF '#5
310 LINE(180,115)-(185,120),2,BF
320 LINE(170,135)-(180,140),2,BF
330 LINE(210,125)-(215,130),2,BF '#6
340 LINE(225,125)-(230,130),2,BF
350 LINE(240,125)-(245,130),2,BF
360 LINE(215,160)-(220,165),2,BF
370 LINE(230,140)-(235,145),2,BF
380 LINE(280,160)-(290,165),2,BF
390 LET X1=20:LET Y1=40'helicopter
400 PSET (X1,Y1)
410 DRAW "S8"
420 DRAW "C3"
430 DRAW"NE3;NR4;NL4;NG3;D3;L1;R3;D1;R1;
    L5;D1;L1;R7;D1;L4;NL3;D2;L3;R7""copter
440 GET (0,20)-(40,57),A
450 PUT (0,20),A,XOR
460 FOR X1=10 TO 270 STEP 10
470 PUT(X1,Y1),A,PSET
480 NEXT X1
490 FOR W=1 TO 100
500 NEXT W
510 X1=270
520 FOR Y1=20 TO 115 STEP 5

```

```
530 PUT (X1,Y1),A,PSET
540 NEXT Y1
550 CIRCLE (250,40),15,3 'moon
560 PAINT(255,35),3
570 LINE(0,180)-(319,180),3'sidewalk
580 PAINT(10,195),2,3
```

NOTES

Line 100: The COLOR statement in Medium Resolution for the program selects gray (8) for the background night sky and Palette 1 for the graphic statements' color options.

Lines 130-190: In Palette 1, the color variable 1 of the LINE statement is cyan (bright blue), so the buildings are drawn with the Box option and filled in with cyan.

Lines 200-380: Using the color variable 2 in the LINE statement, the windows are drawn with the Box option and filled with color 2 from Palette 1, magenta.

Line 420: The color command of the DRAW statement uses the variable 3. This is white on Palette 1, so the helicopter is drawn in white. If 0 is chosen, the background color, gray, is used and the helicopter is invisible.

Line 550: The moon is outlined in color 3. From Palette 1, this is white.

Line 560: The X,Y coordinates of the PAINT statement are set within the area of the moon. The color variable 3 uses white from Palette 1 to PAINT the moon out to the boundary color executed in the CIRCLE statement.

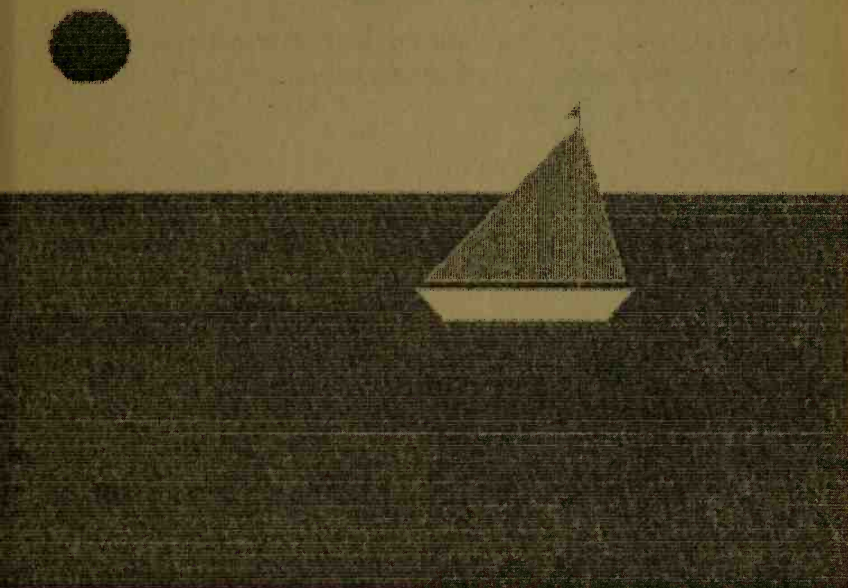
Line 570: This LINE statement is used to set a boundary for the subsequent PAINT statement. You must use the color option (in this case 3, or white), to draw the Line so the PAINT statement will stop at this border. If the color parameter is omitted, the subsequent statement will go on and PAINT the entire screen.

Line 580: This final PAINT statement fills in the sidewalk area with color 2 (magenta on Palette 1) to the border 3 (white on Palette 1). If the sidewalk LINE in Line 570 did not completely delineate the border, (for example, if it went from X,Y of 5,180 to 319,180), the PAINT statement would fill everything in the screen with magenta (color 2) except those items drawn in white (color 3).

Use this program to try various color possibilities with the COLOR and PAINT statements. Then alter other graphics statements' color variables and see what you get. Start by changing the COLOR statement in Line 100 to COLOR 8,1. Run the program and see what happens. Then try COLOR 9,0, and so on.

MOONLIGHT SAIL

Here is the Sailboat program from Chapter 4 with PAINT statements added to the program listing. Now the sails and the boat are colored and the moon and water filled in.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,0:CLS
110 LET BOW$="C1;BM160,100;R80;G10;L60;H10"
120 LET MAIN$="C1;BM220,100;U55;M163,97;M220,97"
130 LET JIB$="C1;BM220,45;M237,97;M220,97"
140 LET FLAG$="C1;BM220,45;U7;G5;R5"
150 DRAW BOW$:DRAW MAIN$:DRAW JIB$:DRAW FLAG$
160 PAINT(215,95),2,1'paint main
170 PAINT(225,95),2,1'paint jib
180 PAINT(217,42),2,1'paint jib
190 DRAW "C1;BMO,68;R195;BM228,68;R92"'horizon
200 PAINT (0,90),3,1 'water
```

```
210 CIRCLE (40,20),15,1  'moon  
220 PAINT (42,22),3,1
```

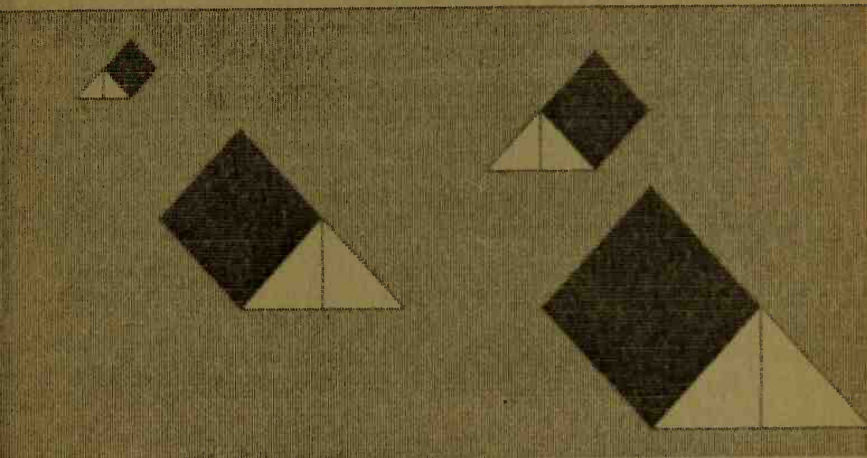
NOTES

Lines 110-140, 190, 210: The sails, bow, flag, horizon, and moon are all drawn in color 1 (green on Pallet 0).

Lines 160-180, 200, 220: All the PAINT statements fill in with specified colors out to the boundaries drawn in color 1.

EVENING CAMPGROUND

The Campground program from Chapter 4 has PAINT statements in this program listing, which color the tents, the grass, and the moon.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,0:CLS
110 LET FRONT$="C2;NG20;ND20;F20;L40"
120 LET SIDES$="C2;E20;F20;G20;H20"
130 LET A$="A0"  "angle of sides
140 LET A2$="A2"
150 FOR X=1 TO 4
160 ON X GOSUB 190,230,270,310
170 NEXT X
180 GOTO 430.
190 LET START$="EM40,70"  "position #1
200 DRAW "S2":GOSUB 350
210 PAINT (45,70),3,2
220 RETURN
```

```

230 LET START$="BM120,120" 'position #2
240 DRAW "S6":GOSUB 390
250 PAINT (115,120),3,2
260 RETURN
270 LET START$="BM200,85" 'position #3
280 DRAW "S4":GOSUB 350
290 PAINT (205,85),3,2
300 RETURN
310 LET START$="BM280,150" 'position #4
320 DRAW "S8":GOSUB 390
330 PAINT (275,150),3,2
340 RETURN
350 'tent routine for #1 & #3
360 DRAW START$:DRAW FRONT$:DRAW START$
370 DRAW A$:DRAW SIDES$
380 RETURN
390 'tent routine for #2 & #4
400 DRAW START$:DRAW FRONT$:DRAW START$
410 DRAW A2$:DRAW SIDES$:DRAW A$
420 RETURN
430 LINE (0,51)-(319,51),2
440 PAINT(3,54),1,2 'paint grass
450 CIRCLE (260,20),10,2 'moon
460 PAINT (265,25),3,2

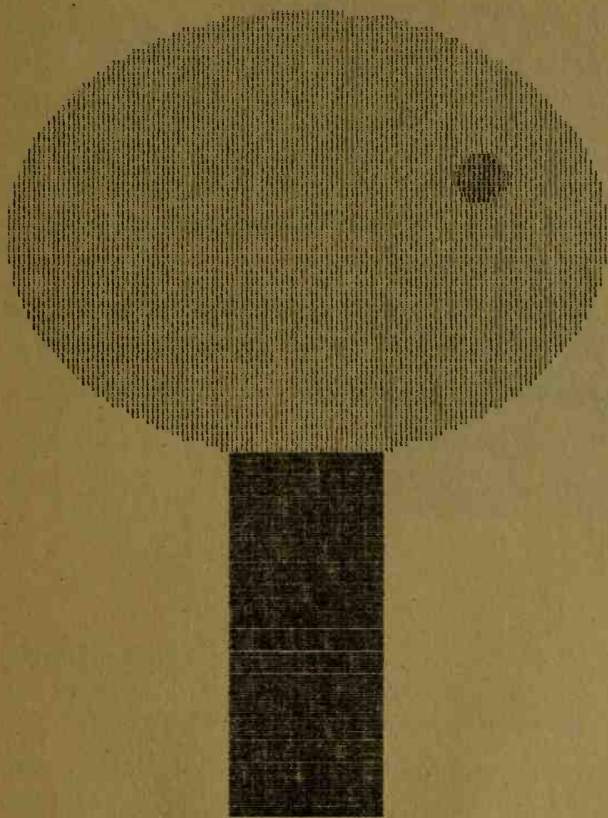
```

NOTE

This program uses the same technique as Moonlight Sail. It draws all the figures with the same color number from the Palette and then PAINTs out to that color-number boundary.

ORCHARD SCENE

Add PAINT statements to the Apple Tree program from Chapter 5 and fill in the tree and apple with color.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 9,0:CLS
110 DIM A(26)
120 CIRCLE(180,50),6,1  'Draw apple
130 PAINT(180,50),2,1
```

```

140 GET(174,45)-(186,55),A
150 CIRCLE(145,60),60,1,,,2/3
160 PAINT(145,60),1,1
170 LINE(130,100)-(160,165),,B 'Trunk of tree
180 PAINT(150,150),3,3
190 I=45
200 PUT(174,I),A,XOR 'Erase apple
210 I=I+1
220 PUT(174,I),A,XOR 'Draw apple
230 IF I<150 THEN 200

```

NOTES

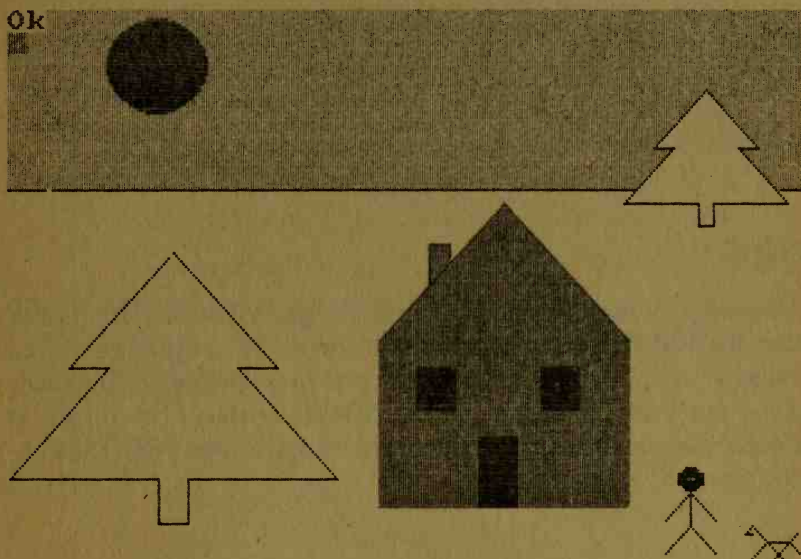
Line 130: This PAINT statement fills in the apple with red (color 2) and gives it a green border (color 1).

Line 160: Fills in the top of the tree with green. Since the apple has a green border, this PAINT statement doesn't cover over the apple.

Line 180: The trunk of the tree is PAINTed with brown (color 3).

PAINTED PICTURE

Picture from Chapter 4 is now filled with color using PAINT and the Box Fill option of the LINE statement in the program listing.



PROGRAM LISTING

```
100 SCREEN 1:KEY OFF:COLOR 10,1:CLS
110 LET TREE$= "C3;G20;R8;G20;R30;D8;R6;U8;
    R30;H20;R8;H21"
120 DRAW "BM65,90"position lrg. tree
130 DRAW "S8 X TREE$;"
140 DRAW "BM280,30"position sm. tree
150 DRAW "S4 X TREE$;"
160 LINE (150,120)-(250,180),2,BF"house
170 DRAW "C3;BM150,120;E50;F50" "roo
180 PAINT(155,117),2,3
190 LINE (165,130)-(180,145),3,BF"l.win
200 LINE (190,155)-(205,180),3,BF"door
210 LINE (215,130)-(230,145),3,BF"r.win
```

```

220 DRAW "C3;BM170,100;U15;R8;D5""chimney
230 PAINT (175,90),2,3
240 CIRCLE (275,170),5,3'man's head
250 PAINT (276,172),3,3
260 PSET (274,170),1
270 PSET (276,170),1
280 DRAW "C3;BM275,175;NF10;NG10;D12;NF10;NG10""man
290 DRAW "BM300,190""dog's position
300 DRAW "C3;L3;E3;F6;NG6;NF6;R9;NG6;NF6;E3" 'dog
310 DRAW"C3;BM0,65;R253;BM311,65;R8""horizon
320 CIRCLE (60,20),20,3'sun
330 PAINT (60,19),3,3
340 PAINT (0,0),1,3

```

NOTE

This program uses a background color of green (number 10), rather than the PAINT statement, to color the trees and grass. This is done because the parameter of the large tree isn't completely enclosed, using the Scale command of the DRAW statement. If you try to PAINT that tree, the PAINT statement will go on and PAINT the entire screen.

7 FUN WITH GEOMETRY

You can use the built-in functions SIN and COS to draw many geometric shapes, such as a square, triangle, or circle. You can even make flowers and fancy designs.

PROGRAMS

1. Warmups
2. Apple Tree Harvest
3. Leaping Fish
4. Bird and Flower
5. Fishing Boat
6. Fancy Graphics
7. Ski Game

WARMUPS

EXERCISE 1

Use the SIN function to draw a wave across the top of your screen.



```
100 SCREEN 1:CLS:KEY OFF
110 H=10: D=20: C=1: A=0: B=20
120 PSET(A,D)
130 FOR X=A TO B STEP C
140 Y=SIN(X)*H+D
150 LINE--(X*H,Y)
160 NEXT X
```

The numeric variables on Line 110 are used as follows:

- H: height of wave.
- D: location on Y axis.
- A: beginning point on X axis.
- B: last point on X axis.
- C: size of step along X axis.

Experiment with different values for these variables and see how the wave changes.

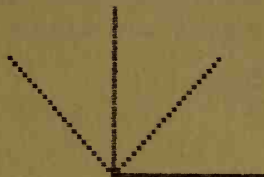
Replace the SIN function with the COS function on Line 140 and see how the wave changes.

EXERCISE 2

We can use the SIN and COS functions to draw any angle we want. This example draws 0 degrees, 45 degrees, 90 degrees, and 135 degrees.

INPUT LENGTH OF LINE,ANGLE

```
? 30,0  
? 30,45  
? 30,90  
? 30,135  
? ■
```



```
100 CLS: SCREEN 1: KEY OFF  
110 PRINT"INPUT LENGTH OF LINE,ANGLE"  
120 INPUT R,A  
130 AR=A*(3.14159/180)  
140 X1=R*COS(AR)+100  
150 Y1=-R*SIN(AR)+100  
160 LINE(100,100)-(X1,Y1)  
170 GOTO 120
```

Experiment with different values of R and A and see what angles you make.

Line 130 converts the angle A from degrees to radians using the equation:

$$\text{Radians} = \text{Degrees} * (\text{PI}/180) \quad \text{—for PI} = 3.14159$$

Lines 140 and 150 determine the ending X,Y coordinate for the line and add these values to the point (100,100). This translates the (0,0) point from the top left point on the screen to (100,100).

Line 150 uses a negative R because the Y coordinate on your screen is the reverse of the Y coordinate on a normal X,Y axis. The Y coordinate on your screen increases as you move down rather than up. Therefore, you must change the sign of Y in order to make it the same as a normal X,Y axis.

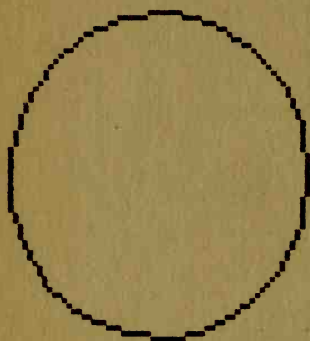
EXERCISE 3

Draw a circle, using the SIN and COS functions. These equations:

$$X = R * \cos(I) + XC$$

$$Y = -R * \sin(I) + YC$$

give the X,Y coordinates for any point on the circumference of a circle with a radius of R and center at XC,YC.



```
100 SCREEN 1: CLS: KEY OFF
110 XC=100: YC=75: R=30: S=0: E=6.6: J=.3
120 PSET(XC+R,YC)
130 FOR I=S TO E STEP J
140 X=R*COS(I)+XC
150 Y=-R*SIN(I)+YC
160 LINE-(X,Y)
170 NEXT I
```

The numeric variables on Line 110 are used as follows:

- XC: center of circle X-axis.
- YC: center of circle Y-axis.
- R: radius of circle.
- S: radians to begin drawing.
- E: radians to end drawing.
- J: increment to step while drawing.

S and E determine which part of the circle is drawn. To make a complete circle, set $S=0$ and $E=6.6$. To draw a part of the circle, change S or E to any number between 0 and 6.6. Experiment with different combinations for S and E and see which part of the circle you make. Just be sure to make E larger than S

Change the value of J and see how it affects your circle.

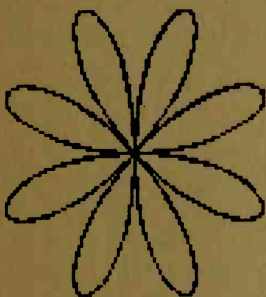
To make an ellipse, make R different for X than Y.

```
150 Y=-1.5*R*COS(I)+YC
```

Vary the shape of the ellipse by changing the radius R for X and Y.

EXERCISE 4

Draw a flower using the SIN and COS functions.



```
100 SCREEN 1: CLS: KEY OFF
110 N=4: A=50: PI=3.14159
120 M=N*PI*2.5
130 PSET(100,100)
140 FOR I=0 TO M STEP .05
150 R=A*SIN(N*I)
160 X=R*COS(I)+100
170 Y=-R*SIN(I)+100
180 LINE-(X,Y)
190 NEXT I
```

The numeric variables on Line 110 are used as follows:

A: size of the flower.

N: number of petals drawn.

If N is even, flower has $2 * N$ petals.

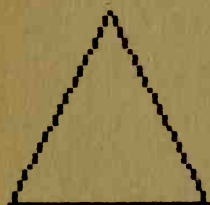
If N is odd, flower has N petals.

Change the value of N and see what your flower looks like. Try using a decimal number for N and see what you get.

Line 150 uses an equation called "N-leaved rose" to find a value for R. Notice that R changes each time the FOR . . . NEXT loop executes.

EXERCISE 5

We can use these equations to draw geometric shapes also. To make a triangle, we need to draw 3 sides, turning 120 degrees each time.



```
100 SCREEN 1:CLS
110 A=120*(3.14159/180)
120 R=40: AR=A
130 X2=R*COS(AR)+100
140 Y2=-R*SIN(AR)+100
150 LINE(100,100)-(X2,Y2)'Draw side 1
160 AR=AR+A: X1=X2: Y1=Y2
170 X2=R*COS(AR)+X1
180 Y2=-R*SIN(AR)+Y1
190 LINE(X1,Y1)-(X2,Y2)'Draw side 2
200 LINE(X2,Y2)-(100,100)'Draw side 3
```

The numeric variables are used as follows:

A: angle in degrees.
 AR: angle in radians.
 R: length of side.
 X1,Y1: X,Y coordinates for start of line.
 X2,Y2: X,Y coordinates for end of line.

Lines 130,140 and 170, 180 add the new values of X,Y to the starting coordinates X1,Y1. This moves the origin each time to the starting point of the line.

EXERCISE 6

To draw a square, draw 4 sides and turn 90 degrees.



```

100 CLS: SCREEN 1
110 A=360/4
120 AR=A*(3.14159/180)
130 X1=150: Y1=190: R=20
140 PSET(X1,Y1)
150 FOR I=0 TO 3
160 X2=R*COS(AR*I)+X1
170 Y2=-R*SIN(AR*I)+Y1
180 LINE-(X2,Y2)
190 X1=X2: Y1=Y2
200 NEXT I
  
```

This program is a more efficient version of Exercise 5. The sides are drawn inside the FOR . . . NEXT loop on Lines 150-200.

EXERCISE 7

Now you can draw a N-sided shape just by changing Exercise 6 as follows:


```
105 INPUT N
110 A=360/N
150 FOR I=0 TO N-1
```

Try different values of N and see what shapes you make. How large must N be before your shape looks like a circle?

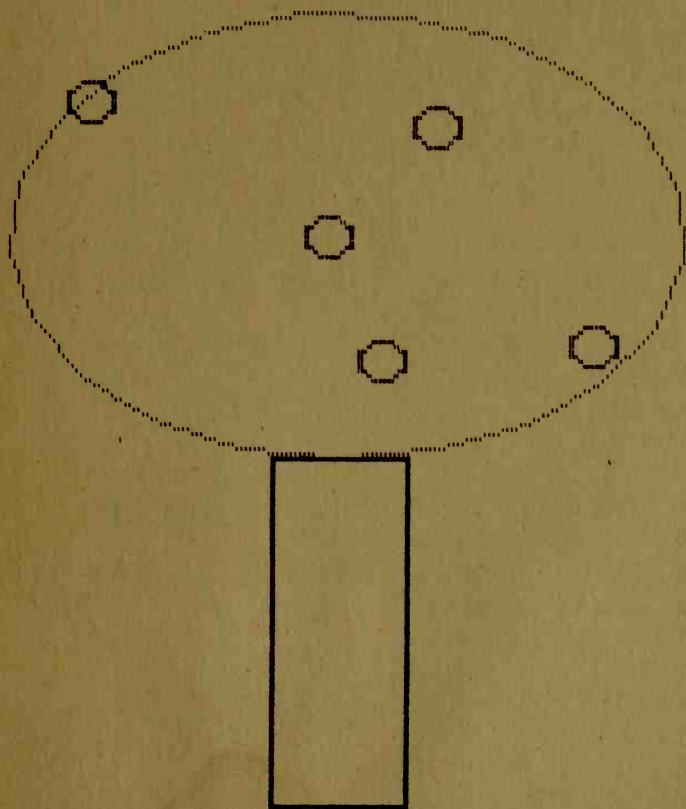
Now try using these ideas in some programs.

APPLE TREE HARVEST

Use the equations for an ellipse to draw the apple tree from Chapter 5. This tree starts out with 10 apples, but some fall off.

Create your own picture by adding other figures to this one.

Change Line 250 to put more apples on the tree. Can you make them all fall off?



PROGRAM LISTING

```
100 SCREEN 1: KEY OFF: COLOR 9,0: CLS
110 DIM AP(30),X(10),Y(10)
120 RANDOMIZE
130 CLS
140 CIRCLE(7,7),5,2 'Draw one apple
150 GET(0,0)-(12,12),AP
160 PUT(0,0),AP,XOR
170 RX=75: RY=45: XC=152: YC=55
180 PSET(XC,RY+YC)
190 FOR I=-1.6 TO 4.9 STEP .3 'Tree
200 A=RX*COS(I)+XC
210 B=-RY*SIN(I)+YC
220 LINE-(A,B),1
230 NEXT I
240 LINE(A-15,B)-(A+15,B+70),3,B'Trunk
250 FOR C=1 TO 10 'Put on apples
260 A=INT(RND*95+105)
270 B=INT(RND*65+20)
280 PUT(A,B),AP
290 X(C)=A
300 Y(C)=B
310 NEXT C
320 FOR C=10 TO 1 STEP -2'Apples fall
330 PUT(X(C),Y(C)),AP,XOR
340 FOR D=Y(C) TO 170 STEP 3
350 PUT(X(C),D),AP
360 PUT(X(C),D),AP
370 NEXT D
380 PUT(X(C),D),AP'Draw apple on ground
390 NEXT C
```

NOTES

Lines 140-160: Draw an apple and store it in array A.

Lines 190-230: Draw an ellipse for the tree.

Lines 250-310: Put apples on the tree and save the X,Y coordinates for each apple in arrays X and Y. Line 260 finds a random X location for the apple within the circumference of the ellipse. Line 270 finds a random Y location.

Lines 320-370: Move every other apple to the bottom of the screen.

LEAPING FISH

Make our fish program from Chapter 5 more realistic by changing the line to a wave and making the fish jump in an arc instead of straight up.

Add some other fish or sea life and create your own underwater picture.



PROGRAM LISTING

```
100 SCREEN 1: KEY OFF: COLOR 9,1: CLS
110 DIM F(260),S(260)
120 GOSUB 390
130 GOSUB 320
140 PI=3.14159:R=50
150 YC=80
160 FOR XC=0 TO 100 STEP 10 'Move fish
170 PUT(XC,YC),F,PSET
180 NEXT XC
190 PUT(100,80),F,XOR
200 FOR I=3 TO 0 STEP -.3 'Jump up
210 X=R*COS(I)+XC
220 Y=-R*SIN(I)*1.5+YC
230 PUT(X,Y),F,XOR
240 FOR K=1 TO 50 : NEXT K
250 PUT(X,Y),F,XOR
260 NEXT I
```

```

270 FOR XC=X TO 200 STEP 10 'Move fish
280 PUT(XC,YC),F,PSET
290 NEXT XC
300 PUT(XC,YC),S,PSET
310 GOTO 150
320 ' Draw wave across top of screen
330 PSET(1,50)
340 FOR X=1 TO 20 STEP 1
350 Y=SIN(X)*10+50
360 LINE-(X*14,Y),1
370 NEXT X
380 RETURN
390 PSET(65,150),3 'Draw fish
400 DRAW"C3;M96,140;M110,150;M96,158;
    M62,145;M65,150"
410 PSET(103,148),2
420 PSET(103,148),2
430 GET(40,120)-(120,168),F
440 CLS
450 GET(40,120)-(120,168),S
460 RETURN

```

NOTES

Lines 200-260: Use the equations for a circle to make the fish jump in an arc.

Line 240: This Wait Loop keeps the image of the fish on the screen longer and reduces the flicker.

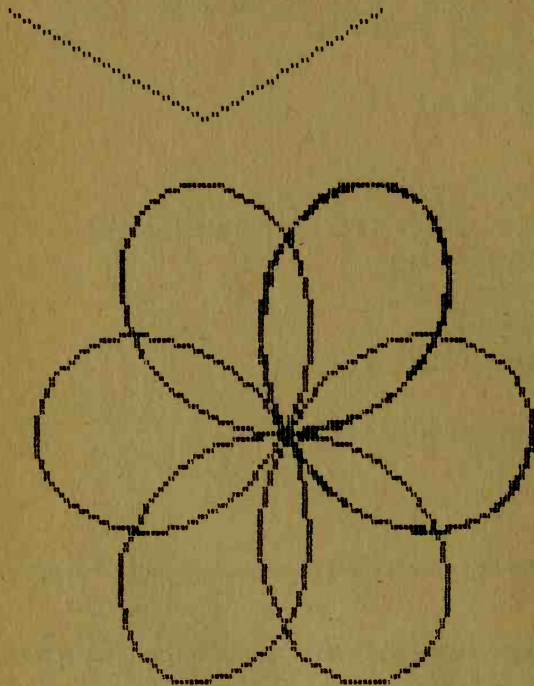
Lines 330-370: Use the SIN function to draw a wave across the screen.

Lines 390-430: Draw the fish.

BIRD AND FLOWER

Make the bird from Chapter 5, fly to a flower.

Change the value of N on Line 130 and see how the flower changes.



PROGRAM LISTING

```
100 SCREEN 1: KEY OFF: COLOR 9,1: CLS
110 DIM B(300),C(300),D(300)
120 X=150: Y=145: PI=3.14159
130 N=1.5
140 PSET(X,Y)
150 FOR I=0 TO 5*PI STEP .1 'Draw flower
```

```

160 R=50*SIN(N*I)
170 C=R*COS(I)+X
180 D=-R*SIN(I)+Y
190 LINE-(C,D),2
200 NEXT I
210 GOSUB 300 'GET BIRD
220 B=30
230 FOR X=1 TO 90 STEP 4
240 Y=(6/15)*X+B
250 PUT(X,Y),B,PSET
260 PUT(X,Y),C,PSET
270 PUT(X,Y),D,PSET
280 NEXT X
290 STOP
300 LINE (15,40)-(40,30),1'Bird fig 1
310 LINE -(50,50),1
320 LINE -(60,30),1
330 LINE -(85,40),1
340 GET(5,28)-(88,52),B
350 PUT(5,28),B
360 LINE (10,35)-(40,35),1'Bird fig 2
370 LINE -(50,50),1
380 LINE -(60,35),1
390 LINE -(85,35),1
400 GET(5,28)-(88,52),C
410 PUT(5,28),C
420 LINE (10,30)-(50,50),1'Bird fig 3
430 LINE -(85,30),1
440 GET(5,28)-(88,52),D
450 PUT(5,28),D
460 RETURN

```

NOTE

Lines 230-280: Move the bird in a straight line from the point (1,30)

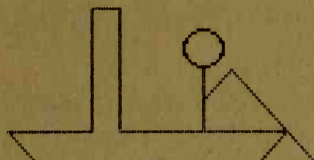
to the top of the flower. Line 240 calculates the Y coordinate using the equation of a straight line, as follows:

$Y = M * X + B$ where M = slope of line, B = Y intercept.
In this case, M = 6/15 and B = 30.

FISHING BOAT

Watch the waves move under this fishing boat.

Can you change the shape of the boat, or add some swimming fish to the picture?



PROGRAM LISTING

```
100 SCREEN 1: KEY OFF: COLOR 9,1: CLS
110 DIM S(374)
120 GOSUB 190
130 GOSUB 270
140 FOR I=0 TO 3 'Make wave move
150 PUT(I,150+I),S,PSET
160 FOR J=1 TO 50: NEXT J
170 NEXT I
180 GOTO 140
190 PSET(0,150) 'Draw wave
200 FOR X=0 TO 20 STEP .5
210 Y=SIN(X)*10+150
220 LINE-(X*14,Y),1
230 NEXT X
240 GET(0,140)-(280,160),S
250 CLS
260 RETURN
270 'Draw man in boat
280 DRAW "BM100,140" 'Position boat
```

```

290 DRAW "C2;R30;U40;R10;D40;R30;U20;D10;
    E10;F60" 'Top
300 CIRCLE (170,113),7,3 'Man's head
310 DRAW "BM170,140" 'Position stern
320 DRAW"C2;R30G20L60H20" 'Bottom of boat
330 RETURN

```

NOTES

Lines 120, 130: Call subroutines which draw the wave and the boat. This makes it easier to change a figure used in a graphics program.

Lines 140-170: Move the wave under the boat by changing the X coordinate used in the PUT command.

Lines 190-240: Draw the wave, using the SIN function, and store it in array S.

Lines 270-320: Draw the fishing boat and the man fishing.

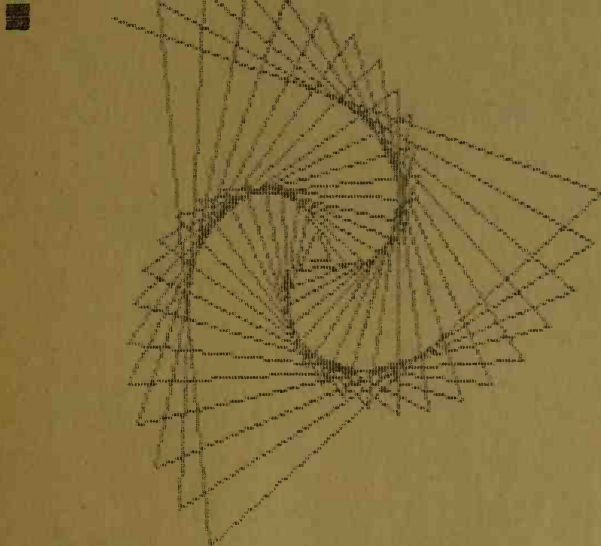
FANCY GRAPHICS

Make your own spirals and fancy designs.

Experiment with different input values and see what fancy designs you can make.

```
INPUT SIDE,ANGLE,INCREMENT,COLOR
? 7,123,3,2
```

```
Ok
```



PROGRAM LISTING

```
100 CLS: SCREEN 1:KEY OFF: COLOR 9,1
110 PRINT"INPUT SIDE,ANGLE,INCREMENT,COLOR"
120 INPUT R,A,D,C
130 N=360/A
140 A=A*(3.14159/180)
150 PSET(100,100)
160 X1=100: Y1=100
170 I=0
```

```

180 X2=R*COS(A*I)+X1
190 Y2=-R*SIN(A*I)+Y1
200 IF X2<0 OR X2>320 THEN END
210 IF Y2<0 OR Y2>190 THEN END
220 LINE-(X2,Y2),C
230 X1=X2: Y1=Y2
240 R=R+D
250 I=I+1
260 GOTO 180

```

NOTES

Line 140: Converts angle A to radians.

Lines 180, 190: Determine next X,Y coordinates.

Lines 200, 210: Stop the program when the design reaches the limits of the screen.

Line 240: Increments the length of side by D.

SKI GAME

A skier is moving down the slope between the trees. See if you can guide him safely through using the right (→) and left (←) arrow keys on your keyboard.



PROGRAM LISTING

```
100 SCREEN 1: KEY OFF: COLOR 9,0: CLS
105 RANDOMIZE
110 DIM C1(65),C2(65),C3(65),C4(65)
115 DIM S(377),YT(5),XT(5)
120 X1=35: Y1=30: SX=100: P=10: SY=20: IC=45
125 GOSUB 305
130 GOSUB 525
135 GOSUB 475
140 PUT(SX,SY),S : PX=SX: PY=SY
145 FOR K=1 TO 200: NEXT K
150 ' Check for keyboard input
155 GOSUB 190
160 SY=SY+6
165 IF SY>130 THEN GOSUB 290
170 GOSUB 220 ' move skier
175 FOR I=1 TO 100: NEXT I
```

```

180 GOTO 155
185 ' Keyboard input routine
190 A$=MID$(INKEY$,2)
195 IF A$="" THEN RETURN
200 SY=SY-6
205 IF A$=CHR$(75) THEN SX=SX-P
210 IF A$=CHR$(77) THEN SX=SX+P
215 RETURN
220 ' Move skier left or right
225 IF SX>10 AND SX<220 THEN 245
230 LINE(SX,SY)-(SX+75,SY+75),0,BF
235 IF SX<10 THEN SX=220
240 IF SX>220 THEN SX=10
245 PUT(PX,PY),S 'erase old image
250 PUT(SX,SY),S
255 PX=SX: PY=SY
260 FOR I=1 TO 5
265 IF YT(I)>SY+IC OR YT(I)+YI<SY THEN 280
270 IF SX+IC <XT(I) OR SX>XT(I)+XI THEN 280
275 GOTO 385
280 NEXT I
285 RETURN
290 PRINT"YOU MADE IT - CONGRATULATIONS"
295 GOTO 615
300 ' Draw crash symbol
305 CIRCLE(160,100),20
310 C$= "NR30;NE20;NU30"
315 DRAW C$
320 GET(160,70)-(190,100),C1
325 DRAW "A1":
330 DRAW "BM160,100 X C$;"
335 GET(130,70)-(160,100),C2
340 DRAW "A2"
345 DRAW "BM160,100 X C$;"
350 GET(130,100)-(160,130),C3
355 DRAW "A3"
360 DRAW "BM160,100 X C$;"
365 GET(160,100)-(190,130),C4
370 CLS
375 RETURN

```



```

380 ' Display crash symbol
385 LOCATE 5,5: PRINT"CRASH"
390 IF SY < 30 THEN SY=30
395 X4= SX+20: Y4=SY+20: X1=X4
400 Y1=Y4-30: X2=X4-30: Y2=Y4-30
405 X3=X4-30: Y3=Y4
410 FOR K=1 TO 4
415 PUT(X1,Y1),C1,OR
420 PUT(X2,Y2),C2,OR
425 PUT(X3,Y3),C3,OR
430 PUT(X4,Y4),C4,OR
435 FOR J=1 TO 100: NEXT J
440 X1=X1+6: Y1=Y1-6
445 X2=X2-6: Y2=Y2-6
450 X3=X3-6: Y3=Y3+6
455 X4=X4+6: Y4=Y4+6
460 NEXT K
465 GOTO 615
470 'Draw trees
475 TREE$="A0;C1;G20;R8;G20;R30;D8;R6;U8;R30;
      H20;R8;H21"
480 FOR X=1 TO 5
485 T=INT(RND*280)+10
490 C=INT(RND*155)+10
495 YT(X)=C
500 XT(X)=T-30
505 PSET(T,C)
510 DRAW "S3 X TREE$;"
515 NEXT X
520 RETURN
525 'DRAW SKIER
530 CIRCLE (150,63),3,2 'head
535 LINE (150,65)-(150,78),2 'torso
540 LINE -(145,100),2 'l. leg
545 LINE (150,78)-(155,100),2 'r. leg
550 LINE (150,68)-(146,68),2 'l. arm
555 LINE -(143,75),2
560 LINE -(135,70),2
565 LINE (136,68)-(125,100),2 'l. pole
570 LINE (150,68)-(154,68),2 'r. arm
575 LINE -(157,75),2

```

```

580 LINE -(165,70),2
585 LINE (166,68)-(155,100),2      'r. pole
590 LINE (140,96)-(157,110),2      'l. ski
595 LINE (150,96)-(167,110),2      'r. ski
600 GET(112,55)-(180,120),5
605 CLS
610 RETURN
615 END

```

NOTES

Lines 155-180: This part of the program actually controls the entire game by calling the subroutines which check for input from the keyboard and move the skier. The program loops through these lines until the skier reaches the bottom, or collides with a tree.

Lines 190-215: Check for input from the keyboard. The computer sets INKEY\$ equal to the key struck. Line 190 puts the second character of INKEY\$ into A\$. IF the key struck was a left arrow, Line 205 deducts 10 from SX. If the key was a right arrow, Line 210 adds 10 to SX.

Lines 225-255: Move the skier to a new location. Line 230 uses the LINE statement to erase the old location. Lines 235 and 240 check for a location off the screen and reset X to the opposite side.

Lines 260-280: Check for a collision with one of the 5 trees. The collision subroutine is called if the Y coordinate of the rectangle containing the skier is between the top and bottom of a tree and the X between the left and right of a tree.

Lines 305-365: Draw a crash cartoon and store each portion in an array C1,C2, etc.

Lines 385-465: PUT these arrays at the spot the skier crashes. Since the PUT statement uses the X,Y coordinates of the top left corner of a rectangle, each of the 4 rectangles need different X,Y coordinates, X1,X2, etc.

Lines 475-520: Draw 5 trees at random locations on the screen.

Lines 530-600: Draw a skier and save it in array S.

SUMMARY OF TERMS USED IN IBM TEXT

TERM	DEFINITION	EXAMPLE	FIRST USED
array	In BASIC, a set of sub-scripted variables.	A, B etc.	p. 88
CHR\$(65)	Returns the character corresponding to the ASCII code in the argument.	CHR\$(65) = A	p. 38
CIRCLE	Draws an ellipse at a specified location with a given radius.	CIRCLE(20,80),5	p. 64
CLS	Clears the screen in either Direct or Deferred mode.	CLS	p. 16
COLOR	Sets the colors for foreground, background, and border screen	COLOR 9,0	p. 110
cursor	Marker or symbol on the screen that marks where the user's next action will take effect or where the next character typed from the keyboard will appear.		p. 29
COS	Returns the trigonometric cosine of the argument.	COS(5) = .2836624	p. 131
DATA	Holds information within a program which is accessed with a READ statement.	DATA 7,12,4	p. 106
DELETE	Erases specified lines from a program.	DELETE 120	p. 92
DIM A(20)	Sets aside space for a string variable, or for an array (here called A).	DIM A(20)	p. 91

DOS	The IBM PC's Disk Operating System (DOS) is a collection of programs to create and manage files, run programs, and use the system devices attached to the computer. BASIC and BASICA are stored on DOS.		p. 7
DRAW	A graphics mode statement that draws line segments using a graphics definition language.	DRAW"R20"	p. 62
END	Terminates the execution of the program and returns control to the user.	END	p. 35
FOR I = 1 TO 4 . . .	Allows repeating of program lines between the FOR and NEXT statement four times.	FOR I = 1 TO 4 . . .	p. 26
NEXT I		NEXT I	
STEP	The loop of lines is incremented by the number in the STEP statement. If there is no STEP statement, the loop is incremented by 1.	FOR I = 4 TO 1 STEP -1 . . .	p. 37
GET	In graphics mode, stores a shape already on the screen into an array.	NEXT I GET(50,60) - (140,80)	p. 88
GOSUB 250	Executes the subroutine which begins at line 250.	GOSUB 250	p. 35
GOTO 200	Branches to line 200.	GOTO 200	p. 35

IF . . . THEN	Executes or skips one or more statements, depending on the truth of the stated condition. In this example IF A = 10, THEN line 200 is executed next. Otherwise the line after the IF . . . THEN statement is executed.	IF A = 10 THEN 200	p. 44
IF . . . THEN . . . ELSE	Executes or skips one or more statements, depending on the truth of the stated condition. In this example IF A = 10, THEN line 200 is executed next. Otherwise line 240 is executed.	IF A = 10 THEN 200 ELSE 240	p. 56
INKEY\$	Checks for input from the keyboard. The computer sets INKEY\$ equal to the key struck.	X\$ = INKEY\$	p. 154
INPUT A	Reads a line of input from the current input device.	INPUT A	p. 44
INT(9.5)	Returns the largest whole number less than or equal to the argument.	INT(9.5) = 9	p. 43
KEY OFF	Removes IBM BASIC's "Soft Key" display statement from row 25 of the display.	KEY OFF	p. 14
LINE	A graphics statement that draws a line on the display at specified locations.	LINE(20,100)-(80,100)	p. 46

LOCATE	Tells the computer in which row and column to position the cursor on the screen. The cursor does not actually move until an input or output command which uses the screen is issued.	LOCATE 12,20	p. 30
MID\$ as a function	As a function, returns the requested part of a given string.	X\$ = MID\$(INKEY\$,1,3)	p. 152
ON . . . GOSUB	Branches the program to one of several subroutines depending on the value of an expression, in this example X. IF X = 2, the second line number in the list will be executed. If X = 3, the third line number, etc.	ON X GOSUB 100,200,300,400	p. 80
PAINT	Fills an area of the screen with a selected color.	PAINT(52,20),2	p. 114
PRESET	When used in conjunction with PSET, erases a point at a specified location on the screen.	PRESET(40,80)	p. 47
PRINT	Writes a line of output to the current display device. This example writes the value of A\$, B, and C.	PRINT A\$,B,C	p. 14
PSET	Plots a point at a specified location on the screen.	PSET (40,80)	p. 47
PUT	In graphics mode, displays on the screen the shape stored in an array by GET.	PUT (50,60)	p. 89

RANDOMIZE	Used in conjunction with the RND function, allows a different series of random numbers each time the program runs by asking for a "Random Number Seed" input.	RANDOMIZE	p. 43
READ	Calls for input of information stored in DATA statements within a program.	READ A,B	p. 106
REM or '	Includes remarks in the body of the program for the benefit of the user.	' This is a remark	p. 41
RETURN	Returns control from a subroutine to the statement following the GOSUB that called the subroutine.	RETURN	p. 35
RND	Returns a random number between 0 and 1.	RND(1)	p. 43
SCREEN function	As a function, returns the ASCII code for the character on the screen at the specified row and column.	S-SCREEN(5,10)	p. 44
SCREEN statement	As a statement, selects one of the three IBM PC's display modes.	SCREEN 0	p. 15
SIN	Returns the trigonometric sine of the argument.	SIN(2) = .90929	p. 131
STOP	Terminates the program execution and returns to the command level.	STOP	p. 100

string variable	String variables in BASIC are allowed to store up to 255 characters. The name can be up to 40 characters. They must start with a letter and end with a \$ sign.	A\$, ARM\$	p. 78
TAB	Tells the computer the column in which to start the following PRINT statement.	TAB(20)	p. 35
WIDTH	Sets the number of columns in IBM PC's display modes.	WIDTH 80	p. 15

COMPUTER CREATIONS WITH YOUR



Once you've explored the many capabilities of your IBM PC, you'll definitely want to take the next step to graphics. This book covers the beginning and intermediate graphics programming statements available in BASIC. By following the exercises, you'll be able to decorate a Christmas tree, build a pyramid, play hide and seek, and create an amazing amount of complex and beautiful artwork. And after you've familiarized yourself with these programs, you can experiment and generate your own new and unique designs.

BASIC FUN WITH GRAPHICS

includes an introduction, an explanation on how to use this book, and warm up exercises at the beginning of each chapter. It also has a chapter that focuses on the color capabilities of your computer.

BASIC FUN WITH GRAPHICS

is written in BASIC for the IBM PC equipped with a color/graphics board. The programs will run on either a color or monochrome monitor.

The possibilities, discoveries, and fun are limitless. All you have to do is sit down at your computer and start creating!

AVON



CAMELOT

PUBLISHED BY AVON BOOKS

012-UP

0



ISBN 0-553-85068-0