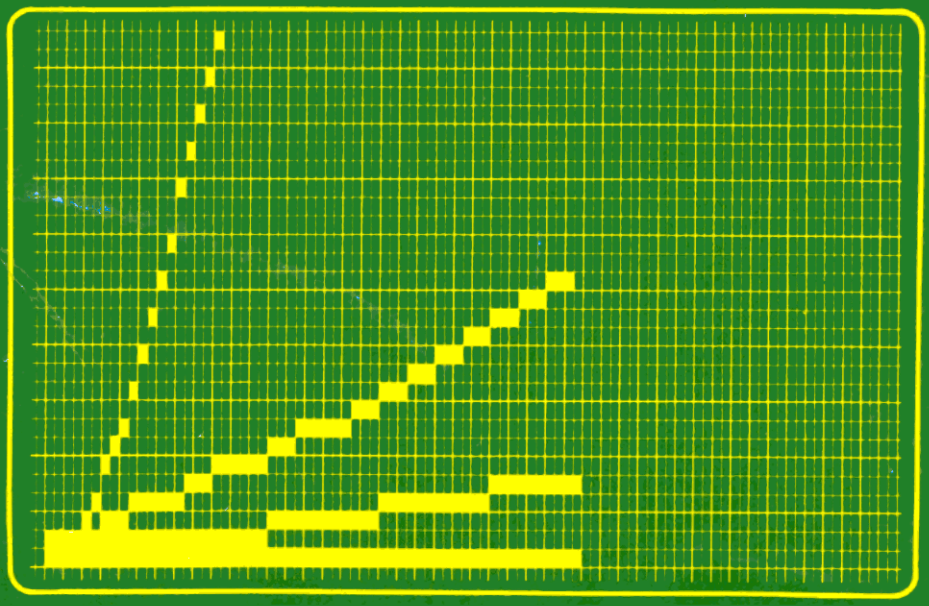
Introduction to TRS-80 Graphics

Don Inman



Introduction to TRS-80 GRAPHICS

Introduction to
TRS-80
GRAPHICS

Don Inman

dilithium Press
Beaverton, Oregon

10 9 8 7

ISBN: 0-918398-18-5
Library of Congress catalog card number: 78-24835

Printed in the United States of America.

dilithium Press
P.O. Box 606
Beaverton, Oregon 97075

PREFACE

The video display is one of the most significant characteristics of the inexpensive personal computer such as the Radio Shack TRS-80. The ability of the user to create images and alter them as he wishes provides an astounding number and variety of uses which were unheard of in the past. In order to make full use of the advantages of the video display over printed copy, you must investigate the manner in which data may be manipulated to create ever-changing pictures on the screen.

The purpose of this book is to introduce you to the capabilities of TRS-80 graphics and to stimulate you to make further investigations yourself. It is intended for fun and entertainment, but it also has educational values. It is *not* intended as a complete course in BASIC language programming.

Although Level I BASIC (an ideal vehicle for beginners) was used in writing this book originally, an appendix has been added to provide Level II BASIC compatibility. Level I BASIC has all the necessary statements and commands for learning the fundamentals of programming, but some modifications are necessary to run some of the programs in Level II. These modifications with comments are shown in the appendix.

Explanations are provided throughout the book for statements and commands used to create the graphics displays. It is assumed that you have a TRS-80 Reference Manual and will use it when necessary.

CONTENTS

INTRODUCTION

The material presented is designed to be used for self-instruction. You should enter and run each program on your own computer. Exercises and answers are provided to apply what you learn. Try them! Suggestions for extensions of key ideas are given at the end of most chapters. Experiment with the extensions.

The programs are written in a straightforward style with logically connected portions separated from other portions. Abbreviations and multiple-line statements are avoided (except for a few examples) so that each program line will stand by itself. This makes the programs easy to read and understand. The programs are not perfect in style or content, but are meant as a basis for you to learn to write your own programs.

1

WHERE ARE WE GOING?

Our goal is to provide some informal assistance in understanding the graphics capabilities of the TRS-80 computer using Level I BASIC. Numerous examples are given to illustrate the techniques discussed. Exercises are provided for you to work out, and numerous additional suggestions for practice are given at the end of each chapter.

No attempt is made to teach programming in BASIC. There are many, many books on the market about programming — some are good and others are not. We will assume you have some programming experience and have read the TRS-80 Level I User's Manual. We will demonstrate the TRS-80 graphics through statements and commands explained in the user's manual.

Chapter two begins with the statements: SET(X,Y), RESET(X,Y), and PRINT AT to show how individual points may be turned on and off and how characters may be printed at any location on the video display. You then progress to a more detailed explanation of manipulating plotted points to draw rectangles. This technique is used to demonstrate games and pictures that will stir your imagination. The use of line drawings to form graphs and charts is explored. We then move from vertical and horizontal lines to the formation of lines at oblique angles, triangles, and finally, curved lines.

By this time, you will be able to display complex figures, and possible applications will burst forth from you imagination with no limitation.

Programs begin with a few simple statements of a few lines, but you will quickly build up the capability of combining short program segments into more complex programs.

Let's begin with a peek at computer graphics in general and then examine the specifics of the TRS-80 system.

GRAPHICS

A graphic video display allows you to draw pictures in addition to printing the usual typewriter characters. In general, black and white picture capability is less expensive than color pictures. Somewhere in between (in cost) is the capability to display one or more shades of grey along with black and white.

You will find a wide range in price for graphics capability in various computers or in various display devices which may be added to a computer. The price range for the displays range from under one hundred dollars to several thousands of dollars. Why such a wide range in price? It is largely a function of the method used and the resolution provided. Careful thought must be given to how "good" a picture you need and how much you are willing to pay for picture quality.

Without going into a technical description, let's take a look at two common methods used to draw pictures on the display. Both methods make use of straight lines to achieve their objective.

Method I

The video screen is divided into a grid of columns and rows as shown in Figure 1.

The finer the division of the screen (into more rows and more columns), the better the *resolution* of the picture will be.

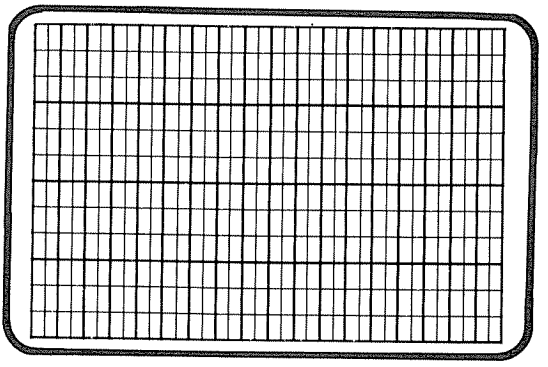


Figure 1

Displays of this kind are able to "light up" (or turn on) any given rectangle (or square) on the screen. They can also "turn off" any given rectangle on the screen.

A display which is divided into 256 columns and 256 rows (256 x 256) is capable of "better" pictures than one which is divided into 128 columns and 128 rows (128 x 128).

For example: a circle attempted in 3 columns and 3 rows compared to a circle attempted in 6 columns and 6 rows.

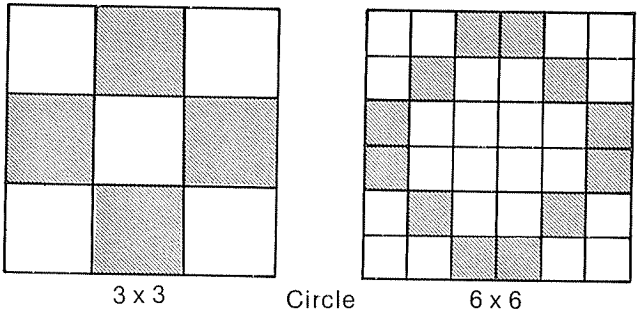


Figure 2

The 6 x 6 (even though far from perfect) more closely resembles a circle than the 3 x 3 display.

Another example: a triangle, on a 4 x 4 grid compared to a triangle on an 8 x 8 grid.

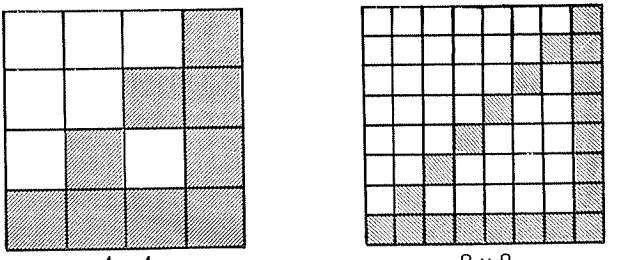


Figure 3

Some imagination must be used in the 4 x 4 display in order to recognize the triangle. The 8 x 8 display is more reasonable.

Figure 3 also shows that the result of drawing a diagonal line leaves something to be desired. This method, however, is easily adapted to today's TV sets and inexpensive monitors.

It stands to reason that, given the same size display, the price will rise as some function of the number of columns and rows which a display is capable of producing.

Method II

The video screen is divided into a grid of points for this method. It is similar to the grid of the first method, but you should now consider only the centers of the rectangles.

Now instead of lighting up individual rectangles, we can select any two points and draw a line between them. This would be called drawing a *vector.*

Once again, the more points provided in the grid, the better the picture will be. But here too, the better picture costs more to produce.

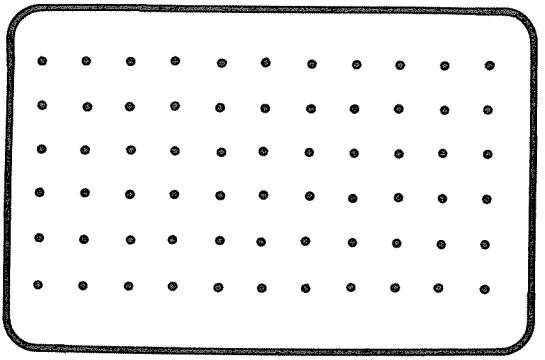


Figure 4

Let's display the same example as in Figure 2, a circle.

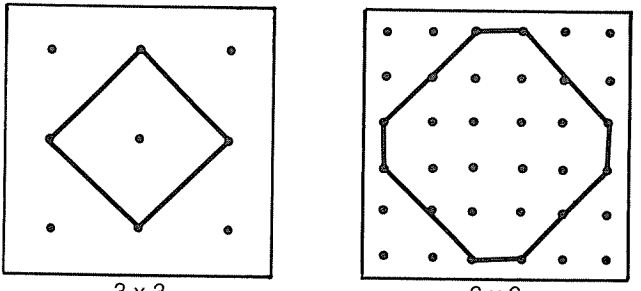


Figure 5

Once again, the 6 x 6 grid produces a more realistic circle than the 3 x 3 grid.

Now for the triangles similar to those of Figure 3.

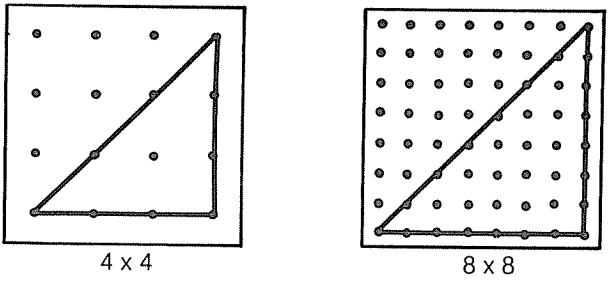


Figure 6

The 4 x 4 and 8 x 8 grids produce equally recognizable triangles. The improvement over Method I is obvious.

The vector plotting method (II) is clearly better than the block plotting (I) method. However, it is more expensive. Some "purists" will not admit that Method I produces valid graphics. But it all lies in the definition of the term.

A manufacturer is faced with two choices; either (a) keep the graphics capability to a reasonable minimum with a price that everybody can afford; or (b) increase the graphics capability and price the product at a higher level (thus eliminating some buyers).

In the TRS-80, Radio Shack has chosen the former alternative, and that will result in increased sales as more people are able to afford the product. Thus, we have a consumer product which is comparable in price to other household appliances. It will appear in homes and schools much faster than expected, and the demand for usable programs and self-teaching materials will be tremendous. We hope this book will be an aid, in a small way, in satisfying that demand.

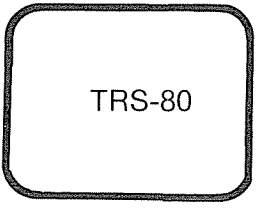
TRS-80 graphics are accomplished using a grid of rectangles arranged in 128 columns and 48 rows as shown on the Video Display Worksheet (page 133). Each individual rectangle may be turned on or off under the programmer's control. Chapter two explains the procedure.

Numeric and alphabetic characters may also be printed at specified locations on the screen. The print positions, numbered 0 through 1023, are also described in Chapter two. They are also shown on the Video Display Worksheet.

With the conclusion of this brief introduction, let's move right into the video display features.

2

THE VIDEO DISPLAY



One of the first desires of all first-time computer users is to make something happen on the video screen. The goal of the rest of this book will be to make that desire reality.

Many experienced programmers and computer scientists (especially those who have million dollar systems!) will call these graphics crude, awkward, and elementary, but many of us are so "computer starved" that any graphics are better than none. Also, one of the challenges to any programmer is to write usable programs within the constraints of his system. The TRS-80 graphics are great as a learning device *because* of their elementary and straightforward nature.

Many may also complain of the slowness of the plotting technique, but this too is an educational advantage. You can actually see the pictures being "painted" on the screen just as you instructed the computer to do it.

There are two methods to light up the screen, and each uses a similar, yet unique, numbering scheme. I will call one the PLOT method and the other the PRINT method.

THE PLOT METHOD

This technique makes use of a numbered grid of 6144 "points" arranged in 48 rows of 128 columns. The grid is numbered from the upper left corner of the screen. Each point is designated by both Column and Row similar to the (X,Y) Cartesian Coordinates used in beginning algebra

courses. The order is important: (column, row). The "points" are not really points. They are tiny rectangles (taller than they are wide). See Figure 7 for the arrangement of the points. The plot position (127,47) is shaded in the figure.

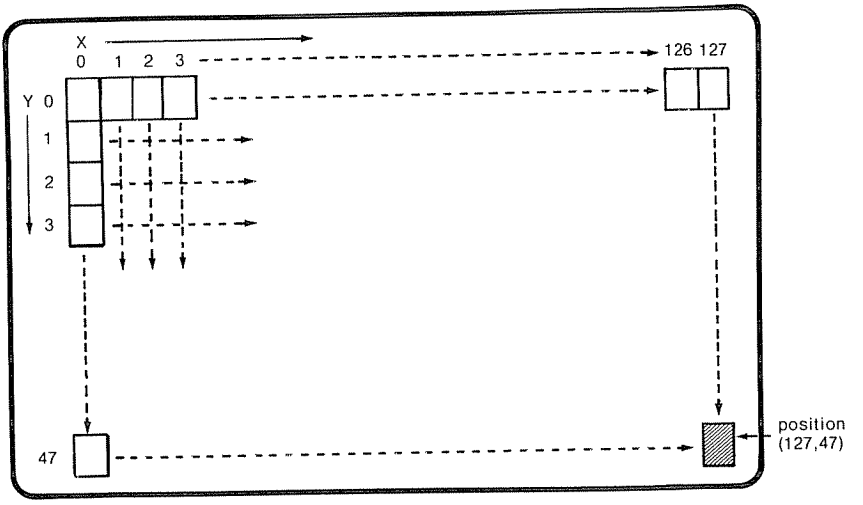


Figure 7
Plot Positions On The Video Screen

The TRS-80 BASIC statement to plot a point is:

```
SET (X,Y)
```

As used in a typical program:

```
10 X = 60          Column
20 Y = 30          Row
30 SET (X,Y)       "Turns on" point (60,30)
```

The statement SET(X,Y) would turn on the rectangle indicated in Figure 8.

There is also an instruction to "erase" a point on the screen. It is:

```
RESET (X,Y)
```

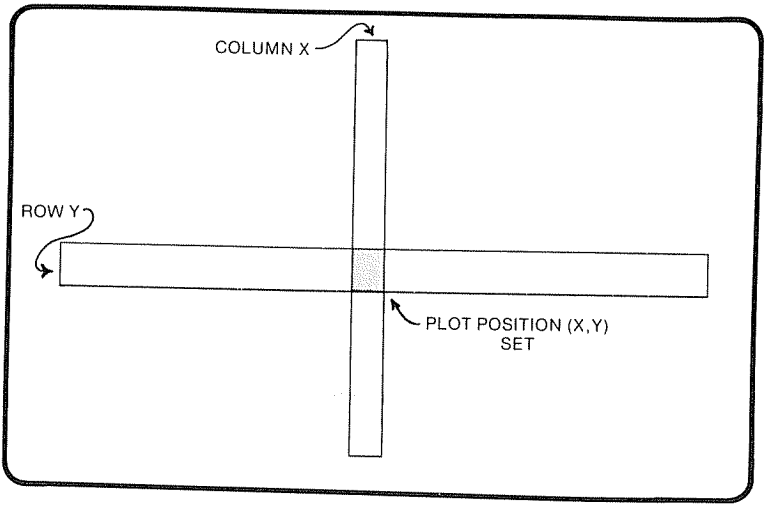


Figure 8

As an example: If we add a new line to the above program segment, we can turn off the point (60,30) by the statement:

```
40 RESET (X,Y)
```

The statement RESET (X,Y) would turn off the rectangle indicted by Figure 9.

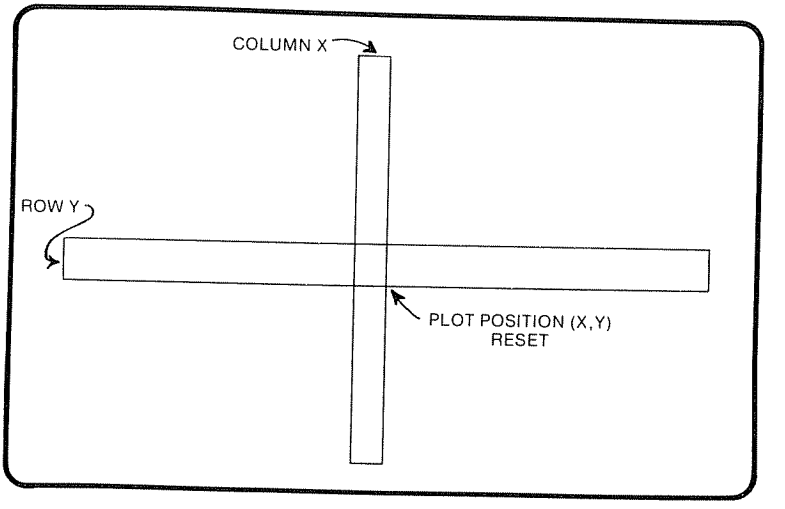


Figure 9

Now that we know how to do it, let's move our point near the center of the screen. We will use a time delay between the SET and RESET statements so that we can see the point before it is turned off.

PROGRAM I

```
10 INPUT C                    Time delay parameter
20 CLS                        Clear the screen
30 A = 64                     Column 64
40 B = 24                     Row 24
50 SET(A,B)                   Turn on point (64,24)
60  FOR N = 1 TO C            Time delay
70  NEXT N
80 RESET(A,B)                 Turn off point (64,24)
90 GOTO 90                    Loop here while you stare
                              at the screen
```

The FOR-NEXT loop at lines 60-70 is a simple time delay which controls the length of time that the point remains lit. The delay parameter (C) is input at line 10. Without the time delay, the point turns off so fast that it cannot be seen. By experimenting with delay values (C = 1 to 10000), you will discover that the point can be seen when C is as low as 1. We will use time delays in many programs throughout the book.

Here are some approximations that I found for the delays.

Value of C	*Light on [seconds]*
10,000	20
5,000	10
1,000	2
500	1

THE PRINT METHOD

The print statement is implemented using a grid of 1024 positions which are also numbered left to right from the upper left corner of the screen. There are sixty-four positions in each row. The numbering continues from row to row (0 through 63 in row 1, 64 through 127 in row 2,... etc. with 960 through 1023 in the last row).

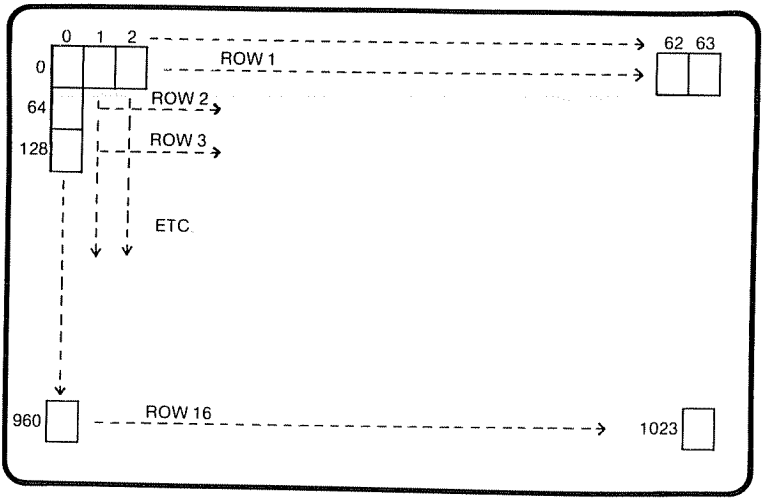


Figure 10 Print positions on the video screen.

A PRINT AT statement for a specific starting position looks like this:

```
50 PRINT AT 66, "THIS BEGINS AT 66"
```

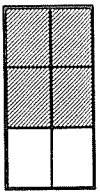
or

```
70 PRINT AT 130, A
```

On the screen you would see:

```
THIS BEGINS AT 66        From line 50
14                       From line 70 (assuming A = 14)
```

There is a relationship between the *print* positions and the *plot* positions. Since there are 6144 plot positions and 1024 print positions, it follows that each print position corresponds to 6144 ÷ 1024, or 6 plot positions. This turns out to be true as illustrated in Figure 11.

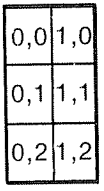


Printed character occupies top 4 plot positions.

Space is part of the print position to allow space between lines.

Figure 11
Print Position And The Six Corresponding Plot Positions

To illustrate the print-plot relationship of Figure 11: PRINT position 0 occupies the same area on the screen as PLOT positions (0,0), (0,1), (0,2), (1,0), (1,1), and (1,2).



Print position 0 (Dark border).

Figure 12
Print, Plot Relationship

To demonstrate the relationship on the TRS-80, let's investigate print position 263. If we run the following program, an S will be printed in the fifth line of the screen. It will be 8 positions in from the left side.

```
 5 CLS                          Clears the program from screen
10 PRINT AT 263, "S"            Prints S at line 5
20 PRINT AT 266, "<--S IS AT PRINT POSITION 263"
30 GOTO 30
```

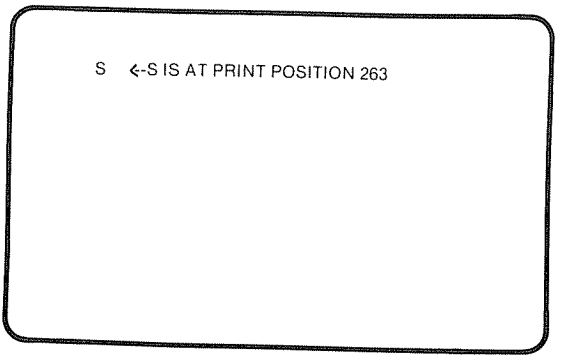


Figure 13
Result of the program on screen.

Looking at the Video Display Worksheet in the TRS-80 User's Manual, we see that print position 263 occupies the area of plot positions (14,12), (15,12), (14,13), (15,13), (14,14), and (15,14).

Now let's add to our program so that all the points adjacent to our S will be lighted. We would want to light points (13,11), (14,11), (15,11), (16,11) along the top; (16,12), (16,13), (16,14), (16,15) along the right side; (15,15), (14,15), (13,15) along the bottom; and (13,14), (13,13), (13,12) along the left side. Like this:

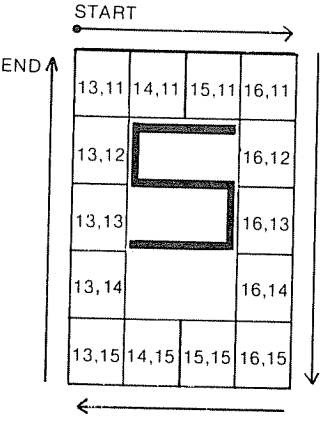


Figure 14

Exercise 1
Write a set of statements which will give the Print Position if the X,Y coordinates of a Plot Position are known. The Print Position must "overlap" the known Plot Position.

PROGRAM II

```
 5 CLS
10 PRINT AT 263, "S     <--S IS AT PRINT POSITION 263"
20 Y = 11
30  FOR X = 13 TO 16                    Top
40   SET(X,Y)
50  NEXT X
60 X = 16
70  FOR Y = 12 TO 15                    Right side
80   SET(X,Y)
90  NEXT Y
100 Y = Y — 1                           This sets Y back to 15
110  FOR X = 15 TO 13 STEP — 1
120   SET(X,Y)                          Bottom
130  NEXT X
140 X = X — 1                           This sets X back to 13
150  FOR Y = 14 to 12 STEP — 1
160   SET(X,Y)                          Left side
170  NEXT Y
180 GOTO 180
```

When we run the program, the screen will look something like this.

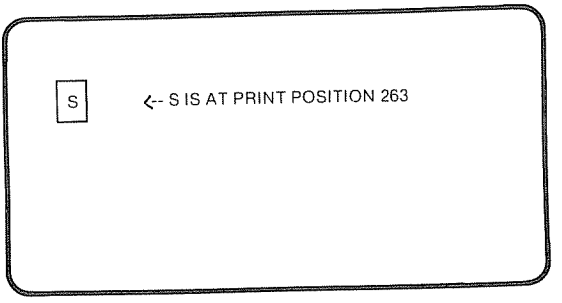


Figure 15

Let's add two more lines to Program II. Inserting the two lines below between lines 10 and 20 will print two more characters. We will print a P directly below S (at position 263+64, or 327) and a T directly below P (at position 327+64 or 391). Here are the two lines:

```
12 PRINT AT 327, "P     <--P IS AT PRINT POSITION 327"
14 PRINT AT 391,"T      <--T IS AT PRINT POSITION 391"
```

Now when we run Program II (modified) we see:

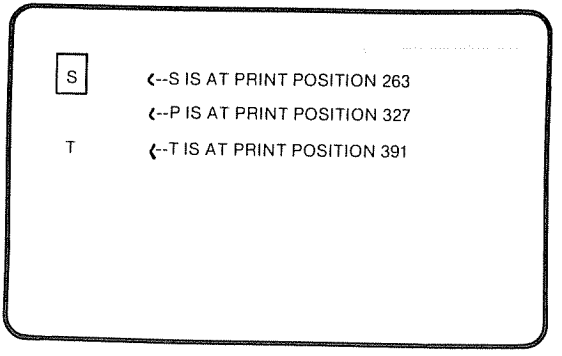


Figure 16

What happened to the P? If you really want to find out, insert a time delay at lines 16 and 18. Then you can see the P wiped out by the rectangle which encloses S.

```
16 FOR N=1 TO 500
18 NEXT N
```

Now when you run the program, you will see the S,P, and T during the time delay. Then the rectangle is drawn around S and the P is wiped out as shown in Figure 17.

Any enroachment by a plotted point on any part of a printed character's 6-block area wipes out the entire character. The same is true in reverse (only more so). A PRINT AT statement wipes out any previously plotted point on its line if the point lies within or to the right of the position specified in the PRINT AT statement. (The last part of this statement is really a simplification which will be modified in Chapter 5.)

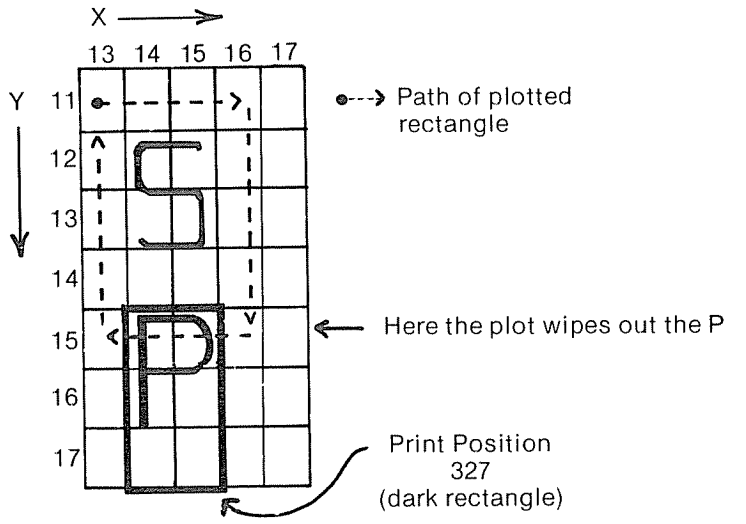


Figure 17

Exercise 2

Change one statement in the previous program to:

(a) Wipe out the S

(b) Wipe out the T

Answers to Chapter 2 Exercises

Exercise 1

One method: P= INT(Y/3*64 + INT (X/2)

where (X,Y) are the plot position. P is the print position, and INT is the *Integer Function.*

The Integer Function Truncates a number

Example: (X,Y) = (15,13)
P=INT(13/3)*64 + INT(15/2)
=INT(4.33..)*64 + INT(7.5)
= 256 + 7
=263 (see Figure 15)

Exercise 2

The statement to be changed is at line 100.

To wipe out S:
$$\left.\begin{array}{l} Y = Y - 2 \\ Y = Y - 3 \\ Y = Y - 4 \end{array}\right\} \text{ or } \left\{\begin{array}{l} Y=14 \\ Y=15 \\ Y=16 \end{array}\right.$$

To wipe out T:
$$\left.\begin{array}{l} Y = Y + 2 \\ Y = Y + 3 \\ Y = Y + 4 \end{array}\right\} \text{ or } \left\{\begin{array}{l} Y=18 \\ Y=19 \\ Y=20 \end{array}\right.$$

Additional Suggestions to Test Your Programming Prowess.

1. Print characters at other Print Positions.
   (a) With FOR-NEXT loops, draw horizontal lines from the left side of the screen that will wipe out each character.
   (b) Do the same with vertical lines.

2. Print a character at some Print Position.
   (a) SET one point to wipe the character out.
   (b) Reprint the character and try a different point.
   (c) Repeat until you find all 6 plot points which will wipe out the character.

3. Print 3 characters at different Print Positions, and see how many guesses it takes you to wipe out all three characters. (Use SET as in 2)

4. Use a random number generator to place 3 characters on the screen. See how many guesses it takes to find plot positions to wipe them out. i.e.:

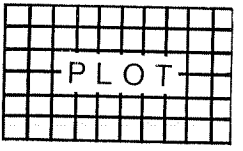
```
FOR N = 1 TO 3
 PRINT AT P, "M"
NEXT N
INPUT X,Y
GOTO XX (where XX is the line number
          of the input statement)
```

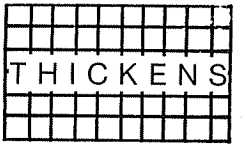
5. Use SET(X,Y) and RESET(X—1,Y) to make a dot move from left to right across the screen.

6. Use SET(X,Y) and RESET(X,Y—1) to make a dot move from top to bottom down the screen.

3

THE PLOT IS EXPLAINED





The use of graphics in a program running on a computer is an important element of good instruction. Many students learn better through pictures than words. Any aid to the non-reader, the slow-reader, or any reader should not be ignored by educators. It is imperative that we learn and understand the graphics capabilities of inexpensive computers such as the Tandy-Radio Shack TRS-80.

In the last chapter, we discussed two methods (Plot and Print) to light various parts (blocks, rectangles, or whatever you want to call them) of the video display. The Plot Method used rectangles designated by their row and column numbers starting from the upper left corner of the screen. Typewriter characters are lighted in the Print Positions which are designated by one number (0-1023). Tiny rectangles are lighted in the Plot Positions which are designated by a pair of numbers (column, row) with columns 0-127 and rows 0-47.

A sketch of the Plot Positions is shown in Figure 18.

Any specific rectangle can be "turned on" by the Level I Basic statement:

SET(X,Y) Where X is the column number and Y the row number.

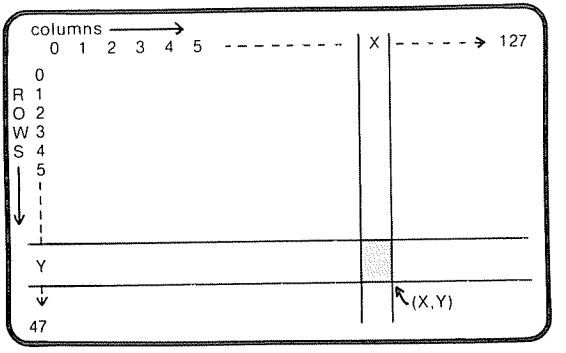


Figure 18

Notice in Figure 18 that X and Y are non-negative integers (no negative values and no fractional parts). People who are inquisitive will want to know what happens if they don't follow the rules. Here are some examples:

Result on screen when run

Program

```
 5 CLS
10 X = — 1
20 Y=2
30 SET(X,Y)
```

```
HOW?
 30 SET(X?,Y)

READY
>
```

This is an error prompt

This ? shows where error is

This says I'm ready for your correction

Figure 19

This illustrates one of the few TRS-80 Level I error messages. Small is beautiful! There is no need for a long list of error codes which must be looked up each time an error is made. The simple message HOW? says, "Your programming may be OK, but I don't know HOW to SET(X,Y) for the value of X which you gave me."

Result on screen when run

Program

```
 5 CLS
10 X=1
20 Y=-2
30 SET(X,Y)
```

```
HOW?
  30 SET(X,Y?)

READY
>
```

Same error message

This time it's the Y value. Doesn't like negative values.

Figure 20

What happens if we try X > 127 or Y > 47?

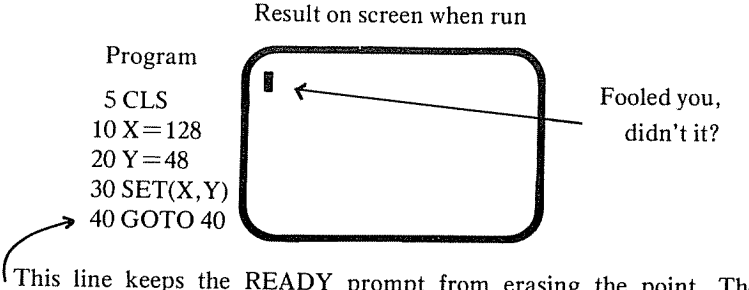
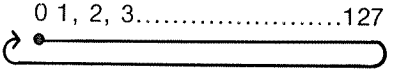
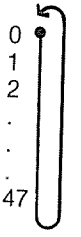


Figure 21

Even though the last column on the display is 127, it "wraps around" to column 0 again since 128=127+1. (It considers 0 as the number following 127 and uses it as its next plot position.)



The same is true for the row. (48=47+1). Therefore, it "wraps around back to position 0.

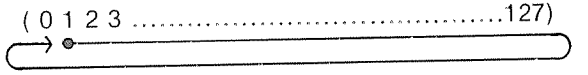


Therefore the rectangle lighted was in plot position (0,0).

Correspondingly, if we say:

SET(129,48)

we will light up the plot position (1,0). Since 129 is two more than 127, it goes to the second position, which equals 1.



The SET statement will take any positive integer ≤ 32767. If we try 32768 for X such as:

```
SET(32768,40)
```

we will get our friend:

```
HOW?
 XX SET(32768?,40)
```

How about non integers such as 1.5, 2.5 or 3.5...etc.?

Our TRS-80 won't take a negative integer for a row or column number, and it does "funny things" when we give it a positive integer larger than the last row or column. Now, will it take a positive non-integer such as 1.5? If so, where will it place the light for the statement: SET(1.5,3)?

Exercise 3

If I input this program, will it light (1,3), (2,3), or ½ of each?

```
 5 CLS
10 SET(1.5,3)
20 GOTO 20
```

Answer (before reading further)

How can we prove it? Here are three short programs, and the results give us the clues we need.

A

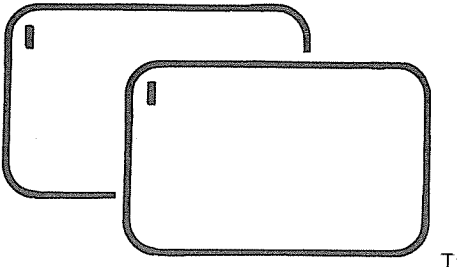
```
 5 CLS
10 X=1.5
20 Y=3
30 SET(X,Y)
40 GOTO 40
```

B

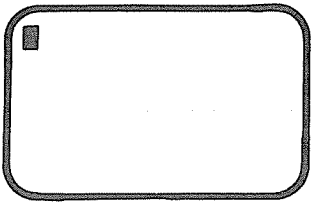
```
 5 CLS
10 X=1.5
20 Y=3
30 SET(X,Y)
40 X=1
50 SET(X,Y)
60 GOTO 60
```

C

```
 5 CLS
10 X=1.5
20 Y=3
30 SET(X,Y)
40 X=2
50 SET(X,Y)
60 GOTO 60
```



Same rectangle lighted (1.5,3) and (1,3).



Two adjacent points lighted when (1.5,3) and (2,3) were attempted.
One must be at (1,3) and the the other at (2,3).

Figure 22

Therefore, we see that SET(1.5,3) causes the plot position (1,3) to light.

A positive non-integer is treated as the greatest integer contained in the non-integer. Mathematicians use the symbol [X] for the Greatest Integer Function. It means the same thing:

[1.4] = 1	It's like always rounding a non-integer
[2.5] = 2	down instead of up (mathematicians
[3.6] = 3	have another fancy name for this —
etc.	truncating a number).

RECTANGLE GRAPHICS

Let's use the SET statement to "draw a series of straight lines. First we will draw a horizontal line somewhere near the middle of the screen. (63,23) is used as a reference

point. This will be the "center" of our rectangle. We will draw a line nine units long line four rows above the reference point. This means setting 9 points: (59,19), (60,19), (61,19), (62,19), (63,19), (64,19), (65,19), (66,19), and (67,19). To do them individually takes too many statements, so we will use a FOR-NEXT loop. The first attempt for the first part of the program might look like this:

```
5 CLS
10 Y = 19
20 FOR X = 59 TO 67
30   SET(X,Y)
40 NEXT X
50 GOTO 50
```

Y will always be 19 for this line.

The FOR-NEXT loop will set each of the points above: first (59,19), then (60,19), ...etc. When X gets to *68,* the statement at Line 50 will be executed. If you think X stops at 67, press the break key and then type PRINT X and press ENTER. The computer will print 68. Each time a pass is made through the loop, X is incremented by one. *After* X reaches 68, the computer sees it's finished and exits the FOR-NEXT loop. Keep this in mind in future parts of the program.

VIDEO DISPLAY WHEN RUN

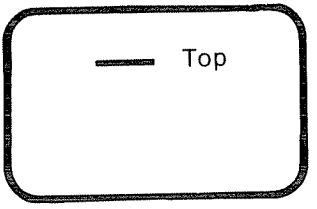


Figure 23

We now have the top of our rectangle. Next we'll add the right side. We want this vertical line to also be 9 units long, but we already have on of them (67,19) included in our top line. Remember, after exiting time the first loop, X is at 68. Therefore, we must back it up to 67. We can do this with the statement X=67 or the statement X=X—1. We will use the latter in order to make the program more general (more on this later).

Exercise 4

Fill in the steps for the FOR-NEXT loop:

```
50 X = X — 1
60 FOR _____
70 _____
80 _____
90 _____
```

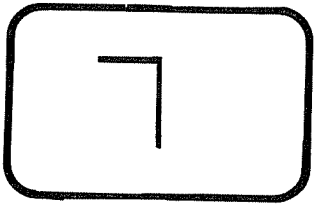
Since Level I BASIC allows multiple statement lines, we can revise our previous program to put all of the first part on line 10.

```
5 CLS
10 Y = 19: FOR X = 59 TO 67: SET(X,Y): NEXT X
20 X = X — 1
30 FOR Y = 20 TO 27
40   SET(X,Y)
50 NEXT Y

60 GOTO 60
```

This draws the TOP

This adds one vertical side

VIDEO DISPLAY WHEN RUN



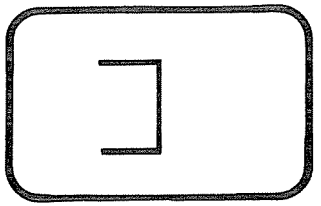
It looks taller than wide because the lighted "points" are really rectangles. □

Figure 24

The FOR-NEXT loop (lines 30-50) is exited with Y=28. Once again, we must back up one to draw the bottom line. We will draw this line backwards from right to left.

```
 5 CLS
10 Y = 19: FOR X = 59 TO 67: SET(X,Y): NEXT X        Top
20 X = X — 1: FOR Y = 20 TO 27: SET(X,Y): NEXT Y     Right side
30 Y = Y — 1
40  FOR X = 66 to 59 STEP — 1    This makes it go back-
50    SET(X,Y)                   wards (X=66,65,64...59).
60  NEXT X
                                 Remember X ends at 58!
70 GOTO 70
```

VIDEO DISPLAY WHEN RUN



Three sides done

Figure 25

For the last side, we move X back to 57 and finish the final values for Y. Again, we go backwards from the bottom to the top.

```
 5  CLS
10 Y = 19: FOR X = 59 TO 67: SET(X,Y): NEXT X  Top
20 X = X — 1: FOR Y = 20 TO 27: SET(X,Y): NEXT Y Right side
30 Y = Y — 1: FOR X = 66 TO 59 STEP — 1: SET(X,Y): NEXT X Bottom
40 X = X + 1: FOR Y = 26 TO 20 STEP — 1: SET(X,Y): NEXT Y Left side
50 GOTO 50
```

We really didn't have to draw the bottom and left side backward, but it makes it more fun to watch the computer "draw" the rectangle in a continuous manner (shown by dotted lines below).

VIDEO DISPLAY WHEN RUN

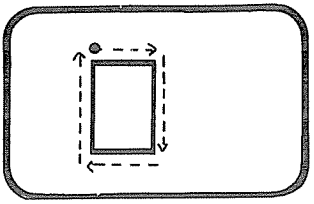


Figure 26

There, now we have drawn the rectangle. Do you remember where the "center" was? Let's put it on the screen too. All we need to add is SET(63,23). It would be more fun to make it blink on and off. So..., one final addition is made.

Add this to the last program.

```
50 SET(63,23): FOR N=1 TO 30: NEXT N
60 RESET (63,23): FOR N=1 TO 30: NEXT N
70 GOTO 50
```

This replaces the GOTO at line 50, and the center keeps blinking away.

Now we know we can draw a rectangle, and we can make its center blink. We have the basis for an entertaining game. Before we go into that, let's return to the idea of making our program more general. Right now we can only put our rectangle in one place unless we rewrite the whole program. We should have written it so that the center could be placed in different places on the screen. Let's go back and run through it once more. We will use the variables M and N for the center of the rectangle. These values will be input at the start of the program. That way you can place the rectangle wherever you like. All SET statements will then be made relative to M and N. Here is what the program looks like.

RECTANGLE PROGRAM

```
5  CLS
10 PRINT "HERE ARE THE DIRECTIONS
     FOR DRAWING A RECTANGLE."
14  FOR W = 1 TO 2000
16  NEXT W
20 PRINT "WHEN YOU SEE THE ? INPUT
     YOUR CENTER."
24  FOR W = 1 TO 2000
26 NEXT W
30 PRINT "GIVE COLUMN FIRST,
     THEN ROW (COMMA IN BETWEEN)."
34  FOR W = 1 TO 2000
36  NEXT W
```

These are directions for turning on the rectangle

W loops are time delays to read directions

```
40 PRINT "COLUMN MUST BE FROM 4 TO
     123, ROW FROM 4 TO 43."
44  FOR W = 1 TO 2000
46  NEXT W
50 PRINT "HERE WE GO - - -"
54  FOR W = 1 TO 2000
56  NEXT W

60 CLS
70 INPUT "WHAT COLUMN AND ROW FOR
     THE CENTER" ; M,N
80 IF (M<4) + (M>123) + (N<4) + (N>43)
     THEN 40

90 Y = N — 4
92  FOR X = M — 4 TO M + 4
94   SET(X,Y)
96  NEXT X

100 X = X — 1
102  FOR Y = N — 4 TO N + 4
104   SET(X,Y)
106  NEXT Y

110 Y = Y — 1
112  FOR X = M + 4 TO M — 4 STEP — 1
114   SET(X,Y)
116  NEXT X

120 X = X + 1
122  FOR Y = N + 4 TO N — 4 STEP — 1
124   SET(X,Y)
126  NEXT Y

130 SET(M,N)
134  FOR W = 1 TO 30
136  NEXT W
140 RESET(M,N)
144  FOR W = 1 TO 30
146  NEXT W

150 GOTO 130
```

Input

Check

Draw the rectangle

We added something new at line 80. Level I BASIC does have logic statements. We used the OR statement. Line 80 says:

(a) If M is less than 4, OR

(b) If M is greater than 123, OR
(c) If N is less than 4, OR
(d) If N is greater than 43, THEN
(e) GOTO line 40.

If any one (or more) of the four conditions are true, the program will ask for a new input. You can't get by that statement unless your inputs satisfy line 40.

Now the program is generalized so that you can place the rectangle wherever you wish (almost).

Answers to Chapter 3 Exercises

Exercise 3

Answer was explained on page 23. It is (1,3).

Exercise 4

```
50 X = X — 1
50  FOR Y = 20 TO 27
70    SET (X,Y)
80  NEXT Y
90 GOTO 90
```

Additional Suggestions for Practice:

1. Use DATA and READ statements to provide data for the FOR-NEXT loops when drawing rectangles.

2. Change the order for drawing the rectangles.
   (TOP, BOTTOM, LEFT, RIGHT)
   (TOP from right to left, LEFT from top to bottom, BOTTOM from left to right, RIGHT from bottom to top) etc.

3. Write a program based on the above that will draw a rectangle in a random location on the screen. Then have the player input a guess as to where the center is located. Make the program blink the location of the guess to give the player a visual clue. The program then goes

back for another guess (provided the first one was incorrect).

Programming Hint: The TRS-80 statement for generating a random number is:

RND(XXX)

It generates a random integer from 1 to the value you insert for (XXX).

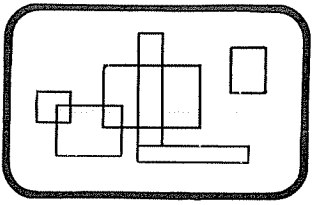
Example: M = RND(43)

will assign a random number from 1 to 43 to the variable M.

4. Modify your program in exercise 3 to keep track of how many guesses it took to find the center, and make a comment such as Excellent, Good, or Not too bad at the appropriate places.

4

GAMES AND ABSTRACT ART



In the last chapter, we posed the problem of creating a Random Rectangle Game. Here is one solution from the author and his 14-year old son, Kurt. We include a description of the game and have separated the program into four distinct parts for ease of understanding.

RANDOM RECTANGLE GAME

Part I Getting Ready

```
5  CLS
10 PRINT "GUESS WHERE THE RECTANGLE'S CENTER IS."
14  FOR W = 1 TO 2000
16  NEXT W
20 PRINT "WHEN YOU SEE THE ? INPUT YOUR GUESS."
24  FOR W = 1 TO 2000
26  NEXT W
30 PRINT "GIVE THE COLUMN FIRST, THEN THE ROW
    (SEPARATED BY A COMMA)"
34  FOR W = 1 TO 2000
36  NEXT W
40 PRINT "LIKE THIS ?62,12 (FOR COLUMN 62, ROW 12)"
44  FOR W = 1 TO 2000
46  NEXT W
50 PRINT "COLUMNS RUN FROM 0 THROUGH 127,
    LEFT TO RIGHT."
```

```
54  FOR W = 1 TO 2000
56  NEXT W
60 PRINT "ROWS RUN FROM 0 THROUGH 47,
     TOP TO BOTTOM."
64  FOR W = 1 TO 2000
66 NEXT W
70 INPUT "PRESS THE ENTER KEY WHEN READY"; A$
```

All of the above is to prepare the player for the game. Never assume that just because you know how to play a game, everyone else will. The time delays at lines 14-16, 24-26, 34-36, 44-46, 54-56, and 64-66 give the player time to read each line separately. The entire message stays on the screen until the player presses the enter key at 70. This allows additional time to make sure the player is ready to start. Although the A$ in line 70 seems to call for an input, the computer will go merrily on its way when the enter key is pressed.

Part II Setting Up The Rectangle

```
100 M = RND(123)
110 N = RND(43)
120 C = 0
130 IF M<4 THEN M = 4
140 IF N<7 THEN N = 7
150 CLS
160 GOSUB 600
```

Lines 100 and 110 select the center (M,N) by choosing M as a random number from 1 to 123 and N as a random number from 1 to 43. Line 120 sets a counter (C) to zero. The counter will keep track of how many guesses the player makes. Since the rectangle will extend 4 points beyond the center, 123 and 43 are used to insure that the rectangle will not go beyond the boundaries of 127 for column and 47 for row. Lines 130 and 140 assure that the rectangle will not extend into the negative values (which the computer doesn't like). Line 150 clears the screen for the game and calls the subroutine (to be at line 600) which will draw the rectangle.

We will use the same program we did in the last chapter to draw the rectangle. (It will come later in the program.)

In Part III we will add the guessing portion of the program. We should be sure to include a trap to make sure the player does not make an illegal entry (column numbers must not be from 0 through 127 and rows from 0 through 47). We must also ask for his input, blink the point he guesses, see if he has guessed the center, count each guess, and either send him back for another guess (if incorrect) or send him on to line 400 (if correct).

> Exercise 5 FOR EXPERTS ONLY
> Write steps for lines 170 through lines 370 to provide everything needed for Part III.

Part III Making The Guess

```
170 GOTO 230
180 PRINT "ILLEGAL ENTRY!! COLUMN 0-127,
      ROW 0-47 PLEASE."
190  FOR W = 1 TO 1000
200  NEXT W
210 CLS
220 GOSUB 600
230 PRINT AT 0, " ": PRINT AT 0, "WHERE IS THE CENTER";
240 INPUT R,S
250 GOSUB 600
260 IF (R<0) + (R >127) + (S<0) + (S>47) THEN 180
270  FOR Z = 1 TO 20
280   SET(R,S)
290    FOR W = 1 TO 30
300    NEXT W
310  RESET(R,S)
320    FOR W = 1 TO 30
330    NEXT W
340  NEXT Z
350 C = C + 1
360 IF (R=M) * (S=N) THEN 400
370 GOTO 230
```

Line 170 forces the program to skip lines 180-220 the first time through the guessing sequence. Line 180 is for those who input an illegal point at line 240. Lines 190 and 200 give time to read the message, and line 220 redraws the rectangle (since the message may have erased part of it). Lines 230 and 240 call for the guess. Line 260 checks to see if your input is legal. If illegal, it sends you back to line 180 which repeats the range of values for your inputs. Lines 270-340 blink the position which was guessed to give a visual clue as to how close the guess is to the rectangle's center. Line 350 counts the guess by increasing C by 1 for each guess made. Line 360 tests with a logical AND to see if your guess is correct. If the guess is correct, you skip over line 370 and go on. If the guess is incorrect, you go back for another guess.

Your program from Exercise 5 may be entirely different from the above. It may be just as good and is probably even better than ours. There are always more ways to reach a solution than there are programmers to do it.

Part IV Final Results

```
400 PRINT "YOU GUESSED IT IN"; C; "GUESSES!"
410 IF (C = 1) + (C = 2) THEN PRINT "EXCELLENT***":
      GOTO 450
420 IF C<5 THEN PRINT "GOOD**": GOTO 450
430 IF C<9 THEN PRINT "NOT BAD*": GOTO 450
440 PRINT "YOU NEED PRACTICE!!!"
450  FOR W = 1 TO 1000
460  NEXT W
470 Y = 1
480 N = 0
490 INPUT "DO YOU WANT TO PLAY AGAIN (Y OR N)";Q
500 CLS
510 IF Q = 1 THEN 100
520 INPUT "THANKS FOR PLAYING! ANYONE ELSE (Y OR N)";Q
530 CLS
540 IF Q = 1 THEN 100
550 END
```

When you have guessed correctly, you land at line 400. It tells you how many guesses you took. Lines 410 and 440

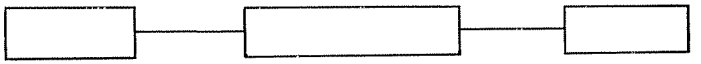
evaluate your performance. Lines 470-490 check to see if you want to play again. If so, they return you to a new rectangle at line 100. If not, the computer thanks you and asks if anyone else wants to play. If they do, they type Y and press the ENTER key. The whole process then repeats.

Part V Rectangle Drawing Subroutine

```
600 Y = N — 4
610  FOR X = M — 4 TO M + 4
620   SET(X,Y)
630  NEXT X
640 X = X — 1
650  FOR Y = N — 4 TO N + 4
660   SET(X,Y)
670  NEXT Y
680 Y = Y — 1
690  FOR X = M + 4 TO M — 4 STEP — 1
700   SET(X,Y)
710  NEXT X
720 X = X + 1
730  FOR Y = N + 4 TO N — 4 STEP — 1
740   SET(X,Y)
750  NEXT Y
760 RETURN
```

> Exercise 6
>
> Design a short program which will place three rectangles on the screen at random locations. You don't have to make the centers blink. You don't have to provide comments or directions.

The rectangle guessing game is fun. It is a great help in learning the approximate positions of plot points on the screen. It also offers many opportunities to test your programming skills with program modifications. A few suggested modifications are listed at the end of the chapter.



ABSTRACT ART

Now that you know how to make the computer draw a rectangle, how about an attempt at abstract art (using rectangles only)? Necessary talent is held to a minimum since we'll make the computer do all the planning and also control the paintbrush.

The heart of the program will be to draw rectangles in random locations on the screen, as we did in the rectangle game. However, instead of clearing the screen, we will leave the rectangles there. It would also be more pleasing to have different sized rectangles (size chosen at random too). Since our rectangles are to be drawn at random locations on the screen, our chances for "painting" two "pictures" exactly alike are extremely small.

We will design the program so that the "painter" can input the number of different sizes for the rectangles. He can also choose how many rectangles he wishes to "paint" in each picture. Once again, the program is broken up into several parts and each part is explained.

ABSTRACT ART PROGRAM

Part I Choosing "Colors" And Form

```
 5 CLS
10 PRINT "ABSTRACT ART PROGRAM"
14  FOR W = 1 TO 1500
16  NEXT W
20 INPUT "HOW MANY DIFFERENT SIZED RECTANGLES
     (1 TO 4)"; F
30 IF (F<1) + (F>4) THEN 20
40 INPUT "HOW MANY RECTANGLES IN PICTURE (10 TO 30)";Q
50 IF (Q<10) + (Q>30) THEN 40
60 CLS: Z = 1
```

Line 5 clears the screen, and line 10 prints the title. At line 20 the painter selects how many rectangles (colors) he wants to use. Line 30 is a trap to make sure he chooses only

the numbers 1, 2, 3, or 4 (this is primitive art). At line 40, the painter decides how many rectangles he will paint (you can see that this is a very restrictive art form). Line 50 is another trap. This time we want to make sure he stays within the limits of the art form. Line 60 clears the screen to prepare the "canvas" for the painting. It also sets a counter (Z) to 1. This provides for counting the number of rectangles so that the computer will know when to stop.

Part II Choosing Placement And Size

```
 70 M = RND(119)
 80 N = RND(39)
 90 P = RND(F)
100 IF M<8 THEN M = 8
110 IF N<8 THEN N = 8
```

Lines 70 and 80 choose where the rectangle currently being drawn will be placed. Line 90 selects the parameter (P) which determines the rectangle's size. This parameter is used to set limits for the lengths of the lines used in "painting" the rectangle. Line 100 says don't start less than 8 units away from the left edge of the screen while line 110 restricts the drawing from running off the top edge.

Exercise 7

What are the possibilities for P at line 90?

_____

Part III Paint

```
120 S = N — 2 * P
130  FOR R = M — 2 * P TO M + 2 * P
140   SET(R,S)
150  NEXT R
160 R = R — 1
170  FOR S = N — 2 * P TO N + 2 * P
180   SET(R,S)
190  NEXT S
```

```
200 S = S — 1
210  FOR R = M + 2 * P TO M — 2 * P STEP — 1
220    SET(R,S)
230  NEXT R
240 R = R + 1
250  FOR S = N + 2 * P TO N — 2 * P STEP — 1
260    SET(R,S)
270  NEXT S
280 Z = Z + 1
290 IF Z = Q + 1 THEN 310
300 GOTO 70
310 GOTO 310
```

Lines 120-270 paint the rectangle chosen from lines 70-90. These lines are similar to those of the rectangle drawing program of Chapter 2. We introduced P to allow for different sized rectangles. In line 280, Z counts how many rectangles have been drawn. If we have drawn all that were requested at line 40, we skip to line 310. If not, line 300 sends us back to line 70 where placement and size are chosen for the next rectangle. Line 310 is our old friend, the infinite loop, which allows us to study the finished project. After careful consideration, you may decide you want to draw a new picture.

1. If you want a new painting,
   (a) press the BREAK key.
   (b) The computer says:

       BREAK AT 310

   (c) If you want to use the same rectangle sizes and the same number of rectangles, type:

       RUN 60

   then press the ENTER key.
   (d) Otherwise just type: RUN and press the ENTER key.

The ABSTRACT ART PROGRAM will bind you to the canvas for hours on end. You will naturally want to paint until you find the most perfect abstract painting. With so many combinations possible, this may take awhile. You

will always wonder, "Will the next one look better?" Who knows? I am sure you will find many modifications for this program. Some are suggested at the end of the chapter.

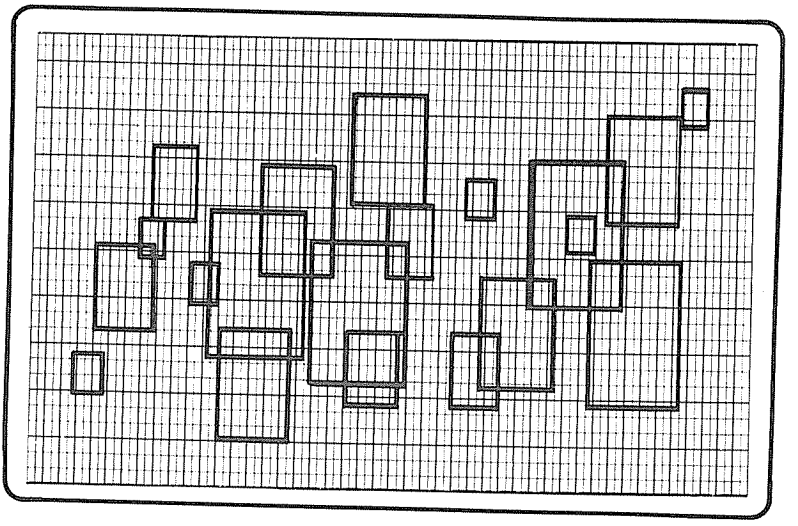


Figure 27
A typical picture.

Answers to Chapter 4 Exercises

Exercise 5

See the answer on page 33.

Exercise 6

```
5 CLS
10 FOR J = 1 to 3
20  M=RND(123): N=RND(43)
30  IF M<4 THEN M=4
40  IF N<4 THEN N=4
50  GOSUB 300
60 NEXT J
70 GOTO 70
```

Exercise 7

Possibilities for P depend upon the input (F) in line 20.
If F is 1, P will be 1.

If F is 2, P may be 1 or 2.
If F is 3, P may be 1, 2, or 3.
If F is 4, P may be 1, 2, 3, or 4.

Additional Suggestions for Practice:

Rectangle Guessing Game

1. Change the restrictions on M and N so that:
   (a) rectangles appear only in the upper portion of the screen - say for Y<24; or
   (b) rectangles appear only in the left portion of the screen. (X<64)
2. Add messages after each guess in the unused portion of the screen in suggestion 1. You can use the statement PRINT AT.
3. Limit the number of guesses (C) so that the computer interrupts the game when this limit is reached.
4. Change the shape of the rectangle.
5. Change the size of the rectangle.
6. Keep all the guesses blinking away.
7. Put more than one rectangular target on the screen.

etc., etc.,

You can probably think of many more. Doctor the program to fit your desires.

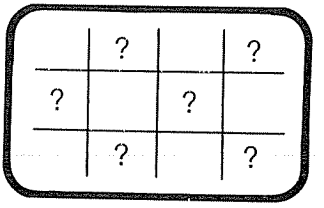
More Suggestions for Practice:

Abstract Art

1. Vary the number of sizes (F in Part I) allowed.
2. Vary the coefficient of P (equal to 2 in Part III). Be careful you don't run the rectangles off the screen.
3. Let Q be a random number (NOT input by the painter).
4. Color in some of the rectangles so that they appear as solid blocks.
5. Paint the whole screen white. Then use the RESET (R,S) statement to turn off points in the drawing. This creates a black on white effect.

5

WHAT'S BEHIND BARS?

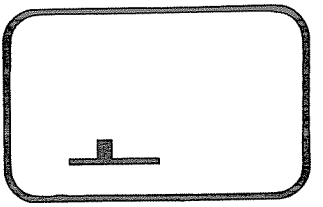


We now know how to draw vertical and horizontal lines and place them anywhere on the screen. For a vertical line, we merely pick a value for X, vary Y, and turn on the points with SET(X,Y) statements. For a horizontal line, we pick a value for Y and vary X.

VERTICAL LINES

We could make a vertical line twice as thick as before by using two adjacent values for X like this:

```
5  CLS
10 Y = 40
15  FOR X = 12 TO 57
20   SET(X,Y)
25  NEXT X
30 X = 14
35  FOR Y = 35 TO 39
40   SET(X,Y)
45  NEXT Y
50 X = 15
55  FOR Y = 35 TO 39
60   SET(X,Y)
65  NEXT Y
70 GOTO 70
```



You can see each line being drawn separately, but then it looks like 1 guy at the bar.

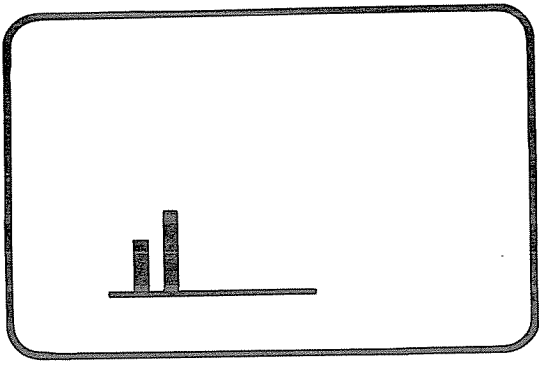
Figure 28

Now let's take out line 70, replace it, and make some additions.

```
70 FOR X=24 TO 25
75  FOR Y= 30 TO 39
80  SET(X,Y)
85  NEXT Y
90 NEXT X
95 GOTO 95
```

This is another way to draw a double line similar to that of lines 30-65 in the previous program.

Lines 70 through 90 form a loop within a loop. Lines 75-85 form the inner loop and are performed 10 times for each pass made through the outer loop (lines 70 and 90). X is fixed at 24 first. Then Y varies from 30 through 39 with points (24,30), (24,31), (24,32), (24,33), (24,34), (24,35), (24,36), (24,37), (24,38), and (24,39) being SET. Then X is fixed at 20, and Y again varies from 30 through 39. This gives us a line 10 units long and 2 units wide.



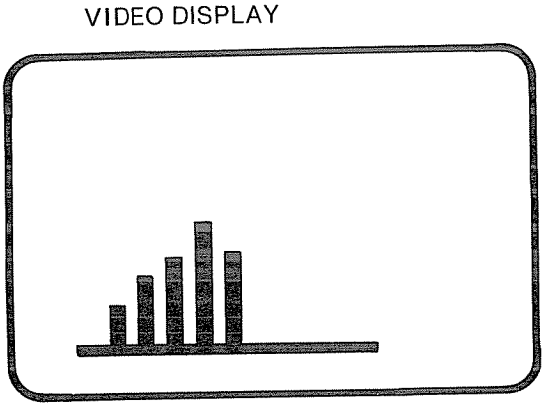
Two guys now standing at the bar.

Figure 29 Video display for first 95 lines

> Exercise 8
> Using the above program, add three more guys at the bar to make a total of five.

If you did Exercise 8, you know that we can put five guys at the bar by adding three in a manner similar to that used for the first two. Here's how one possible program looks.

```
 50 CLS
100 Y = 40
110  FOR X = 12 TO 57                    THE BAR
120   SET(X,Y)
130  NEXT X
140  FOR X = 14 TO 15
150   FOR Y = 35 TO 39
160    SET(X,Y)                          NUMBER ONE
170   NEXT Y
180  NEXT X
190  FOR X = 24 TO 25
200   FOR Y = 30 TO 39
210    SET(X,Y)                          NUMBER TWO
220   NEXT Y
230  NEXT X
240  FOR X = 34 TO 35
250   FOR Y = 25 TO 39
260    SET(X,Y)                          NUMBER THREE
270   NEXT Y
280  NEXT X
290  FOR X = 44 TO 45
300   FOR Y = 20 TO 39
310    SET(X,Y)                          NUMBER FOUR
320   NEXT Y
330  NEXT X
340  FOR X = 54 TO 55
350   FOR Y = 25 TO 39
360    SET(X,Y)                          NUMBER FIVE
370   NEXT Y
380  NEXT X
390 GOTO 390
```



Five guys standing at the bar.

Figure 30

Next we'd like to give each of these guys a number. We'll number them according to when they moved into the picture (1,2,3,4, and 5). This way, the bartender can keep track of their order(s). Let's print the number underneath each figure.

> Exercise 9
> Add statements to the above program to number the guys.

Now, how do we get the numbers in the correct position? If you look back to Figure 10 in Chapter 2, you will see a diagram of the TRS-80 print positions. What we need is the print position directly under each guy at the bar. The secret lies in the relationship of the (X,Y) positions and the print positions. This is shown on page 106 of the TRS-80 User's Manual for Level I BASIC. It is titled TRS-80 Video Display Worksheet.

The number one guy stands in plot positions 14 and 15 for x. The bar itself has a Y value of 40. Therefore, the correct print position is just below (14,40) and (15,40). From the User's Manual, we see this would be print position 896 + 7, or 903. Therefore, we will print a 1 at position 903. The next guy (2) is 10 plot positions to the right of the first. This corresponds to five print positions.

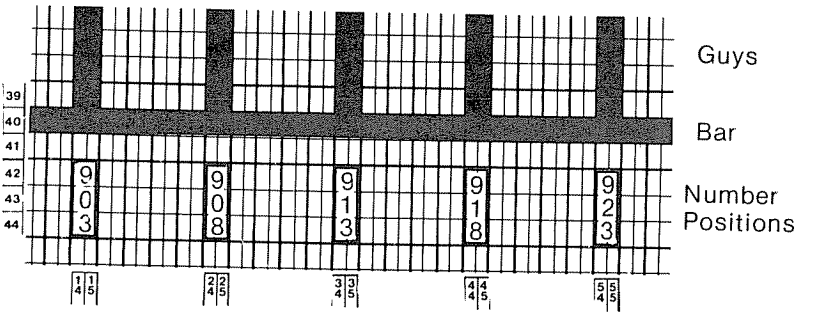


Figure 31

Therefore, print a 1 at 903
a 2 at 908
a 3 at 913
a 4 at 918
and a 5 at 923

> Exercise 10
> From the Video Display Worksheet, derive a formula which will give the print position within which any given plot position lies.

Now we are ready to add to our program by deleting the infinite loop at line 390 and adding:

```
400 PRINT AT 903,"1"
410 PRINT AT 908,"2"
420 PRINT AT 913,"3"
430 PRINT AT 918,"4"
440 PRINT AT 923,"5"
450 GOTO 450
```

By this time, you surely know that you've been tricked into drawing a BAR GRAPH. All we need to do to finish it

up is to add some more lines and labels. The vertical scale will be labeled on the right side since there is more room left over there. Let's finish it up.

```
460 X = 58
470  FOR Y = 18 TO 40
480   SET(X,Y)
490  NEXT Y
500 X = 59
510  FOR Y = 20 TO 40 STEP 5
520   SET(X,Y)
530  NEXT Y
540 PRINT AT 975, "YEARS";
550 PRINT AT 798, "1"
560 PRINT AT 670, "2"
570 PRINT AT 542, "3"
580 PRINT AT 414, "4"
590 PRINT AT 132, "SALES IN MILLIONS FOR THE"
600 PRINT AT 197, "FIRST FIVE YEARS OF THE"
610 PRINT AT 261, "DELTA COMPANY'S EXISTENCE"
620 GOTO 620
```

Exercise 10.5
why the ; in
line 540?

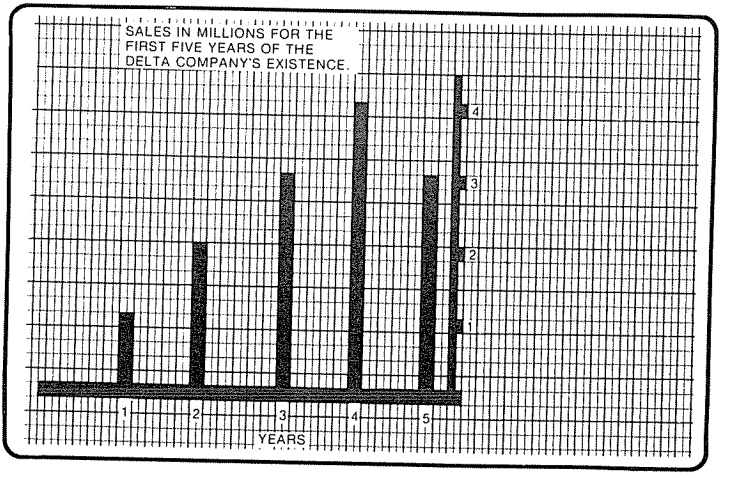
Lines 460-490 draw a vertical line for the vertical scale. Lines 500-530 place markers on that scale at every fifth spot. Line 540 prints the scale's label. The markers are numbered in lines 550 through 580, and the graph's title is placed by lines 590 through 610. Line 620 allows us to admire our masterpiece.

Exercise 11
Derive a formula which will give all six plot positions if the print position number is known.

Our final result looks something like Figure 32.

HORIZONTAL LINES

In the program for the vertical graph, the printed information was placed in an area which would not disturb



of course the grid is not seen

Figure 32

the previous plotted information. In Chapter 2, we warned of dire results if print and plot points encroach upon one another. Let's draw a portion of a horizontal bar graph and show what might happen. We will put the bar on the right side of the screen and name it on the left side like this.

1977 ▬▬▬▬▬▬ Looks simple enough.

Try Number One

```
5  CLS
10 Y = 18
20 FOR X = 14 TO 44
30  SET(X,Y)
40  NEXT X
50 PRINT AT 386, "1977"
60 GOTO 60
```

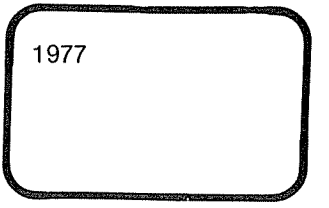


Figure 33

Run the above program. What happened? The bar appeared on the screen momentarily, then all that remained is shown in Figure 33.

Where did the bar go? The printing of 1977 takes up print positions 386, 387, 388, and 389. From the User's Manual the bar starts to the right of this. Now you probably know why the bar vanished.

Try Number Two

```
 5 CLS
10 PRINT AT 386, "1977"
20 Y = 18
30  FOR X = 14 TO 44
40   SET(X,Y)
50  NEXT X
60 GOTO 60
```

Now, run this program. All we've done is move some lines around to do the printing *before* the plotting. Here is what is seen on the screen:

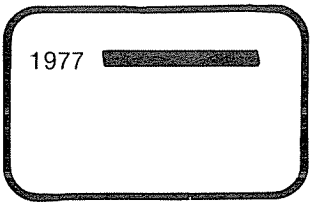


Figure 34

Now it's perfect! Why the difference? A print statement will wipe out everything in its line to the right of the position where it starts (as well as the starting position itself). In the second try, the PRINT AT 386 takes place *before* the drawing of the bar and does not affect it. Notice that the bar starts to the right of the position occupied by the last 7 in PRINT AT 386,"1977".

There is another sneaky way to get around this problem — going back to try number one. Add a semicolon at the end of line 50.

```
50 PRINT AT 386, "1977";
```

This will accomplish the same thing. It tells the computer that there is information to follow on the same line. Therefore, it does not wipe out the rest of the line.

Now let's get on with our horizontal bar graph. Let's use the same data that we did for the vertical graph.

```
5 CLS
10 PRINT AT 386, "1973"
12 PRINT AT 450, "1974"
14 PRINT AT 514, "1975"
16 PRINT AT 578, "1976"
18 PRINT AT 642, "1977"
20 Y=18: FOR X=14 TO 23: SET(X,Y): NEXT X
22 Y=21: FOR X=14 TO 33: SET(X,Y): NEXT X
24 Y=24: FOR X=14 TO 43: SET(X,Y): NEXT X
26 Y=27: FOR X=14 TO 53: SET(X,Y): NEXT X
28 Y=30: FOR X=14 TO 43: SET(X,Y): NEXT X
30 GOTO 30
```

For those who insist on short-cuts, each loop can be logically placed on one line.

Exercise 12

Run the above program. Then devise a scale to show the value of sales per year. Draw the scale under the graph with appropriate markers for each Million $ in sales and label the markers. Place a Title wherever you wish on the screen.

When the horizontal bar graph program is run, it should look something like this.

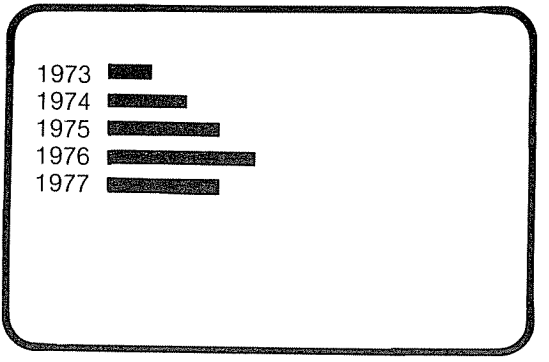
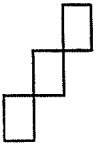


Figure 35

Here is an alternative for lines 20 through 28 which you might like better than ours. It reads the X limits from a data statement.

```
20 FOR Z = 1 TO 5
21  READ Y,A
22   FOR X = 14 TO A
23    SET(X,Y)
24   NEXT X
25  NEXT Z
26 GOTO 260
27 DATA 18,23,21,33,24,43,27,53,30,43
```

So far, all we have drawn have been rectangles. Rectangles were used to draw the rectangles. It is easy to draw vertical and horizontal lines with this method. How about straight lines drawn at some angle other than 0 degrees or 90 degrees? Using rectangles? What would it look like? This?



We will explore the possibilities in Chapter 6.

Answers to Chapter 5 Exercises

Exercise 8

Given in text on page 43.

Exercise 9

Given in text on page 45.

Exercise 10

P = (INT(Y/3)) *64 + INT(X/2)

Exercise 10.5

Keeps the cursor from moving the whole picture up a line.

Exercise 11

$X_1 = (P - INT(P/64)*2;\ X_2 = X_1 + 1$
$Y_1 = INT(P/64)*3;\ Y_2 = Y_1 + 1;\ Y_3 = Y_1 + 2$

Plot positions:

(X_1,Y_1), (X_1,Y_2), (X_1,Y_3)
(X_2,Y_1), (X_2,Y_2), (X_2,Y_3)

Exercise 12

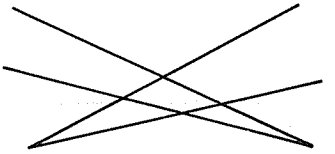
This will vary from individual to individual. Keep trying until you find something that works.

Additional Suggestions for Practice

1. Experiment with bar graphs by varying the distance between bars for the most pleasing effect. Try them adjacent, 1 space apart, 2,3, etc.
2. Put a border of your own design around the graph.
3. Put more than one graph on the screen. Separate them with borders.
4. Find some statistical data or complete graphs in newspapers, magazines, etc. and try to duplicate them on the screen.
5. Compile some data related to your household finances and plot a graph for visual effect.
6. Try any other original ideas which you can come up with.

6

STRAIGHT LINES AT ODD ANGLES



Earlier in the book, we drew horizontal lines by assigning a value to the Y coordinate and varied X with a FOR-NEXT loop using the SET(X,Y) statement for each point. Regardless of what value we assigned to X, Y always stayed the same. A mathematician would say that we drew the line of Y=B (where B was selected before we drew the line). In this chapter, we will investigate the more general case of the straight line given by the formula:

I. $Y = AX + B$ (where A and B are both constants)

For the time being, we will let B=0. Then we only have to consider this form of equation I:

II. $Y = AX$

CASE #1

Let's further simplify equation II by letting A=1. Then we have:

IIa. $Y = X$

The graph of this equation should be easy to display on the video screen by our computer.

PROGRAM I

```
 5 CLS
10  FOR X = 1 TO 48          Plots (0,0), (1,1), (2,2)....(47,47)
```

```
20   Y = X
30   SET(X—1,Y—1)
40  NEXT X
50 GOTO 50
```

This is what it looks like on the display.

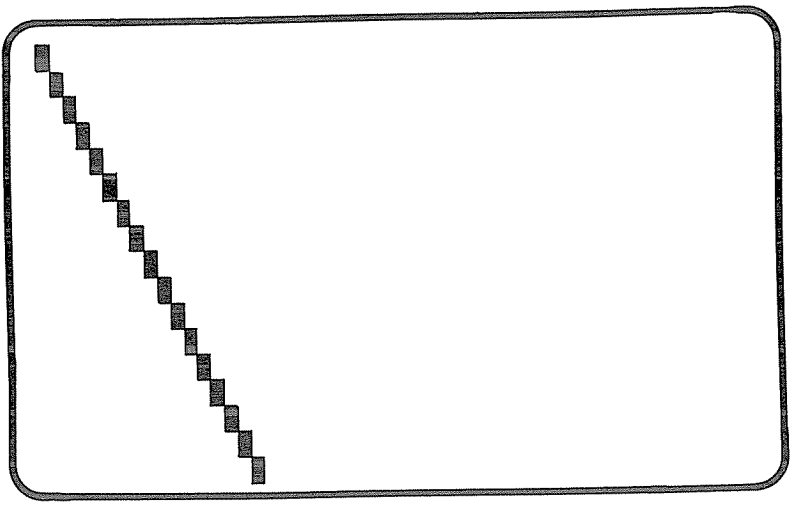


Figure 36

Now, this doesn't look anything like what a first year algebra student would expect. Why not? It's all because of the way the TRS-80's axes are oriented. Ordinarily, the first quadrant seen in mathematics texts (and this is how most of us "think" of it) looks like that of Figure 37a. The orientation of the TRS-80 video display is shown in Figure 37b.

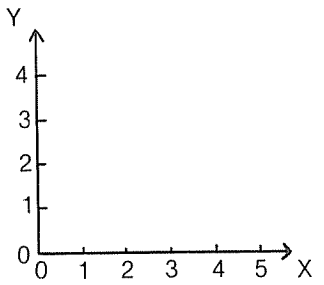


Figure 37a Math text orientation

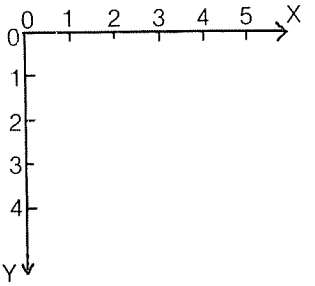


Figure 37b TRS-80 video orientation

A comparison of our straight line Y=X is shown in Figure 38a and 38b.

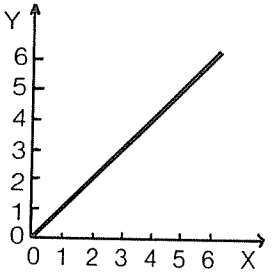


Figure 38a Math Text

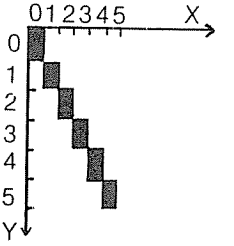


Figure 38b TRS — 80

Let's see if we can interpret the TRS-80 graphics in terms of the usual Cartesian Coordinate System (also known as the Rectangular Coordinate system).

Since the TRS-80 system seems to have the Y axis upside down, pick the TRS-80 coordinate (0,47) to be the rectangular coordinate (0,0). Now let Y = 47 — W (or W = 47 — Y). If we then substitute for Y in equation 11a, we have:

$$47 - W = X \text{ or}$$

$$\text{IIb } W = 47 - X$$

Now if we re-write Program I using equation IIb, we have:

```
5  CLS
10  FOR X = 1 TO 48
20    W = 47 — X          Plots (0,47), (1,46), (2,45),.....(47,0)
30   SET(X—1,W+1)
40  NEXT X
50 GOTO 50
```

If you try running Program II, you will produce a picture on your display which looks something like Figure 39.

This looks more like what we had expected earlier. With a little imagination, the tiny rectangles appear to lie in a straight line.

One difficulty that still remains is the fact that the Y values appear to increase faster than the X values (the slope

of the line seems to be greater than 1). This is due to the fact that the "Points" are really rectangles (taller than they are wide). We will ignore this for the time being. However, the appearance would be more realistic if we placed some scales for the X and Y coordinates along the left side and bottom of the graph.

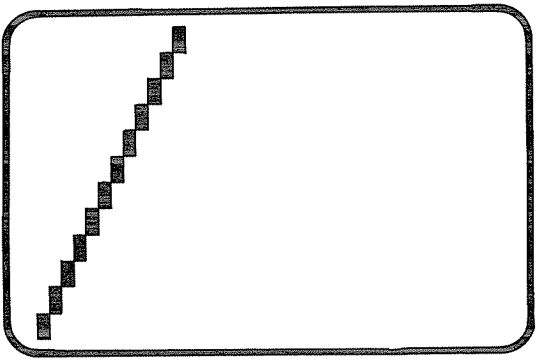


Figure 39

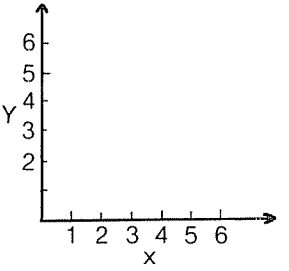
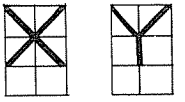


Figure 40

When we print X and Y, six plot positions will be used.



In planning the necessary layout, we must consider that:

1. two columns will be necessary to print the Y,
2. three rows will be necessary to print the X,
3. numbers on the scales will also take two columns and three rows, and
4. the axes will occupy 1 row and 1 column.

What all this means is that the origin (0,0) of the old line must be moved. Let's get the axes drawn and labeled before we worry about where to draw the line.

You should lay it out on a worksheet before trying it on the computer. Here are some suggestions to go with Figure 41.

1. Vertical axis at X=11
2. Horizontal axis at Y=40
3. A marker at every 5 plot positions
4. Number the Y axis at the markers
5. Number the X axis at the markers

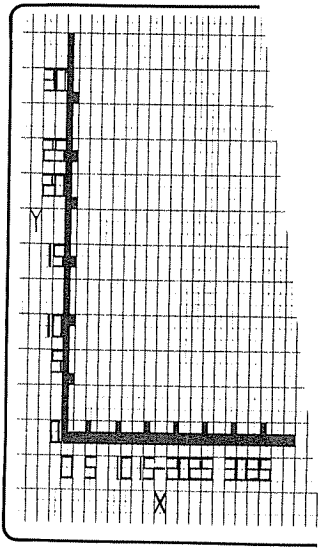


Figure 41

Even before we implement this on the computer, you can see some problems. Since each print position does not correspond to a unique plot position, our numbers and markers do not match very well. Let's work on the Y axis markers and numbers first.

Since there are three plot positions for each print position vertically, some multiple of three will work better than the five used originally. Six (2 x 3) seems to be a good choice in this case. Figure 42 shows how this choice looks.

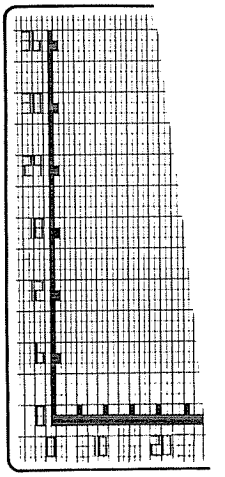


Figure 42

That looks much better, although some may think that intervals of six are not very handy.

Now we must find a better looking X scale. Once again, the numbers and markers don't match very well.

Let's try labeling every 10 X values and take another look.

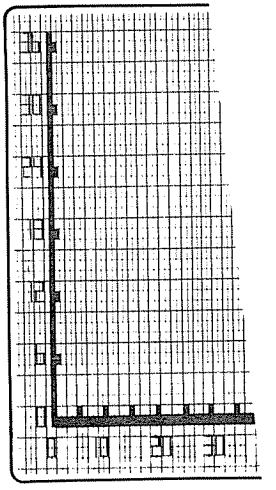


Figure 43

That's even better. Although the X values could be spread out more, let's get on with this preliminary graph.

We are now ready to write a program to produce the labels, scales, numbers, and letters. The program is separated into two parts. The first part will take care of all the print instructions, and the second part will take care of all the plotting.

Part I The Print Positions

```
5  CLS
10 PRINT AT 67, "36"
12 PRINT AT 195, "30"
14 PRINT AT 323, "24"
16 PRINT AT 451, "18"
18 PRINT AT 513, "Y"
20 PRINT AT 579, "12"
22 PRINT AT 708, "6"
24 PRINT AT 836, "0"
26 PRINT AT 902, "0   10   20   30"
28 PRINT AT 973, "X";
```

Note: three spaces between numbers

Part II Plot Positions

```
30 X = 11
32  FOR Y = 3 TO 40                    Y AXIS
34   SET(X,Y)
36  NEXT X

40 X = 12
42  FOR Y = 4 TO 34 STEP 6             Y MARKERS
44   SET(X,Y)
46  NEXT Y

50 Y = 40
52  FOR X = 12 TO 51                   X AXIS
54   SET(X,Y)
56  NEXT X

60 Y = 39
62  FOR X = 16 TO 46 STEP 5            X MARKERS
64   SET(X,Y)
66  NEXT X

70 GOTO 70                             NOW, HAVE A LOOK
```

Now, when you run the program, you get the picture shown in Figure 43. All we have to do now is add Part III which will draw the line of equation IIb (W = 47 — X).

Looking at Figure 43, we can see that the line must start with an X value of 1 and a Y value of 1 according to the graph. However, the actual plot position on the video display is (12,39). We want to vary X until it reaches 37 according to the graph (but this is 48 on the video display). Therefore, we must vary X in equation IIb from 12 to 48.

Using equation IIb, we see that when X = 12, W will equal 47 — 12 or 35. This is not correct. When we moved our axes, we moved the X axis up eight positions and the Y axis over twelve positions. Therefore, we have a discrepancy and must adjust our equation. The equation we must now use is:

IIc W = 51 — X

Using equation IIc, we write Part III of our program which draws in the line for the final graph shown in Figure 44.

Part III Drawing the line

```
70 FOR X = 12 TO 48
72  W = 51 — X
74  SET(X,W)
76 NEXT X

80 GOTO 80
```

Figure 44 is the final display for the graph of the equation of Cas #1 which was Y = AX. In this Case, we fixed A =1.

We will next consider other values for A.

> Exercise 13
>
> Draw a sketch of what the graph might look like if we drew three lines, one for A = 2, one for A = 0.5, along with A = 1 which was shown. You can draw lines instead of rectangles.

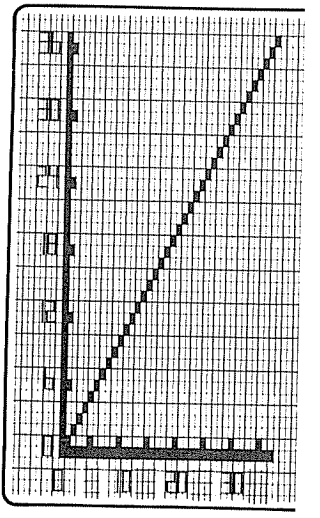


Figure 44

CASE #2

Consider the possibility for other values of A. For the line Y = 2X, the values for Y are double those for X, so the line would not climb at a steeper angle than for Y = X (the slope of the line now equals 2 instead of 1). For the values to be shown on our graph, our FOR-NEXT loop only needs to go FOR X = 12 TO 30. To draw it we add:

Part IV Drawing Y = 2X

```
80  FOR X = 12 TO 30
82    W = 63 — 2 * X
84    SET(X,W)
86  NEXT X
```

CASE #3

Now let's change A to 0.5. This line will only be one-half as steep as it was originally. The equation is now Y = 0.5X, and the slope of the line is 0.5. Our FOR-NEXT loop needs to go FOR X= 12 TO 84. This is fine as long as X is an even number. However, we also need to consider what happens when X is an odd number (such as 13). When the value of Y is calculated, we get some fractional value (Y=

0.5*13 = 6.5). We know from Chapter 3 that the computer truncates the result and only plots integers (for X= 13, it would plot (13,6)). Let's try it anyway and see what happens.

Part V Drawing Y = 0.5X

```
90   FOR X = 12 TO 84
92    W = 45 — INT(0.5 * X)
94    SET(X,W)
96   NEXT X

100 GOTO 100
```

Also modify Part I and Part II of the program to extend the X axis.

We now run the completed program to show the graphs of all three equations.

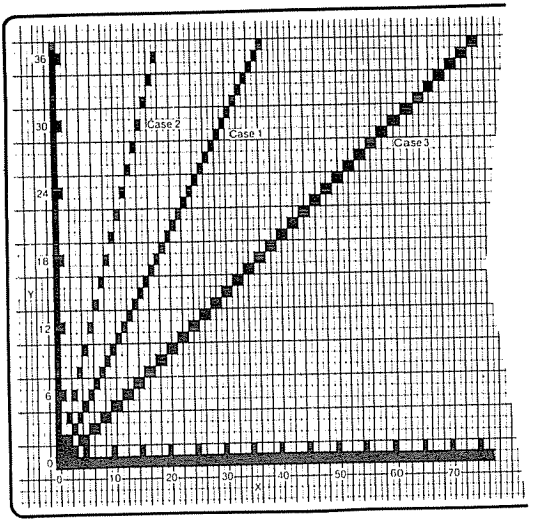


Figure 45

We have now graphed the three equations:

Case 1 Y = X
Case 2 Y = 2X
Case 3 Y = 0.5X

When they are all plotted on the same axis, it is easy to see

how the slope of a line (A in the equation, $Y = AX$) affects the graph of the equation.

> Exercise 14
>
> Consider the graph of $Y=X$ (Figure 44). Sketch the graph of the equation $Y=X+5$ on the same axis as $Y=X$. Where does the graph of $Y=X+5$ cross (intercept) the Y axis?

We have one last consideration for graphing a straight line. Suppose the line does not start at (0,0). We are now back to equation I which was given at the beginning of the chapter.

$$Y = AX + B$$

The constant B determines where the line crosses the Y axis. This point is called the line's Y-intercept since it is the value of Y where the line intercepts the Y axis. That is, if $X=0$, $Y=B$.

In the sketch of Figure 46, lines are drawn for $Y=X$, $Y=X+2$, and $Y=X+4$. The slope of each line is 1 since the value of A in each of the equations ($Y=AX+B$) is 1. However, each line intercepts the Y axis in a different place (depending on the value of B).

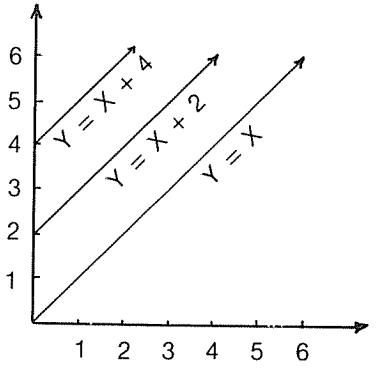


Figure 46

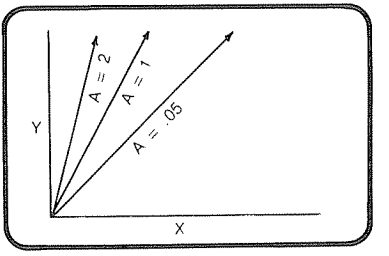
How do we implement this in our computer program? All we have to do is change lines 72, 82, and 92. Since each point on the graph is changed by the value of B, we must change the value of W in those lines by the value of B.

Line 72 changes from W = 51 — X to W = (51 — B) — X
Line 82 changes from W = 63 — 2*X to W = (63 — B) — 2*X
Line 92 changes from W = 45 — INT(0.5 * X) to W = (45 — B) — INT(0.5 * X)

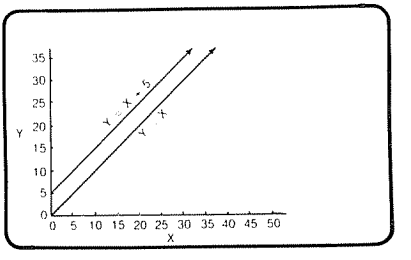
Exercise 15
Modify our line program so that A and B values can be input at the start of the program. Therefore, you can draw any line you wish.

Answers to Chapter 6 Exercises

Exercise 13



Exercise 14



Exercise 15

Part I the same as page 59 except:

```
26 PRINT AT 902, "0 10 20 30 40 50 60 70"
```

Part II the same as page 59 except:

```
52  FOR X = 12 TO 89
62  FOR X = 16 TO 86 STEP 5
```

Part III changes completely to:

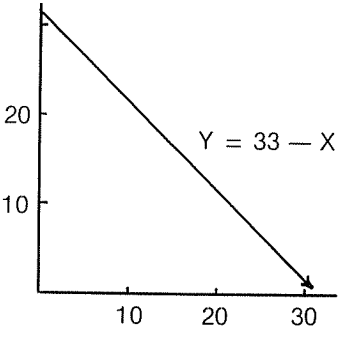
```
70 PRINT AT O, "WHAT SLOPE AND
   INTERCEPT (A,B)";: INPUT A,B
72 PRINT AT O, " "
74  FOR X = 12 TO 70
76   W = 39 + INT(A * (12—X)) — B
78   IF W = 0 THEN 84
80   SET(X,W)
82  NEXT X
84 GOTO 84
```

Additional Suggestions for Practice

1. Use the lines 70-80 in Part III of our program on page 60 to draw a line — EXCEPT turn off the points that are two points behind the present point by inserting the line:

   ```
   75 FOR W = 1 TO 50: NEXT W:
   RESET (X — 2, W — 2)
   ```

2. Use the technique of suggestion 1 in the more general program of Exercise 15. Insert your modification at line 79.
3. Try graphing a straight line with a negative slope such as that shown in this sketch.



4. Solve two simultaneous linear equations by graphing them. (The solution is the value of X where the two lines cross.)

   Try: Y=X and Y= 0.5X + 6
   or Y=X+2 and Y=0.6X+5
   etc.

5. Try many values for A and B in the equation Y=AX+B.

   Suggestions:

A	B
0.1	1
0.2	1
0.3	0
0.4	2
0.5	3
0.6	0

   etc.

7

SITTING DUCKS

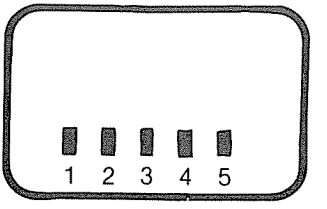
In this chapter, we will explore how a program is developed from start to finish. The computer solves problems, but the problem must be well defined before the computer can be used. The problem should be thoroughly studied, and the plan of attack carefully laid out.

Suppose we wish to set up a shooting gallery with movable targets and a selection of shots from several gun sites. This may sound like a formidable problem, but let's take a look at each item we would have to include.

PHASE I STATIONARY TARGETS

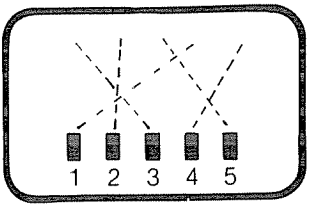
Stationary targets with only the shots moving:

1. Draw the gun sites. They could be simple rectangles placed at the bottom of the screen.



2. Place ducks at random locations along the top of the screen. Ducks will be of some simple shape.

3. Let the player choose which gun is to be fired and have each gun shoot in a different direction.



4. Test to see if a duck has been hit. If so, print "ZAP" at the spot where the duck was.

The POINT statement is used to test whether a given point on the screen is lighted or not. For example, the statement:

```
PRINT POINT (25,14)
```

will print a one (1) if that particular plot point is lighted or will print a zero (0) if that point is not lighted. We can take advantage of this statement in our program to jump out of the regular sequence of steps in our main program and "wipe out" the duck with our "ZAP". We do this by using it in the form:

```
IF POINT(X,Y) = 1 THEN 600
```

Thus, if the shot is traveling along points determined by the values of X and Y and a duck happens to occupy the given point, the POINT statement would have a value of 1. The program would then jump to line 600 where the duck would be ZAPPED. If the duck does not occupy the given point, the shot would continue moving on.

Once we are sure our program is working in the stationary mode of Phase I, we can move on to the second phase.

PHASE II

Make the ducks move

1. Make the head go forward by setting a new point and erasing the old.

2. Make the body move forward by setting a new point forward and resetting the rear point.

(a)  Original position

(b)  Move head

(c)  Move body

Now let's return to Phase I of the program and complete our plans for that part. We first need to tell the player something about the program so that he can use it intelligently. The description does not have to be long or complete, but it should give him some idea as to what is to come.

Lines 100 through 150 give the program description, and lines 50 and 155 merely clear the screen before and after the instructions. Line 150 allows the player as much time as he needs to study the description.

```
                PHASE I — THE SITTING DUCK SHOOT

50  CLS

100 PRINT "TWO SITTING DUCKS WILL APPEAR AT
        THE TOP OF THE SCREEN"
110 PRINT "AND FIVE GUN POSITIONS AT THE BOTTOM."

115  FOR W = 1 TO 1500: NEXT W
```

```
120 PRINT "EACH GUN POSITION FIRES IN A DIFFERENT
      DIRECTION AND"
130 PRINT "YOU MAY CHOOSE TO FIRE FROM ONLY ONE
      POSITION."

135  FOR W = 1 TO 1500: NEXT W

140 PRINT " "
150 INPUT "WHEN YOU ARE READY, PRESS THE
      ENTER KEY.";A$

155 CLS
```

Notice the time delays at lines 115 and 135. They allow time to read each sentence. We are now ready to set up the gun sites. Each site will consist of two rectangles, one above the other. We will also add a number under each for identification.

```
200 SET(20,44): SET(20,43)          #1
210 SET(40,44): SET(40,43)          #2
220 SET(60,44): SET(60,43)          #3   Guns
230 SET(80,44): SET(80,43)          #4
240 SET(100,44): SET(100,43)        #5

250 PRINT AT 970,"1         2         3         4         5"
```

9 spaces between each label

Now, we will place the ducks randomly.

```
300 A = RND(5)          Choose a number 1 through 5
310 X = 55 + A*10       Determine X parameter for
320 M = 5 + A*10        the two ducks

330 SET(X — 2,4): SET(X — 1,4): SET(X,4): SET(X,3)      duck #1
340 SET(M — 2,4): SET(M — 1,4): SET(M,4): SET(M,3)      duck #2
```

So far, you have seen that the program develops naturally if you think through the problem carefully and break it into short, simple parts. We are now ready for the next section where the shots will be made.

(a) The ducks have now appeared on the screen, and the player must choose which gun to fire.

```
400 PRINT AT 0, "WHICH GUN DO YOU WISH TO FIRE (1-5)";:
    INPUT S
410 PRINT AT 0," "
```

Line 410 is added to erase the question after the choice has been made.

(b) Parameters for the starting point of the shot (I) and the direction of the shot (T) are determined for the gun selected (S).

```
420 I = 20*S
430 IF S = 1 THEN T = 1.95
440 IF S = 2 THEN T = 0.7
450 IF S = 3 THEN T = — 1.1
460 IF S = 4 THEN T = 0.7
470 IF S = 5 THEN T = — 1.6
```

I "discovered" the values for T by "playing around" with the program after it was written.

(c) The shot is made.

```
500  FOR Y = 39 to 2 STEP — 1
510   P = I + INT(T*(39 — Y))
520   SET(P,Y — 1)
530   IF POINT (P,Y — 2)=1 THEN 600
540    FOR W = 1 TO 3: NEXT W
550   RESET(P,Y — 1)
560  NEXT Y
```

"Here's where we check to see if a duck is hit. Notice we check one point ahead of the shot.

```
570 GOTO 400
```

Go back and try again.

The next, and last, section is actually only used when a duck has been hit. If the point directly above the position which the shot currently occupies is "on" (i.e. occupied by a duck), we jump to this section. The word "ZAP" is printed at the spot occupied by the duck. The screen is then cleared and play resumes.

```
600 Z = INT(P/2) + 64
610 PRINT AT Z," ZAP "
620  FOR W = 1 TO 300: NEXT W
630 CLS

640 GOTO 200
```

When a duck has been hit and this section executed, the program then returns to line 200 to draw two new ducks.

You should play with phase one of this program for some time before proceeding to phase two. This will give you some familiarity with the angles at which each gun shoots and the time necessary for a shot to arrive at its destination. Sitting ducks are easy to hit, but moving targets are harder.

Phase I has been designed so that only one of the five guns will be able to destroy one of the two ducks. The other four guns will hit nothing. Also, one duck is impossible to hit. In phase II, it is possible for any of the five guns to hit either of two ducks if the correct conditions exist.

PHASE II — MOVING TARGETS

Since both the targets and the shots will be moving, why not move them together? That seems like the simplest way. In the shot segment of phase one, line 540 contained a short time delay. We will now replace that delay by a subroutine which moves the ducks. All this happens within the FOR-NEXT loop which moves the shots. Therefore, each time the shot moves up a line, the duck moves forward one point.

The only change necessary to phase one is to replace line 540 with:

```
540 GOSUB 1000
```

The subroutine is:

```
1000 SET(X+1,3): RESET(X,3)
1010 SET(M+1,3): RESET(M,3)          Move the heads of
                                     both ducks forward

1020 SET(X+1,4): RESET(X - 2,4)
1030 SET(M+1,4): RESET(M - 2,4)      Move the bodies of
                                     both ducks forward

1040 X = X + 1: M = M + 1            Advance the reference points.

1050 RETURN
```

Figure 47 shows 15 frames of the duck animation. The head animation takes place on the line where Y = 3. The body moves along the line where Y = 4.

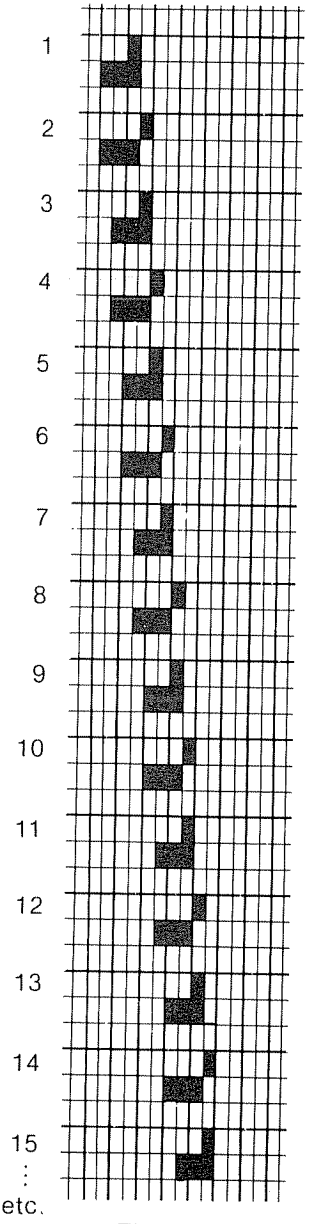
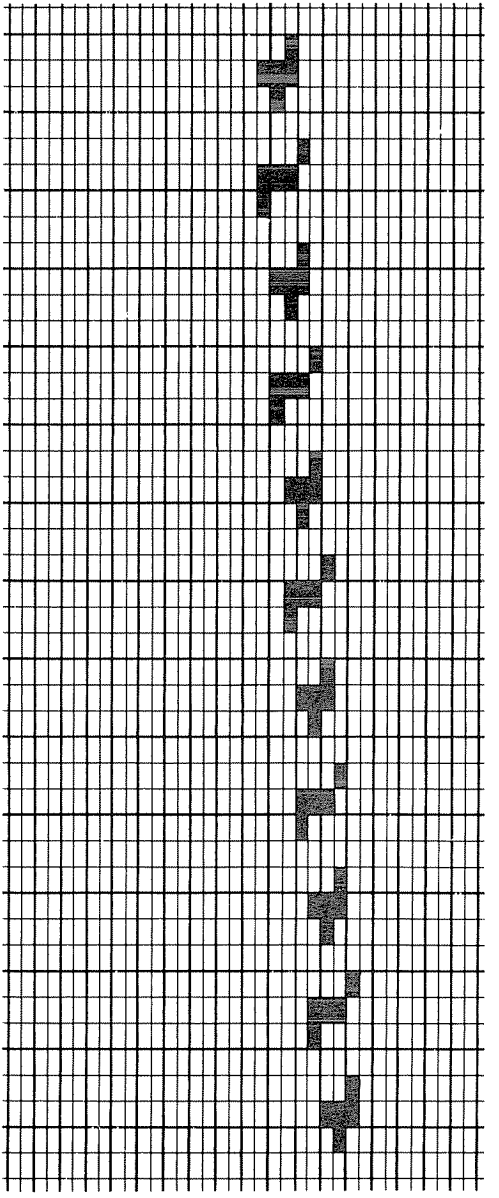


Figure 47

Exercise 16

Write a subroutine at line 1000 to change the duck and its animation to conform to the sketch below.



Answer to Exercise 16

The original drawing on the duck at line 330 would have to change to reflect the addition of the foot:

SET(X—1,5) add

```
1000 SET(X+1,3): RESET (X,3)
1010 SET(X—2,5): RESET (X—1,5)
1020 SET(X+1,4): SET(X,5)
1030 RESET(X—2,4): RESET(X—2,5)
1040 X = X+1
1050 RETURN
```

The second duck's parameters would be similar.

Additional Suggestions for Practice

1. Change the program to allow a player only a given number of shots (say 10) by putting a counter in the shot loop.
2. Keep track of how many hits are made by using a counter in the "ZAP" subroutine.
3. Display the number of shots and the number of hits after each attempt.
4. Replace the "ZAP" with a display of a disintegrating duck.
5. Make one duck move from right to left and the other from left to right.
6. Use only one gun, but make the angle of the shot selectable by the player.
7. Add more ducks.
8. Dream up your own ideas!!

8

BENDING A STRAIGHT LINE

In Chapter Six we found we could approximate straight lines by means of rectangular plotting points. We will now study the more difficult job of bending a straight line into a curve.

We might be tempted to draw the graph of the equation $Y = AX^2$ which looks like the curved lines in Figure 48 when plotted on the usual X,Y coordinate plane.

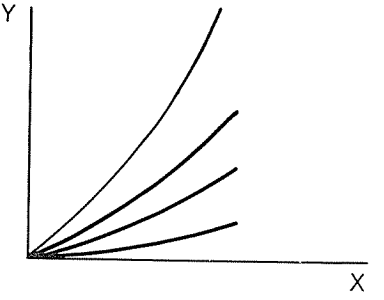


Figure 48

The steepness of the curve depends upon the value of A. Without providing the X and Y scales, let's experiment with a short program to draw the curve for $Y=AX^2$ (a second degree equation).

CURVE DRAWING PROGRAM

```
5  INPUT "WHAT VALUE FOR A IN Y=A*X*X"; A
10 CLS
20  FOR X = 0 TO 62 STEP 2
30    Y = INT(47 — A * X * X)
40    IF Y < 0 THEN 70
50    SET(X,Y)
60  NEXT X
70 PRINT AT 35, "WHAT VALUE FOR A";: INPUT A
80 GOTO 20
```

> Exercise 17
> Run the Curve Drawing Program with several values of A from 0 to 0.02.

The results of running the program will look similar to the display in Figure 49 depending on the values chosen for A.

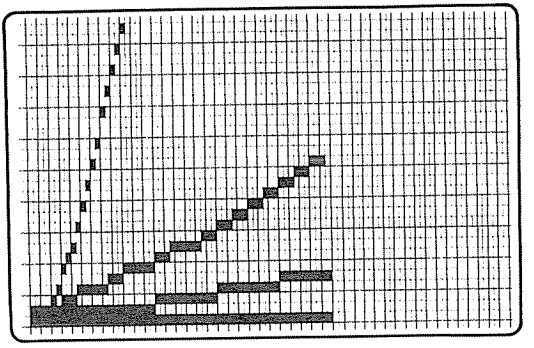


Figure 49

To get a feel for the values calculated for the points, replace line 50 and add these lines to the program. Also change line 80 to : 80 GOTO 10.

```
      15 PRINT "A="; A
      18 PRINT TAB(1); "X"; TAB(5); "Y"
      50 PRINT X;Y
      54  FOR W = 1 TO 1000
      56  NEXT W
> RUN
WHAT VALUE FOR A IN Y+A*X*X  ?.02
A= .02
  X   Y          TAB(1) SETS X IN 1 POSITION
  0  47          TAB(5) SETS Y IN 5 POSITIONS
  2  46
  4  46          If the values pass by too fast,
  6  46          press the ↑ key. This will
  8  45          freeze the computer. Release
 10  45          the key to continue.
 12  45
  .   .
  .   .
  .   .
 44   8          When Y is less than zero,
 46   4          line 40 causes an exit from
 48   0          the loop.
```

Note that the TAB function is used to position the headings for the table of X,Y values in our modified program. TAB is related to the print positions and *could be* thought of as a graphics tool.

Here is a short program to demonstrate the TAB function. The TAB values correspond to the print values for the FIRST ROW of print positions.

TAB DEMONSTRATION PROGRAM I

```
5  CLS
10  FOR X = 1 TO 15
20   Z = (8 — X) * (8 — X)
30   PRINT TAB(Z); "0"
40  NEXT X
50 GOTO 50
```

When the program is run, you will get a display like that of Figure 50a. Figure 50b shows the result of replacing line 30 with: 30 PRINT X;Z.

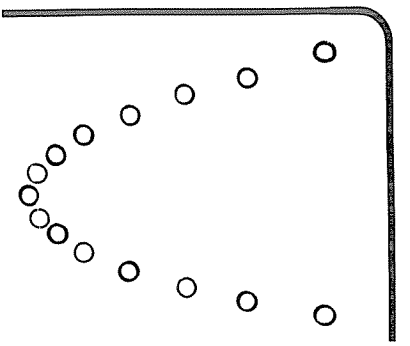


Figure 50a

Tabulation of Results

X	Z
1	49
2	36
3	25
4	16
5	9
6	4
7	1
8	0
9	1
10	4
11	9
12	16
13	25
14	36
15	49

Headings not printed

Figure 50b

The TAB function can also be used to draw pictures on other representations of data. Here is an example.

TAB DEMONSTRATION PROGRAM II

```
100 CLS
110 PRINT TAB(8); "XXXXXXX"
120 PRINT TAB(7); "XX"; TAB(14); "XX"
130 PRINT TAB(5); "(X"; TAB(10); "0"; TAB(12); "0"; TAB(16); "X)"
140 PRINT TAB(5); "(X"; TAB(11); "I"; TAB(16); "X)"
```

```
150 PRINT TAB(7); "X"; TAB(10); ":::"; TAB(15); "X"
160 PRINT TAB(8); "X"; TAB (14); "X"
170 PRINT TAB(9); "X X X"
180 PRINT TAB(10); "X0X"
190 PRINT TAB(9); "X"; TAB(13); "X"
200 PRINT TAB(8); "X"; TAB(14); "X"
210 GOTO 210
```

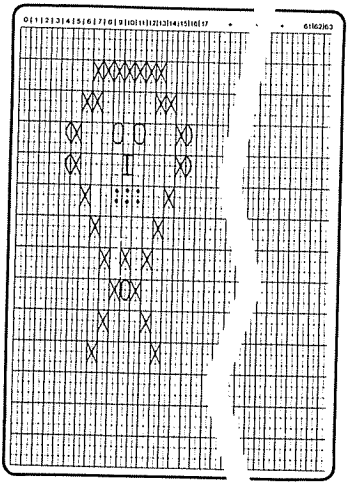
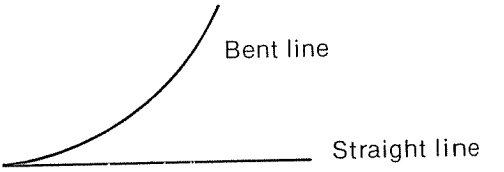


Figure 51 Tab Positions 0 through 63

If you use your imagination, you might see curved lines in Figures 49, 50a, and 51. Remember, however, that there are only 1024 print positions as compared to 6144 plot positions. Therefore, graphics attempted by printed characters are not as well defined as they are for plotted points.

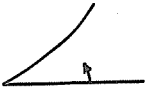
Getting back to the curve of $Y=AX2$, we used some trickery in our Curve Drawing Program for Y=A*X*X. We inverted our Y axis and really drew Y=47 — A*X*X. This made it appear on the video screen like a mathematics student would expect for $Y = AX2$.

Let's forget the mathematics and get back to our original idea of bending a straight line. We showed we could bend one end.

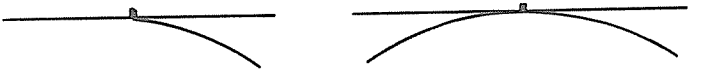


Now we can bend both ends of the straight line at the same time like this?

Since Y = 47—A*X*X started at the bottom of the screen and bent the line upwards, the equation Y=A*X*X should start at the top of the screen and bend the line downwards.



In addition to bending the line downwards, we want to move the starting point over near the center of the screen. Then, we want to also bend the other end.



Let's describe how it is done.

1. Draw the line Y=0 which gives the straight line at the top of the screen. Draw it from X=0 to X=124.
2. For the bent line, we will start at the middle, where X=62. Now go both ways at once. Let X

go from 62 to 0, calculate Y, and set *two* points:
(a) SET(62—X,Y)
(b) SET(62+X,Y)

BENT LINE PROGRAM I

```
 5 PRINT AT 960, "WHAT IS THE VALUE FOR A";: INPUT A
10 CLS
20 Y = 0
30  FOR X = 0 TO 124 STEP 2
40   SET(X,Y)
50  NEXT X
100 FOR X = 62 TO 0 STEP — 2
110  Y = INT(A * X * X)
120  IF Y > 47 THEN 150
130  SET(62-X,Y)
140  SET(62+X,Y)
150 NEXT X
160 GOTO 5
```

> Exercise 18
> Run the Bent Line Program I with several values for A between .001 and 0.02.

How about putting more than one bent line on the display at the same time? To do so, we need to provide for more than one value of A. The easiest way will be to use a FOR-NEXT loop.

BENT LINE PROGRAM II

```
100 CLS
110 PRINT AT 0, "HOW MANY BENT LINES";: INPUT Q
120 Y = 3
130  FOR X = 0 TO 124 STEP 2
140   SET(X,Y)
150  NEXT X
200  FOR A = 1 TO Q
210   FOR X = 62 TO 0 STEP — 2
220    Y = INT(.005 * A * X * X)
225    IF Y > 47 GOTO 250
230    SET(62—X,Y)
240    SET(62+X,Y)
```

```
250   NEXT X
260  NEXT A

300 GOTO 300
```

When this program is run, you get a variety of bent lines. The number depends on your input, Q. The multiple of X^2 will be .005, .01, .015, ... etc. (provided the value of Q is a positive integer).

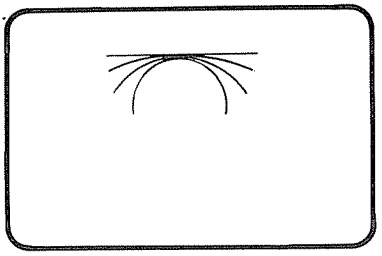


Figure 52

It would be interesting to be able to erase any undesired lines after you have studied the display. This can be accomplished by adding a few lines.

The additional lines are:

```
300 PRINT AT 0, "WHAT LINE DO YOU WANT ERASED
      (1 TO";Q;")";: INPUT A
310 PRINT AT 0, " "
320  FOR X = 62 TO 0 STEP — 2
330   Y = INT(.005 * A * X * X)
340   IF Y >47 THEN 370
350   RESET(62—X,Y)
360   RESET(62+X,Y)
370  NEXT X

400 GOTO 400
```

So, now we can bend a straight line into a simple curve. Can we bend it into a more complex shape? How about Figure 53?

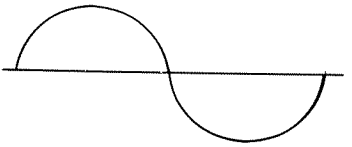


Figure 53

It looks pretty tough, but not if we break it down into four parts. First, let's divide it into two parts as in Figure 54a, and then divide it once again so that we end up with four parts as in Figure 54b.

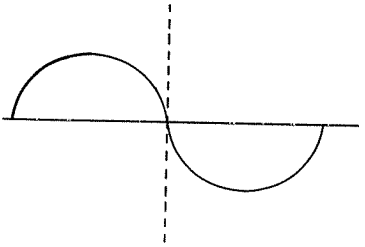


Figure 54a

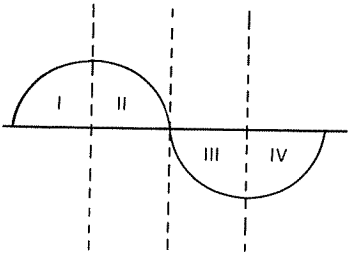


Figure 54b

Each of the four parts is alike except for its orientation with the axis. Part II is the mirror image of Part I. Parts III and IV are the inverted image of Parts I and II together. Therefore, when we plot this curve, we need only consider one-quarter of the graph and SET four points (one in each part). To make it easy, let's set up the graph with the divisions at X=25,50 and 75.

CYCLIC CURVES PROGRAM

```
100 CLS

110 Y = 23
120  FOR X = 0 TO 100
130   SET(X,Y)
140  NEXT X

150  FOR X = 0 TO 25
160   Z = 25 — X
170   Y = INT(23—.036*Z*Z)
```

```
180   SET(X,23—Y)
190   SET(50—X,23—Y)
200   SET(50+X,23+Y)
210   SET(100—X,23+Y)
220  NEXT X
230 GOTO 230
```

Mathematics oriented people will recognize our cyclic curve as a sine wave of sorts.

Quite often, programs can be shortened and simplified by using this reflection of points about an axis. We can look at Part I of our cyclic curve and reflect it about the X axis by making two line changes.

```
200 SET(X,23+Y)
210 SET(50—X,23+Y)
```

We would then get a reflection which would look like Figure 55.

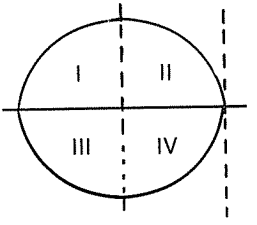


Figure 55

Here is a program which demonstrates reflections. A little imagination is required. Picture a body of water, such as a lake, which reflects an object in the sky. The surface of the water is represented by the straight line. The sky is above the line, and the water is below the line.

REFLECTION PROGRAM

```
100 CLS
110 Y = 23                                   Draws the
120  FOR X = 0 TO 127                        water surface
```

```
130   SET(X,Y)
140  NEXT X
150 SET(20,4): SET(21,4): SET(19,4): SET(21,3)      An H in the
160 SET(21,5): SET(19,3): SET(19,5)                 sky
170 SET(22,44): SET(23,44): SET(21,44): SET(23,43)  Reflected
180 SET(23,45): SET(21,45): SET(21,43)              H
190 SET(42,20): SET(42,19): SET(42,18)
200 SET(43,18): SET(41,18)                          A T in the sky
210 SET(43,26): SET(43,27): SET(43,28)
220 SET(42,28): SET(44,28)                          Reflected T
230 GOTO 230
```

You should run the program and observe a display similar to Figure 56.

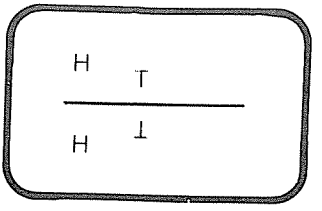
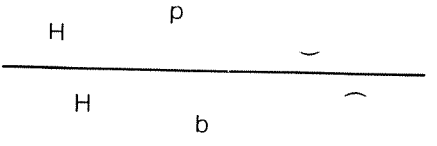


Figure 56

Exercise 19

Draw something similar using your own design to show reflections such as:



Answers to Chapter 8 Exercises

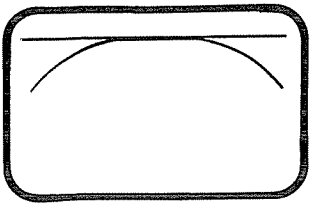
Exercise 17

Results will vary depending upon your inputs. Figure 49 shows a sketch of a typical result.

Exercise 18

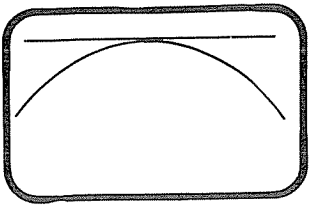
Results will vary. Below are typical runs for A = .003, .01, and .02 with a partial table of values for X, 62—X, 62+X, and Y with the points that are set for those values.

X	62—X	62+X
62	0	124
50	12	112
38	24	100
26	36	88
14	48	76
0	62	62



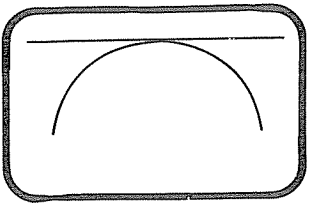
A = .003

Y	POINTS
11	(0,11) (124,11)
7	(12,7) (112,7)
4	(24,4) (100,4)
2	(36,2) (88,2)
0	(48,0) (76,0)
0	(62,0) (62,0)



A = .01

Y	POINTS
38	(0,38) (124,38)
25	(12,25) (112,25)
14	(24,14) (100,14)
6	(36,6) (88,6)
1	(48,1) (76,1)
0	(62,0) (62,0)



A = .02

Y	POINTS
76	(0,76) (124,76)
50	(12,50) (112,50)
28	(24,28) (100,28)
13	(36,13) (88,13)
3	(48,3) (76,3)
0	(62,0) (62,0)

Exercise 19

Displays will vary.

Additional Suggestions for Practice

1. Refer to Figure 48. Make the curves:
   (a) originate in upper left and curve downward
   (b) originate in upper right and curve downward
   (c) originate in lower right and curve upward

   Hint: 1. change the SET instruction or
   2. modify lines 20 and 30.

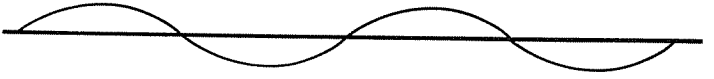
2. Use the TAB function in a FOR-NEXT loop to create designs and pictures:

```
FOR X = ? TO ? STEP ?
PRINT TAB(X);"*"; TAB(?X); "'"
    ...etc.
NEXT X
```

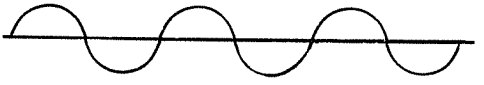
3. Try other curves for the Bent Line Program I such as:

```
Y = A * X * X + B * X + C or
Y = A * X * X * X or
Y = A / ( X * X )
 ...etc.
```

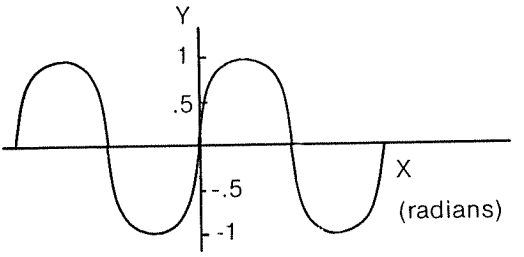
4. Write a variation of the Cyclic Curve program that will draw two complete cycles.



-or three cycles.

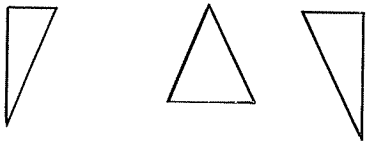


5. Put in a vertical axis to suggestion 4 and label the axes.



9

THE ETERNAL TRIANGLE



In this chapter, we will investigate the difficulties involved in drawing a recognizable triangle. As in the case of straight lines, angles with vertical and horizontal sides are easy to duplicate. However, when an angle other than 90 degrees is encountered, some imagination is required.

Let's begin with angles. There should be no question about drawing a right angle. All we need are two straight lines, one vertical and one horizontal. The only problem arises in making sure that the lines meet at the end of each line.

The program for right angles is described in two parts. Part I draws the horizontal components of four angles. Part II then draws the vertical components.

RIGHT ANGLE PROGRAM

Part I

```
100 CLS
110 A = 0: B = 10: Y = 0

120  FOR Z = 1 TO 2
130   FOR N = 1 TO 2
140    FOR X = A TO B
150     SET(X,Y)
160    NEXT X
170    A = A + 15: B = B + 15
```

```
180   NEXT N
190   A = 30: B = 40: Y = 10
200 NEXT Z
```

Three FOR-NEXT loops are nested (placed inside one another). Working from the inside outwards, the inner loop (FOR X = A TO B) plots one horizontal side for Y=0 and X=0 to 10. A and B are then changed when this loop is exited (A and B are increased by 15). N is then increased to 2, and the inner loop draws another horizontal line for Y=0 and X=15 to 25. The middle loop (N=1 to 2) is then exited. A is reset to 30, B to 40, and Y to 10 to draw the bottom sides of the last two angles. Z is then changed to 2, and the bottom sides are drawn in a similar manner. (See Figure 57)

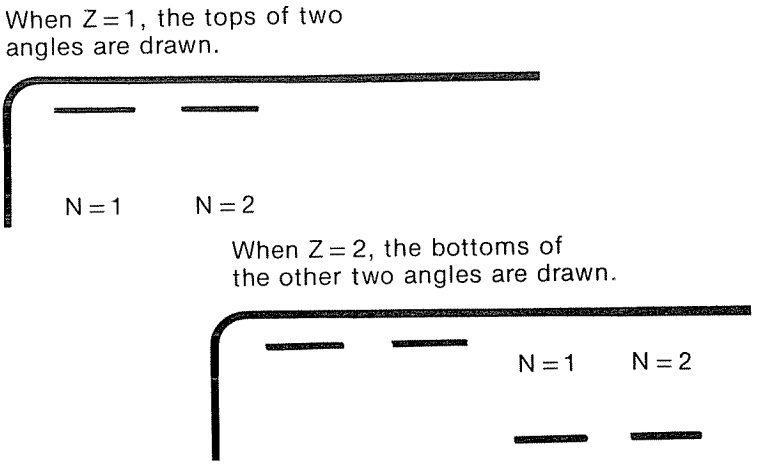


Figure 57

PART II

```
210 X = 0
220  FOR Z==1 TO 2
230   FOR N = 1 TO 2
240    FOR Y = 0 TO 10
250     SET(X,Y)
260    NEXT Y
```

```
270     X = X + 25
280   NEXT N
290     X = 30
300   NEXT Z

310 GOTO 310
```

Once again, three nested loops draw the horizontal lines of the four angles. Two are drawn for Z = 1 and two for Z = 2. N=1 is used for the first side in each case (Z=1, N=1 or Z=2, N=1), and N=2 is used for the second side in each case (Z=1, N=2 or Z=2, N=2). This is illustrated in Figure 58.

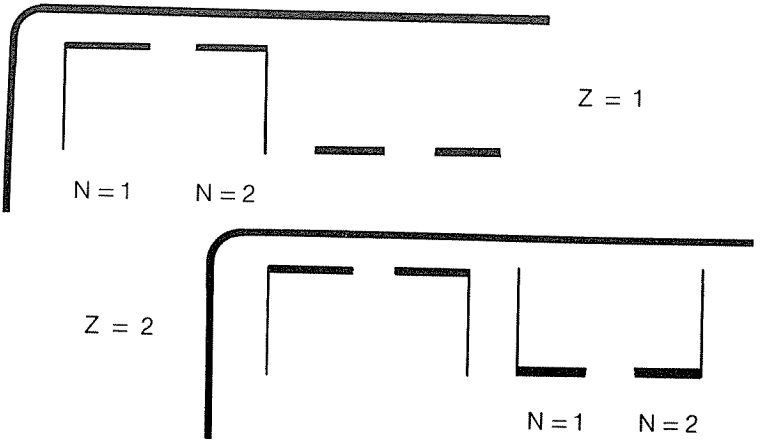
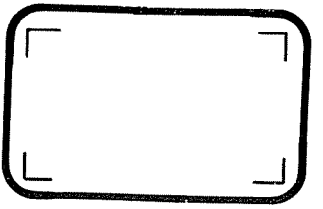


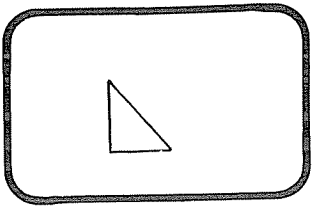
Figure 58

Exercise 20

Modify the Right Angle Program to display the right angles in each corner of the screen as:



You can now draw the right angle for your triangle. Let's now try to add the third side (the hypotenuse). We will draw only one triangle and place it near the screen's center.



RIGHT TRIANGLE PROGRAM

```
100 CLS

110 X = 60
120  FOR Y = 18 TO 27
130   SET(X,Y)                    Draw the left side
140  NEXT Y

150 Y = 27
160  FOR X = 60 TO 69
170   SET(X,Y)                    Draw the bottom
180  NEXT X

190 X = 60: Y = 18
200  FOR N = 1 TO 10
210   SET(X,Y)                    Draw the hypotenuse
220   X = X + 1: Y = Y + 1
230 NEXT N

240 GOTO 240
```

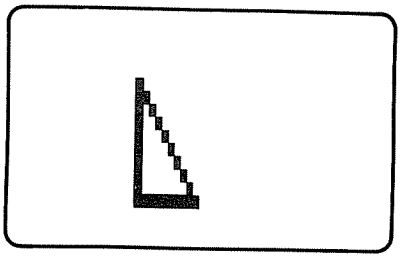
The resulting display looks similar to Figure 59. A table of tabulated values for the hypotenuse is shown on the right of the display.

You could also plot the hypotenuse as shown in Figure 60 by changing lines 190 and 200.

```
190 X = 61: Y= 18
200  FOR N = 1 TO 9
```

Neither program does a perfect job, but the choice is yours.

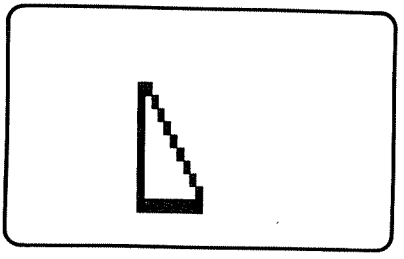
We can complete the job by labeling our triangle in the conventional manner. We do this by replacing the GOTO at line 240 with the statements shown on page 95.



PLOTTED HYPOTENUSE VALUES

N	X	Y
1	60	18
2	61	19
3	62	20
4	63	21
5	64	22
6	65	23
7	66	24
8	67	25
9	68	26
10	69	27

Figure 59



MODIFIED VALUES

N	X	Y
1	61	18
2	62	19
3	63	20
4	64	21
5	65	22
6	66	23
7	67	24
8	68	25
9	69	26

Figure 60

```
240 PRINT AT 412, "B"
250 PRINT AT 604, "C"
260 PRINT AT 612, "A"
270 PRINT AT 730, "RIGHT TRIANGLE ABC"
280 GOTO 280
```

Our finished right triangle now looks like that of Figure 61. Far from perfect, but it is recognizable.

In the right triangle problems, we have been careful to utilize horizontal and vertical lines for the two sides. If you want to get really ridiculous, you might try drawing a right

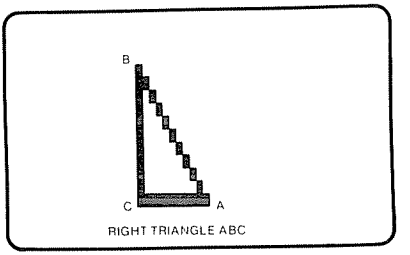


Figure 61

triangle which has its hypotenuse lying vertical or horizontal. Even more ridiculous would be a right triangle with no sides horizontal or vertical.

> Exercise 21
> Change the vertical side of the Right Triangle so that it will be 20 units long and leave the horizontal side at 10 units. Then draw the two sides and the hypotenuse.

Our next attempt will be a triangle with one horizontal side, but no vertical sides. It will be an isosceles triangle (two equal sides). We could think of it as two right triangles (back to back) with their common side omitted. As can be seen from Figure 62, it is a right triangle (I) plus its image (II) reflected about one of its sides.

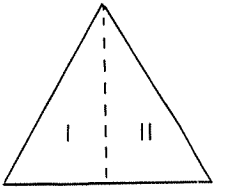


Figure 62

A few simple statements will produce the result we want.

ISOSCELES TRIANGLE PROGRAM

Part I — The horizontal side

```
100 CLS
110 Y = 40
120  FOR X = 20 TO 40
130   SET(X,Y)
140  NEXT X
```

Part II — Both remaining sides at once

```
150  FOR Y = 30 TO 40 : X = Y
160   SET(X,Y)                    right side
170   SET(60—X,Y)                 left side
180  NEXT Y
190 GOTO 190
```

The results of the program are shown in Figure 63. Note that we now have two ragged sides with their rectangles exposed.

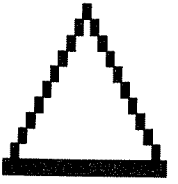
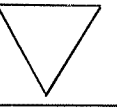


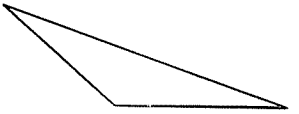
Figure 63

Exercise 22

Invert the isosceles triangle so that the horizontal side is on top.



Our next attempt will be to make the triangle lean back at an angle greater than 90 degrees like this:



An Obtuse Triangle

To do this, we need:

1. a straight line
2. two oblique lines that start at each end of the straight line (A and C) and meet at B.

OBTUSE TRIANGLE PROGRAM

```
100 CLS

110 Y = 20
120  FOR X = 20 TO 40
130   SET(X,Y)
140  NEXT X

150 X = 20
160  FOR Y = 20 TO 10 STEP — 1
170   SET(X,Y)
180   SET(3*X—20,Y)
190   X = X — 1
200  NEXT Y

210 GOTO 210
```

Figure 64 points up the difficulty in drawing some figures. Notice the space between points on the long side. (See additional suggestions for practice at the end of this chapter for a variation of this program.)

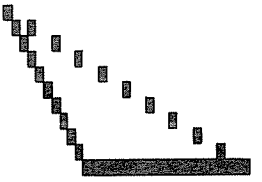


Figure 64

We can approximate any kind of triangle we wish. Some look better than others on the display. It all depends on how far a given side deviates from a horizontal or vertical line.

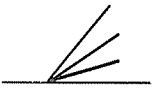
Before leaving the triangle, you may want to explore the possibilities by varying the angle at which a line intersects a vertical line.

ANGLE VARIATION PROGRAM

```
100 CLS

110 Y = 47
120  FOR X = 0 TO 124
130   SET(X,Y)
140  NEXT X

150  FOR A = 1 TO 5
160   FOR Y = 0 TO 47
170    X = INT(0.2*A*(47 — Y)) + 62
180    IF X > 127 THEN 200
190    SET(X,Y)
200   NEXT Y
210   FOR W = 1 TO 500: NEXT W
220  NEXT A

230 GOTO 230
```

Draws horizontal reference line

Draws lines inclined to the right.



The table below shows the calculated values of X at line 170 for various values of A and Y which are used to plot the X,Y points.

	A = 1		A = 3		A = 5	
Y	X	Point	X	Point	X	Point
0	71	(71,0)	90	(90,0)	109	(109,0)
6	70	(70,6)	86	(86,6)	103	(103,6)
12	69	(69,12)	83	(83,12)	97	(97,12)
18	67	(67,18)	79	(79,18)	91	(91,18)
24	66	(66,24)	75	(75,24)	85	(85,24)
30	65	(65,30)	72	(72,30)	79	(79,30)
36	64	(64,36)	68	(68,36)	73	(73,36)
42	63	(63,42)	65	(65,42)	67	(67,42)
47	62	(62,47)	62	(62,47)	62	(62,47)

Exercise 23

Run the angle variation program. Then change the value of the constant (0.2) in line 170 and the range for A in line 150.

By changing line 170 to:

```
170 X = 62 — INT(0.2*A*(47—Y))
```

the line will incline to the left of vertical (have a negative slope) instead of to the right.

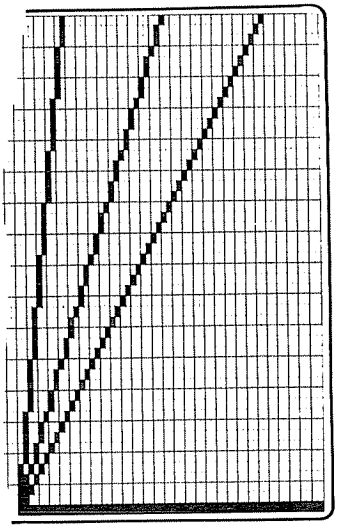


Figure 65 Display of the angle variation program.

Answers to Chapter 9 Exercises

Exercise 20 This is one way — yours may be completely different.

Part I change:
```
170 A = A + 117: B = B  117
190 A = 0: B = 10: Y = 47
```

Part II change:
```
230
264  FOR Y = 37 TO 47
266       SET(X,Y)
270  NEXT Y
280
290 X = 127
```

Lines 230 and 280 completely delete the N loop.

Exercise 21 Once again, this is only one way.

Change:
```
120 FOR Y = 18 TO 37
220 X = X + 1:  Y = Y + 2
240 PRINT AT 220, "B"
```

Exercise 22 And again, this is only one way.

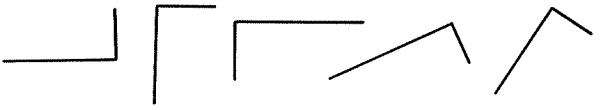
Change:
```
110 Y = 30
160 SET(X—10,Y)
170 SET(70—X,Y)
```

Exercise 23

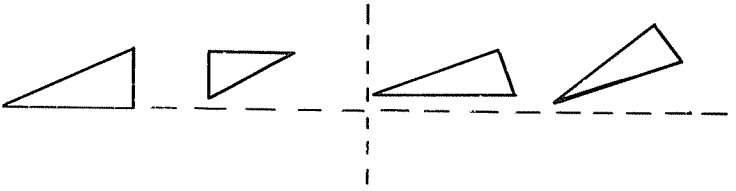
This will vary for the individual programmer. All results should be variations of Figure 65.

Additional Suggestions for Practice

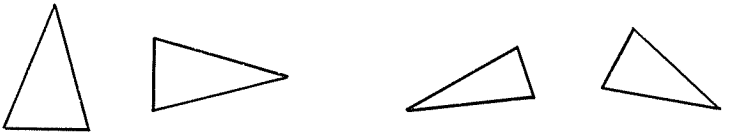
1. Try turning the angle of the Right Angle Program in different directions:



2. If you are successful with suggestion 1, now take the Right Triangle Program and do the same thing.



3. Do the same for the Isosceles Triangle Program.

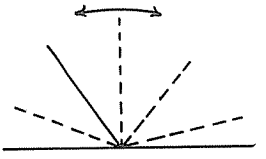


4. Try filling in the missing points in the long side of the obtuse triangle in Figure 64. You might try something like:

```
188 IF Y = 10 THEN 220

192 SET(3X—9,Y)
194 SET(3X—8,Y)
```

5. Modify the Angle Variation Program so that after each new line is drawn, the previous line is erased by the RESET statement.

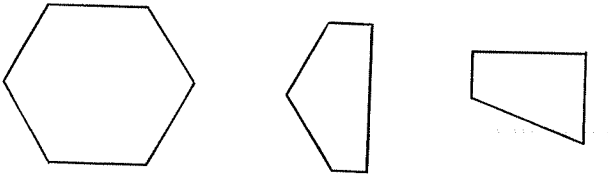
6. Modify your program for suggestion 5 so that the line will move back and forth like a windshield wiper on an automobile.



Make it go back and forth.

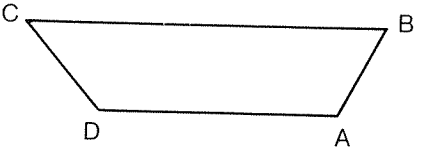
10

OTHER GEOMETRIC FIGURES



We have discussed rectangular and triangular geometric figures in previous chapters. We will now concern ourselves with quadrilaterals and other geometric shapes.

Let's first look at the family of simple quadrilaterals consisting of four sides and four successive pairs of intersections.



Rectangles belong to the family of quadrilaterals. However, you are probably sick and tired of drawing rectangles by this time. Therefore, we will move right past the rectangle and square members of the quadrilateral family.

The family of quadrilaterals is pictured in the pseudo-flowchart shown in Figure 66. Rectangles and squares all have 90 degree angles. They both belong to the group called parallelograms. Members of the parallelogram group do not necessarily have right angles, but they must all have their opposite sides parallel. This is shown in the first

decision box of the flowchart right under the box labeled QUADRILATERALS.

We have already seen that it is easy to draw two parellel sides horizontally or vertically. The other pair of sides is a little more difficult. Since two parallel lines cut off equal segments of another pair of intersecting parallel lines, we know the opposite sides of a parallelogram must be equal in length. Since they are parellel, they must also have the same slope.

Our first program in this chapter will draw a parallelogram. Later, we will tackle other members of the quadrilateral family.

Part I of our Parallelogram Program will draw the horizontal sides, and Part II will draw the other pair of sides. The user inputs the X and Y coordinates for one point, and the program takes care of the rest.

PARALLELOGRAM PROGRAM

Part I

```
10  CLS
20  INPUT "THE X COORDINATE OF POINT D IS"; D
30  INPUT "THE Y COORDINATE OF POINT D IS": E
110  FOR N = 1 TO 2
120   Y = E
130    FOR X = D TO D + 40
140     SET(X,Y)
150    NEXT X
160   D = D + 5
170   E = E — 8
180  NEXT N
```

Part II

```
200 D = D — 10                    Set D and E back to
210 E = E + 16                    original values.
220  FOR N = 1 TO 2
230   FOR Y = E TO E — 8 STEP — 1
240    X = D + INT((E — Y) * 0.625)   Slope-intercept form
250    SET(X,Y)                       of linear equation.
260   NEXT Y
270   D = D + 40
280  NEXT N
290 GOTO 290
```

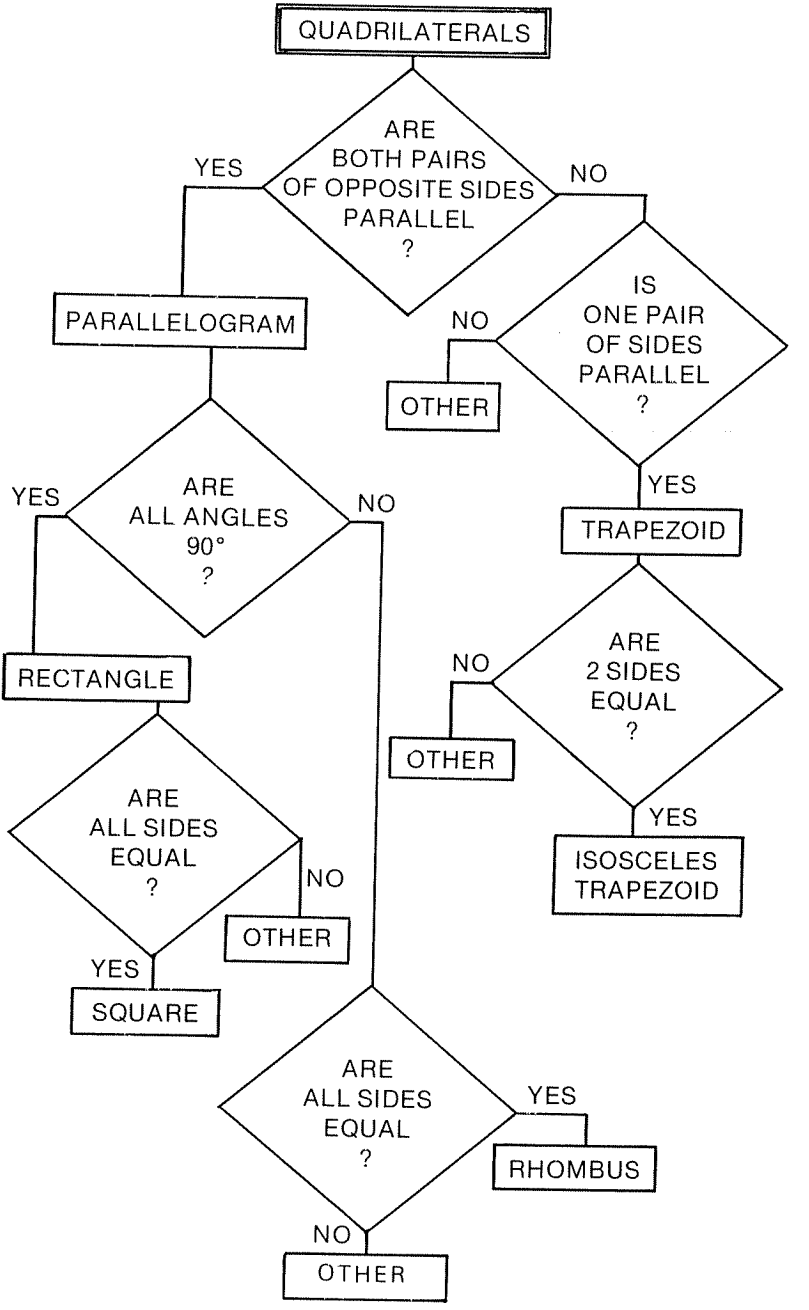

QUADRILATERALS
ARE BOTH PAIRS OF OPPOSITE SIDES PARALLEL ?
YES
NO
PARALLELOGRAM
NO
OTHER
IS ONE PAIR OF SIDES PARALLEL ?
YES
TRAPEZOID
ARE ALL ANGLES 90° ?
YES
NO
RECTANGLE
ARE 2 SIDES EQUAL ?
NO
OTHER
YES
ISOSCELES TRAPEZOID
ARE ALL SIDES EQUAL ?
NO
OTHER
YES
SQUARE
ARE ALL SIDES EQUAL ?
YES
RHOMBUS
NO
OTHER

Figure 66

Figure 67 gives a representation of the display and shows the results of the calculations in line 240. In Part II, when N = 1, the value for D is 10. When N = 2, the value for D has been changed to 50 by line 270 of the program.

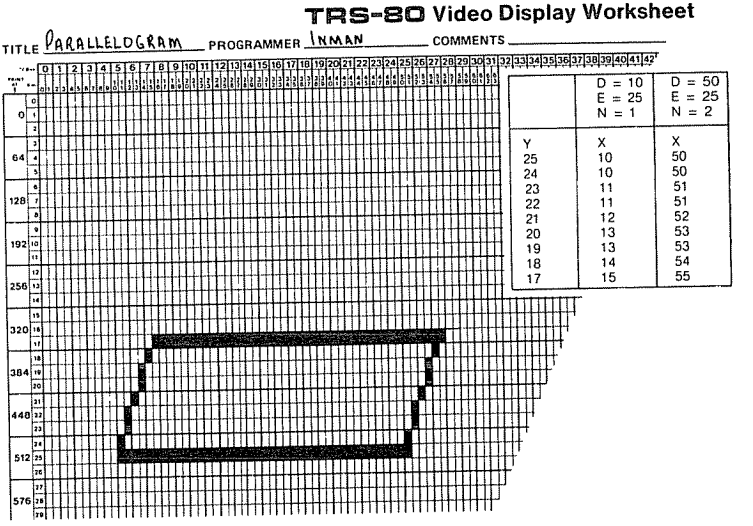


Figure 67

Once again, the non-horizontal sides are not true lines, but look more like a series of irregular stair steps.

A variation of the parallelogram could provide some perspective by making the top side shorter than the bottom. The two sides would not be parallel in that case, but let's see what it might look like.

We will move the top line off into the distance and shorten it. This will require some program modification. We will make it specific to one particular location, but still use the Parallelogram Program as a guide.

STRETCH THE PARALLELOGRAM PROGRAM

```
100 CLS
110 Y = 25
120  FOR X = 10 TO 50
130   SET(X,Y)
140  NEXT X
```

```
150 Y = 1
160  FOR X = 58 TO 65
170   SET(X,Y)
180  NEXT X

200  FOR Y = 25 TO 1 STEP — 1
210   X = 10 + (25 — Y) * 2
220   SET(X,Y)
230   X = 50 + ((25 — Y) * .625)
240   SET(X,Y)
250  NEXT Y

260 GOTO 260
```

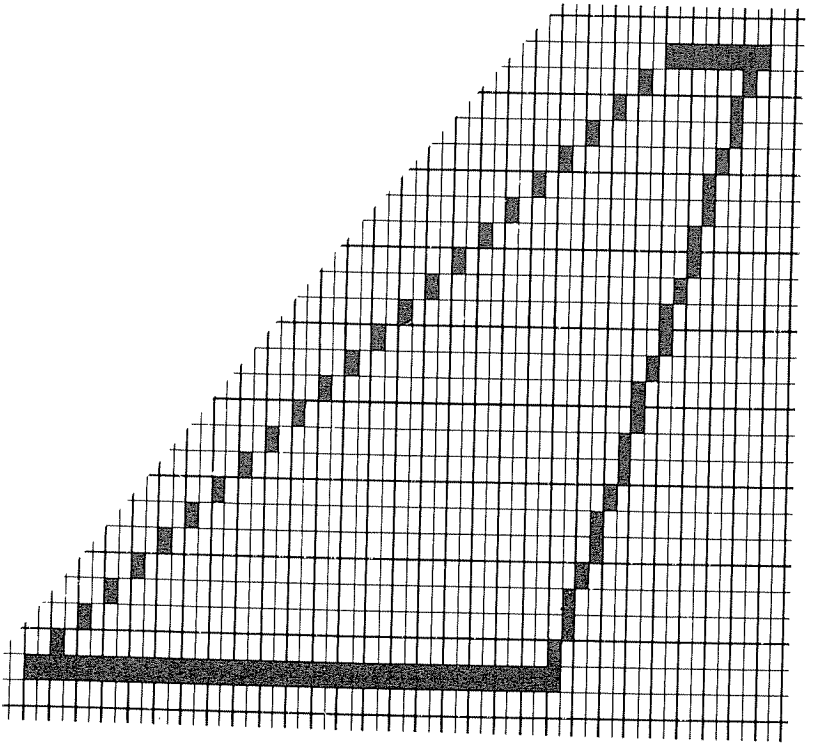


Figure 68

This program produces a result similar to Figure 68. All of you who are students of geometry will realize that the

shape shown in Figure 68 is a trapezoid, another member of the quadrilateral family.

Looking at the flowchart of Figure 66, the first decision box asks, "Are both pairs of opposite sides parallel?" From the display, we would now have to say, "No." We would then branch to the right and the next decision box asks, "Is one pair of sides parallel?" The answer is yes. Therefore, we arrive at the trapezoid. If we proceed to the next decision box, "Are two sides equal," the answer is no. Therefore, we do not have an isosceles trapezoid.

> Exercise 24
> Write a program to modify the shape of Figure 68 so that it is an *isosceles* trapezoid.

The trapezoid program we used is too specific. If programs are to be used more than once, they should be written in a more general form. In our case, we should be able to input parameters which can change the size and shape of our geometric figure.

Our next program for drawing a trapezoid will do this. It also states restrictions and gives REMarks to explain each section of the program. Refer to Figure 69 when using the program. Notice how this method of program writing improves its readability.

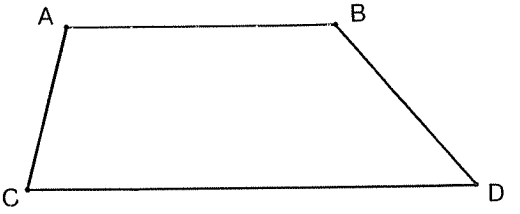


Figure 69

TRAPEZOID PROGRAM

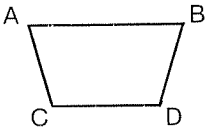
```
10 REM GIVEN: VERTICLES A,B,C,D WITH LINES
     PARALLEL TO
20 REM          AND ABOVE CD.
```

```
30  REM GIVEN: B LIES TO THE RIGHT OF A.
40  REM GIVEN: D LIES TO THE RIGHT OF C.
50  REM GIVEN: BOTH LINES AB AND CD ARE HORIZONTAL.

100 INPUT "THE X COORDINATE OF A IS"; A
110 INPUT "THE X COORDINATE OF B IS"; B
120 INPUT "THE X COORDINATE OF C IS"; C
130 INPUT "THE X COORDINATE OF D IS"; D
135 CLS
140 REM DRAW AB

150 INPUT "THE Y COORDINATE OF A IS";E
160  FOR X = A TO B
170   SET(X,E)
180  NEXT X

190 REM DRAW CD

200 INPUT "THE Y COORDINATE OF C IS";F
210  FOR X = C TO D
220   SET(X,F)
230  NEXT X

240 REM DRAW AC

250 Z = (A — C)/(F — E)
260  FOR Y = E TO F
270   X = A — INT(Z * (Y — E))
280   SET(X,Y)
290  NEXT Y

300 REM DRAW BD

310 Z = (B — D)/(F — E)
320  FOR Y = E TO F
330   X = B — INT(Z * (Y — E))
340   SET(X,Y)
350  NEXT Y

360 GOTO 360
```

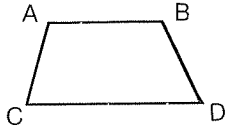
You should run the trapezoid program many times, experimenting with different values for A, B, C, D, E, and F. Some of your trapezoids will look nice, and others will hardly be recognizable.

To form an *isosceles* trapezoid, the difference between the X coordinates for A and C (below) should be the same as the difference between the X coordinates for B and D.

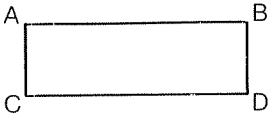
It can be like this



or this



Of course, if the X coordinates for A and C are the same, and the X coordinates for B and D are the same, then we get our old friend, the rectangle.



We will not go into the technicalities of whether the rectangle is a special case of the trapezoid or not. It all depends upon the definition of the trapezoid. Some say that a trapezoid has two sides parallel. Others say that a trapezoid has *only* two sides parallel. According to our flowchart, we are using the latter definition. Therefore, we would not classify the rectangle as a special trapezoid.

The above variations of the trapezoid should be tried along with many others. Making the sides of linear shapes intersect as you desire is a good exercise in controlling your computer.

> Exercise 25
>
> Run the trapezoid program so that A and B coincide (have the same X and Y coordinates). What kind of shape do you have?

If we increase the number of sides of a quadrilateral by one, we will have a *pentagon* (a 5-sided polygon). This can be most easily accomplished by modifying our trapezoid program. We would have to eliminate the section which drew side AB and insert statements to draw sides AE and BE of the sketch in Figure 70.

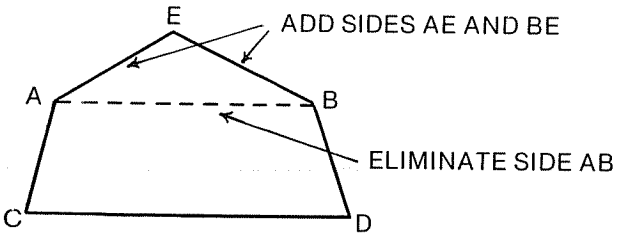


Figure 70

The X coordinate of point E in our new program will lie mid-way between that of A and B. The Y coordinate of E will be less than that of A and B. Here are the suggested modifications.

```
140 INPUT "THE Y COORDINATE OF E IS"; M

160 REM DRAW AE AND BE

170 R = INT((A + B) /2)
172 Z = INT((R — A) / (E —M))
174  FOR Y = M TO E
176    X = R — Z*(Y — M)
178    SET(X,Y)
180    SET(2*R—X,Y)
182  NEXT Y
```

> Exercise 26
> Place the Y coordinate of E equal to the Y coordinate of A. What kind of shape results?

One interesting result which you may experience will occur when the Y coordinate of E is chosen so that side AE has the same slope as side AC. The pentagon then reduces to a quadrilateral with AE being merely an extension of side ACas in Figure 71.

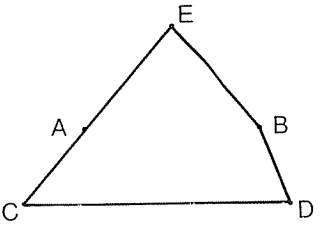


Figure 71

The same thing could happen with BE and BD, or it could happen with both.

One other interesting occurence is pictured in Figure 72. One part of the pentagon seems to be "caved in".

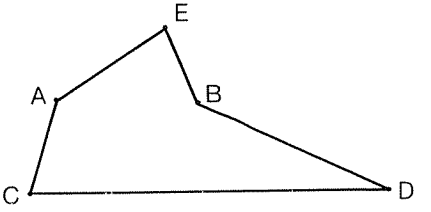


Figure 72

We call this a *concave pentagon* as opposed to a *convex pentagon.* Both are pictured in Figure 73.

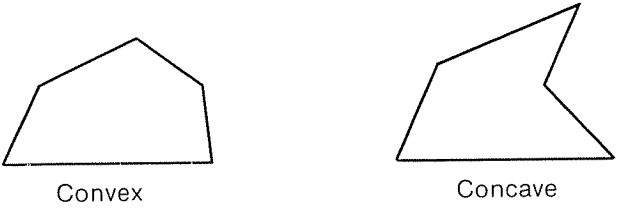


Figure 73

We can extend the number of sides to as large a number (n) as we want. This general class of figures is the family of polygons (or n-gons). The figure resulting from each unique value of n is assigned a name:

3 - Triangle
4 - Quadrilateral
5 - Pentagon
6 - Hexagon
etc.

The hexagon is the next step up from the pentagon and it can be easily developed from a rectangle as shown in Figure 74. A program to accomplish this follows.

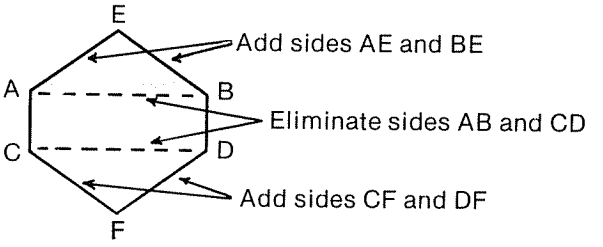


Figure 74

HEXAGON PROGRAM

```
100 CLS
110 X = 10
120  FOR N = 1 TO 2                    Draw AC and BD
130   FOR Y = 10 TO 15
140    SET(X,Y)
150   NEXT Y
160   X = X + 40
170  NEXT N

200  FOR Y = 10 TO 0 STEP — 1
210   X = 10 + 2 * (10 — Y)            Draw AE and BE
220   SET(X,Y)
230   SET(60—X,Y)
240  NEXT Y

300  FOR Y = 15 TO 25
310   X = 10 + 2 * (Y — 15)            Draw CF and FD
320   SET(X,Y)
330   SET(60—X,Y)
340  NEXT Y
350 GOTO 350
```

Exercise 27

Place E Below AB, and place F above CD (but keep E above F). Draw a sketch of the result.

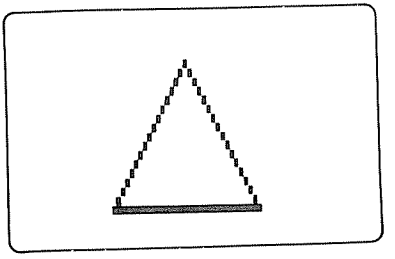
Answers to Chapter Ten Exercises

Exercise 24

```
160     FOR X = 26 TO 34
210     X = INT(10 + (25 — Y) * 2 / 3)
220     SET(X,Y)
230
240     SET(60 — X,Y)
```

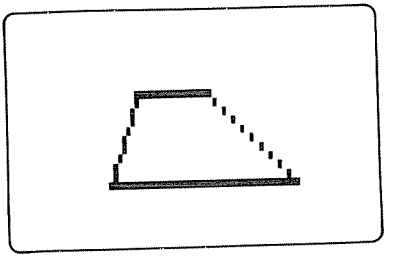
Exercise 25

A Triangle (shapes will vary)



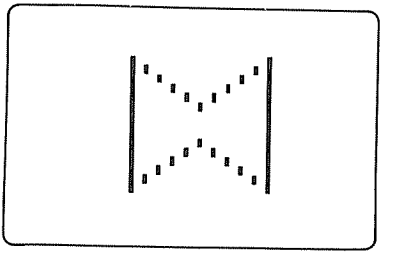
Exercise 26

A Trapezoid (shapes will vary)



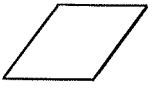
Exercise 27

Concave Hexagon (shapes will vary)



Additional Suggestions for Practice

1. Using the Parallelogram Program, try to draw what looks like a Rhombus.



2. Try drawing a shape similar to Figure 67 with two curved sides.

3. Try making solid figures.

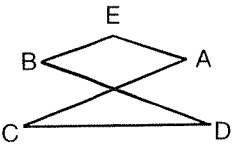
4. Write a program which can draw a given shape and then erase sides and change the shape or size.

5. Modify the Trapezoid Program to draw a hexagon.

6. Try an 8-sided polygon (octagon).

7. Experiment with the hexagon until you arrive at one which best approximates a circle. How about an octagon — would that work better?

8. Can you make the sides of a concave pentagon intersect like this?



11

FAR OUT IDEAS

The printing of characters from the TRS-80 keyboard onto the video screen is itself a form of graphics. Each character is formed by the selection of certain points within a 5 by 7 matrix shown in Figure 75. The points within the matrix which are turned on by a given keystroke are pre-determined by electronic components within the computer.

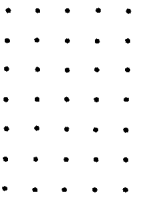


Figure 75

For example, type in the letter A on your TRS-80. By adjusting the brightness of your video monitor, you should be able to see the individual points which have been turned on. Monitor displays of the letters A, B and C are shown in Figure 76.

If our graphics capability were fine enough to display each one of the points within a printed character, think what kind of pictures we could draw! Since there are 1024

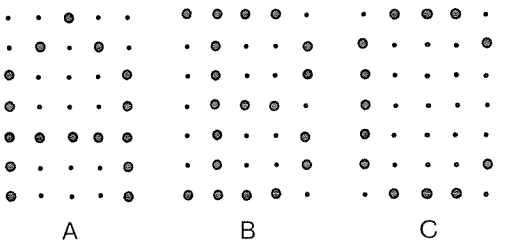


Figure 76

print positions and 5 x 7 (or 35) points within each printed character, that would be over 65,000 positions. There are also spaces between printed characters and between lines of characters, so there would actually be more positions than that. Well, let's be practical. For the price of the TRS-80, we can't expect that. Maybe some day...

For the time being, let's be content with what we have. We will duplicate some of the printed characters by means of our plot positions starting with the letter A. Imagine the 5 x 7 matrix as plot positions on your screen. Figure 77 shows the arrangement we will use.

(11,1) (12,1) (13,1) (14,1) (15,1)
(11,2) (12,2) (13,2) (14,2) (15,2)
(11,3) (12,3) (13,3) (14,3) (15,3)
(11,4) (12,4) (13,4) (14,4) (15,4)
(11,5) (12,5) (13,5) (14,5) (15,5)
(11,6) (12,6) (13,6) (14,6) (15,6)
(11,7) (12,7) (13,7) (14,7) (15,7)

Figure 77

Exercise 28

In Figure 77, circle the pairs of coordinates which would be lighted to form the A shown in Figure 76.

To print our A, we need to turn on points (11,7), (11,6), (11,5), (11,4), (11,3), (12,2), (13,1), (14,2), (15,3), (15,4),

(15,5), (15,6), (15,7), (12,5), (13,5), and (14,5). We will use a data statement in our program and read in the coordinates.

PROGRAM TO PRINT AN A

```
100  CLS

110   FOR N = 1 TO 16
120    READ X,Y
130    SET(X,Y)
140   NEXT N

150  GOTO 150

1000 DATA 11,7,11,6,11,5,11,4,11,3,12,2,13,1,14,2,15,3,
1010 DATA 15,4,15,5,15,6,15,7,12,5,13,5,14,5
```

That's all there is to it. Put in your computer and RUN. You have duplicated the printed character A, but have made it much larger.

Now let's get a little fancier and print TRS-80. You can predict that the number of data statements will be quite long for this program. The program stays the same except for:

```
110  FOR N = 1 TO 82

1000 DATA 11,1,12,1,13,1,14,1,15,1
1010 DATA 13,2,13,3,13,4,13,5,13,6,13,7          T = 11 data reads

1020 DATA 21,1,22,1,23,1,24,1,21,2,25,2,21,3,25,3
1030 DATA 21,4,22,4,23,4,24,4,21,5,23,5,21,6,24,6      R = 18
1040 DATA 21,7,25,7

1050 DATA 32,1,33,1,34,1,31,2,35,2,31,3,32,4,33,4
1060 DATA 35,4,35,5,31,6,35,6,32,7,33,7,34,7           S = 15

1070 DATA 41,4,42,4,43,4,44,4,45,4                     — = 5

1080 DATA 52,1,53,1,54,1,51,2,55,2,51,3,55,3,52,4
1090 DATA 53,4,54,4,51,5,55,5,51,6,55,6,52,7,53,7      8 = 17
1100 DATA 54,7

1110 DATA 62,1,63,1,64,1,61,2,65,2,61,3,65,3,61,4,65,4
1120 DATA 61,5,65,5,61,6,65,6,62,7,63,7,64,7           0 = 16

                                        Total data reads = 82
```

Figure 78 shows a representation of the video display when the program is run.

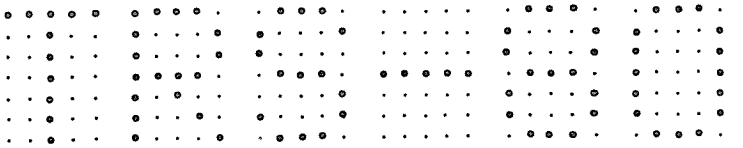


Figure 78

LEAVE THE DATA IN THE COMPUTER!!! You'll need it for this exercise and the next program.

> Exercise 29
> Modify the TRS-80 display of Figure 78 to make it blink. (Hint: add one line to make a time delay. Also change line 150 to GOTO a different line than 150.)

Now let's make it even more exciting. Instead of making it blink, we will light one letter at a time from left to right and make a moving display. The data is the same as before, but the main part of the program will be changed.

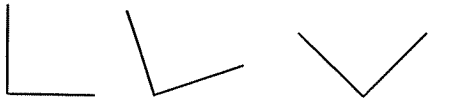
MOVING DISPLAY PROGRAM

```
100 CLS

110 A = 11
120  FOR R = 1 TO 6
130   FOR S = 1 TO A
140    READ X,Y
150    SET(X,Y)
160   NEXT S

170   FOR W = 1 TO 300
180   NEXT W

190   CLS

200   IF R = 1 THEN A = 18
210   IF R = 2 THEN A = 15
220   IF R = 3 THEN A = 5
```

```
230   IF R = 4 THEN A = 17
240   IF R = 5 THEN A = 16
250  NEXT R

260 RESTORE
270 GOTO 110
```

There are many variations that you can try with this display and the previous one. A moving display can be quite fascinating. That is why you see it so often on billboard displays, movie marquees, etc.

You might even want to attempt some animation such as that used in movie cartoons. For example, imagine the letter L being tilted, or revolved, until it becomes a V as in Figure 79.

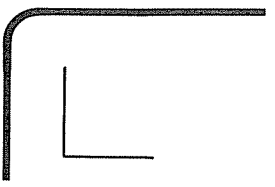


1. Print it 2. Tilt it 3. Tilt it more

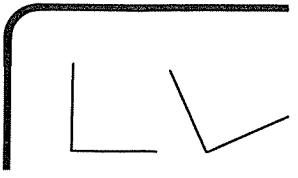
Figure 79

L TO V PROGRAM

```
100  CLS

110   FOR Z = 1 TO 7
120    READ A,B,C,D
130    SET(C,D)          Prints L
140   NEXT Z

150   FOR W = 1 TO 1000
160   NEXT W
170  RESTORE

200   FOR Z = 1 TO 7
210    READ A,B,C,D
220    M = C + 0.5 * (A — C)
230    N = D — 0.5 * (D — B)
240    SET(M,N)
250   NEXT Z

260   FOR W = 1 TO 1000
270   NEXT W
280  RESTORE
```

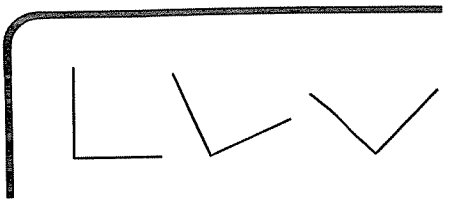


Turns L
at an angle.



```
300   FOR Z = 1 TO 7
310    READ A,B,C,D
320    SET(A,B)
330   NEXT Z
340   FOR W = 1 TO 2000
350   NEXT W
360  RESTORE
370  GOTO 100
1000 DATA 114,0,0,0
1010 DATA 116,2,0,2
1020 DATA 118,4,0,4
1030 DATA 120,6,0,6
1040 DATA 122,4,2,6
1050 DATA 124,2,4,6
1060 DATA 126,0,6,6,
           A B C D
```

Turns angled L to a V



This may seem to waste space, but it does make it clear that each line represents new values for A,B,C, and D.

Exercise 30

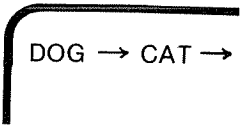
Modify the L TO V Program so that L appears first on the screen, then it is replaced in the same approximate location by the slanted L, and finally, that it is replaced by the V. This will *approximate* an animation of the L being converted to a V. More intermediate positions and a faster plotting rate would make the animation more realistic.

Now we can move shapes around on the screen, turn them on and off, and even change their shape. The next thing to try is moving complete words around. The keyboard will provide the characters this time. You won't have to plot them.

Let's start with the words CAT and DOG. It seems natural to have the DOG chase the CAT around the screen. We will use the string variables A$ and B$ for the words CAT and DOG. A simple loop using the PRINT AT statement should accomplish our task.

CAT AND DOG PROGRAM

```
100 CLS

110 A$ = CAT
120 B$ = DOG
130 A = 0

140 PRINT AT A, A$
150  FOR W = 1 TO 100    The CAT shows up.
160  NEXT W

170 PRINT AT A, " "

180 A = A + 8
190 PRINT AT A — 16, " "
200 PRINT AT A, A$
210 PRINT AT A — 8, B$, A$  The chase
220  FOR W = 1 TO 100          is on.
230  NEXT W
240 IF A>= 950 THEN 260  Did the CAT
250 GOTO 180                get away?

260 CLS
270 PRINT "THE CAT GOT AWAY."
280 GOTO 280                Yes, the CAT finally got away.
```



How could the method of the CAT and DOG Program be used in a practical situation? One that comes to mind is in a drill problem in an English class in an elementary school. We will use a simple example to demonstrate one possibility.

A sentence is printed with the verb missing. Possible verbs are listed, and the student is asked to select the correct verb from the list. The selected verb is then moved in several steps closer and closer to the blank space in the sentence. At last, it arrives in the correct spot. The student is then told whether or not she, or he, is correct. If incorrect, the student is told the correct response. Figure 80 shows the display at the start of the problem.

```
10  REM: READ IN DATA AND PRINT THE PROBLEM
20  CLS
30  READ A$
40  READ B$
50  DATA 1.IS  2.ARE  3.HAVE  4.WILL
```

```
60  PRINT AT 320, A$
70  PRINT AT 336, B$
80  PRINT AT 448, "PICK THE NUMBER OF THE
      CORRECT ANSWER."
90  PRINT AT 74, "THE CAT AND DOG   EATING THEIR
      FOOD.";: INPUT Q                    6 spaces
95  PRINT AT 448," "
100 REM : ROUTING OF CHOICE TO THE CORRECT
      SUBROUTINE
110 IF (Q<>1) * (Q<>2) * (Q<>3) * (Q<>4) THEN 80
120 IF Q = 1 GOSUB 300: GOTO 250
140 IF Q = 2 GOSUB 500: GOTO 280
160 IF Q = 3 GOSUB 700: GOTO 250
180 IF Q = 4 GOSUB 900
240 REM : PRINT MESSAGES
250 PRINT AT 256, "I'M SORRY BUT THE CORRECT
      ANSWER IS"
260 PRINT "THE CAT AND DOG ARE EATING THEIR FOOD."
270 GOTO 270
280 PRINT AT 256, "THAT'S CORRECT."
290 GOTO 270
300 REM : SUBROUTINE FOR IS      7 spaces  3 spaces
310 PRINT AT 320, "      2.ARE  3.HAVE  4.WILL"
320 PRINT AT 263, "IS"
330  FOR W = 1 TO 100
340  NEXT W
350 PRINT AT 263, " "
360 PRINT AT 204, "IS"
370  FOR W = 1 TO 100
380  NEXT W
390 PRINT AT 204, " "
400 PRINT AT 145, "IS"
410  FOR W = 1 TO 100
420  NEXT W
430 PRINT AT 145, " "
440 PRINT AT 90, "IS EATING THEIR FOOD."
450 RETURN
500 REM : SUBROUTINE FOR ARE
```

```
510 PRINT AT 320, "1.IS          3.HAVE  4.WILL"
520 PRINT AT 272, "ARE"
530  FOR W = 1 TO 100          11 spaces  3 spaces
540  NEXT W
550 PRINT AT 272, " "
560 PRINT AT 213,"ARE"
570  FOR W = 1 TO 100
580  NEXT W
590 PRINT AT 213, " "
600 PRINT AT 154, "ARE"
610  FOR W = 1 TO 100
620  NEXT W
630 PRINT AT 154, " "
640 PRINT AT 90, "ARE EATING THEIR FOOD."
650 RETURN                                  3 spaces
700 REM: SUBROUTINE FOR HAVE                12 spaces
710 PRINT AT 320, "1.IS  2. ARE             4.WILL"
720 PRINT AT 281, "HAVE"
730  FOR W = 1 TO 100
740  NEXT W
750 PRINT AT 281, "    "
760 PRINT AT 218, "HAVE"
770  FOR W = 1 TO 100
780  NEXT W
790 PRINT AT 218, " "
800 PRINT AT 154, "HAVE"
810  FOR W = 1 TO 100
820  NEXT W
830 PRINT AT 154, " "
840 PRINT AT 90, "HAVE EATING THEIR FOOD."
850 RETURN

900 REM : SUBROUTINE FOR WILL                3 spaces
910 PRINT AT 320, "1.IS  2.ARE  3.HAVE"
920 PRINT AT 282, "WILL"
930  FOR W = 1 TO 100
940  NEXT W
950 PRINT AT 282, " "
960 PRINT AT 218, "WILL"
970  FOR W = 1 TO 100
980  NEXT W
990 PRINT AT 218, " "
1000 PRINT AT 154, "WILL"
1010  FOR W = 1 TO 100
1020  NEXT W
```

```
1030 PRINT AT 154, " "
1040 PRINT AT 90, "WILL EATING THEIR FOOD."
1050 RETURN
```

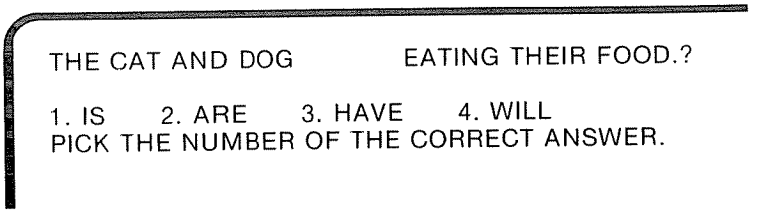
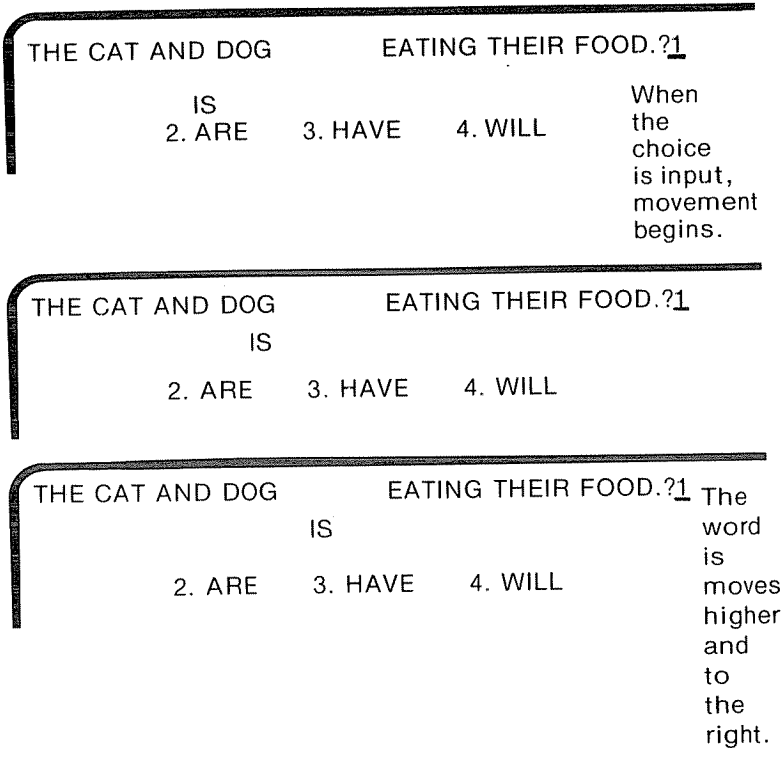


Figure 80

Displays of a typical run are shown in Figure 81. This is but one isolated example. You can surely think of many more practical uses. Many games can be animated in a similar way. Extend the limit of your imagination.



THE CAT AND DOG IS EATING THEIR FOOD."

2. ARE 3. HAVE 4. WILL

I'M SORRY BUT THE CORRECT ANSWER IS
'THE CAT AND DOG ARE EATING THEIR FOOD.'

Response

Figure 81

In addition to moving things around on the screen, you can also draw simulated three dimensional shapes. Expanding on our geometric figures of Chapter 8, we could construct the simplest solid figure by drawing quadrilateral surfaces.

A rectangle is used for the front surface with parallelograms forming the top and side surfaces as in the sketch of Figure 82.

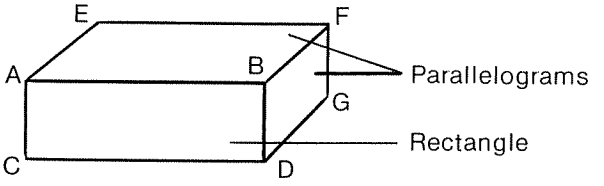


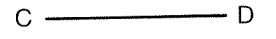
Figure 82

We have drawn these shapes before. Therefore, it's just a matter of putting them all together.

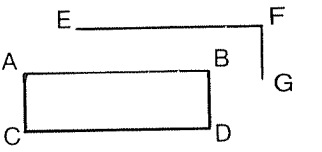
QUADRILATERALS IN 3-D PROGRAM

```
100 CLS

110 INPUT "THE X VALUE FOR POINT A IS"; A
120 INPUT "THE Y VALUE FOR LINE AB IS"; R
130 INPUT "THE X VALUE FOR POINT B IS"; B
140 INPUT "THE Y VALUE FOR LINE CD IS"; S
```

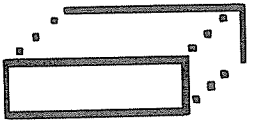
```
150 INPUT "THE X VALUE FOR POINT E IS"; E
160 INPUT "THE Y VALUE FOR LINE EF IS"; T
170 CLS

200 Y = S
210  FOR X = A TO B
220   SET(X,Y)          Draws CD
230  NEXT X

240 Y = R
250  FOR X = A TO B
260   SET(X,Y)          Draws AB
270  NEXT X

300 Y = T
310 F = E + B - A
320  FOR X = E TO F
330   SET(X,Y)          Draws EF
340  NEXT X

350  FOR Y = R TO S
360   SET(A,Y)
370   SET(B,Y)
380  NEXT Y

400  FOR Y = T TO T + S - R
410   SET(B + E - A + 1,Y)
420  NEXT Y

430  FOR Y = T TO R - 1
440   X = E - (E - A) / (R - T) * (Y - T) + 1
450   SET(X,Y)
460   SET(X + B - A,Y)
470  NEXT Y

480 Q = T + S - R
490  FOR Y = Q TO S - 1
500   X = B - (F - B) / (R - T) * (Y - S) + 1
510   SET(X,Y)
520  NEXT Y

530 GOTO 530
```



Vertical's added



Last sides added



A typical run of this program is shown in Figure 83. First the input values are shown and then the final display.

```
THE X VALUE FOR POINT A IS? 10
THE Y VALUE FOR LINE AB IS? 12
THE X VALUE FOR POINT B IS? 40
THE Y VALUE FOR LINE CD IS? 17
THE X VALUE FOR POINT E IS? 18
THE Y VALUE FOR LINE EF IS? 8
```

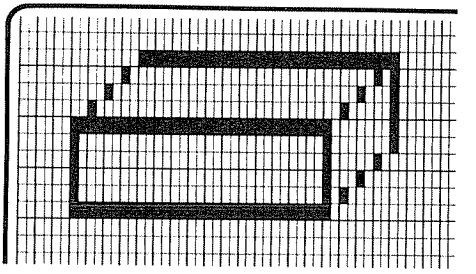


Figure 83

To conclude this chapter, and the book, we will show you a simulated Target Game concocted by the author and modified by his 14-year-old son. It is a two player game, and needless to say, my son beats me consistently.

TARGET GAME PROGRAM

```
10  REM : THIS TARGET GAME IS FOR 2 PLAYERS
20  REM : EACH PLAYER WILL HAVE 5 TARGETS (X or 0)
30  REM : EACH PLAYER WILL HAVE 10 SHOTS
40  REM : GUN SETTING RANGES FROM — 2 (SHOOTS TO LEFT)
50  REM : TO + 2 (SHOOTS TO THE RIGHT)
60  REM : A ZERO SETTING SHOOTS STRAIGHT AHEAD
70  REM : DECIMAL FRACTIONS MAY BE USED (0.5, -1.2, ETC.)

80  PRINT "IF YOU WANT A BRIEF DESCRIPTION PRESS
      THE BREAK KEY"
85  PRINT "THEN LIST THE PROGRAM. INSTRUCTIONS
      LINES 10 — 70."
90   FOR W = 1 TO 1500
95   NEXT W

100 CLS
110  FOR Z = 3 TO 7
120   M = RND(31)
130   N = RND(31)                    Display targets
140   IF M = 31 THEN M = 0
```

```
150   N = N + 31
160   PRINT AT M + Z * 64, "X"
170   PRINT AT N + Z * 64, "0"
180  NEXT Z

200 Y = 43
210  FOR X = 0 TO 127          Set gun line
220   SET(X,Y)
230  NEXT X

240 X = 63
250  FOR Y = 3 TO 42           Player dividing line

260   SET(X,Y)
270  NEXT Y

280 X = 31
290 Y = 42
300 SET(X,Y)
                               Set gun locations
310 X = 96
320 Y = 42
330 SET(X,Y)

340 C = 0                      Set shot counters
350 D = 0                      to zero.

400 PRINT AT 0, " "                                  Player #1
410 PRINT AT 0, "WHAT GUN SETTING";                  aim your shot.
420 INPUT R

430 PRINT AT 0, " "
440 PRINT AT 0, "SHOT #"; C + 1        Shot count is displayed

450 X = 31
460  FOR Y = 41 TO 3 STEP -1
470   X = X + R
480   IF (X>61) + (X<1) THEN 540       There goes the shot.
490   SET(X,Y)
500    FOR W = 1 TO 10
510     NEXT W
520   RESET(X,Y)
530  NEXT Y

540 C = C + 1                  Shot counter increased.

600 PRINT AT 0," "
610 PRINT AT 33, "WHAT GUN SETTING";   Player #2
620 INPUT S                            aim your shot.
```

```
630 PRINT AT 0, " "
640 PRINT AT 33, "SHOT #"; D + 1      Shot count displayed.
650 X = 96
660  FOR Y = 41 TO 3 STEP — 1
670   X = X + S
680   IF (X >126) + (X<66) THEN 740   There goes the shot.
690   SET(X,Y)
700    FOR W = 1 TO 10
710    NEXT W
720   RESET(X,Y)
730  NEXT Y
740 D = D + 1                         Shot counter increased.
750 IF D = 10 THEN 770
760 GOTO 400
770 PRINT AT 960, "OUT OF AMMO!! PLAYER WITH FEWEST
      TARGETS LEFT WINS"
780 GOTO 780
```

The Target Game is by no means complete. But it should give you and yours many hours of enjoyment. You will no doubt want to make modifications of your own choice. Have fun with this and other programs in this book. I hope you enjoy it as much as I have enjoyed writing it.

Answers to Chapter 11 Exercises

Exercise 28

Pairs circled to form the A should be:
(11,7), (11,6), (11,5), (11,4), (11,3)
(12,2), (13,1), (14,2)
(15,3), (15,4), (15,5), (15,6), (15,7)

Exercise 29

One way to do it:

```
145 FOR W = 1 TO 100 : NEXT W
150 GOTO 100
```

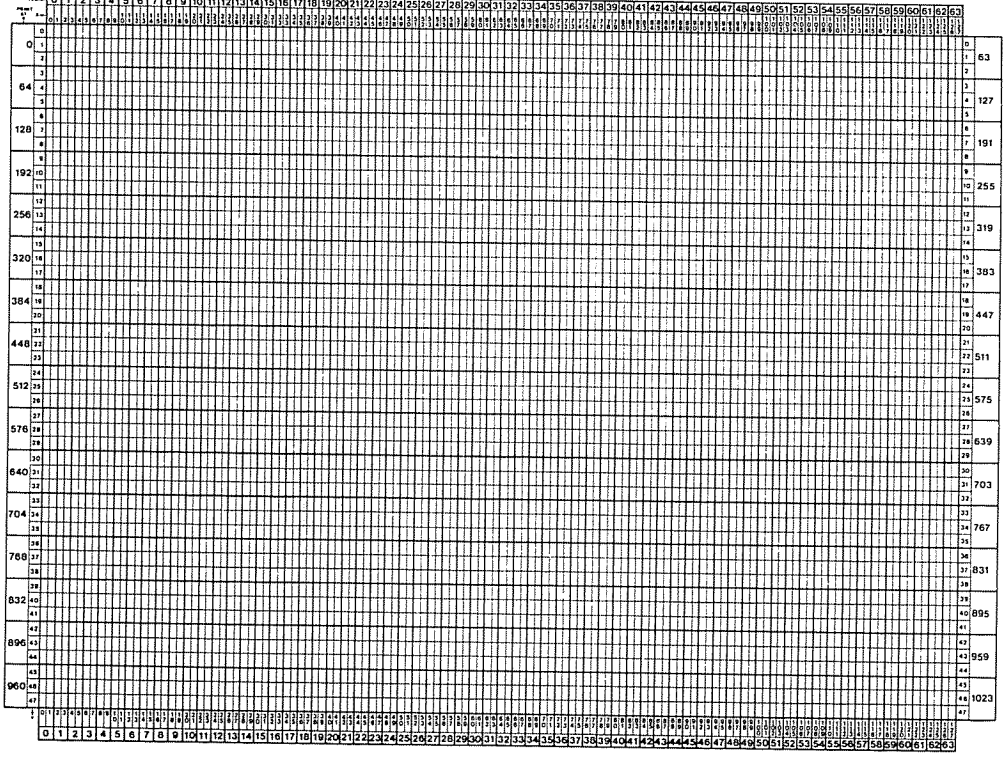
Exercise 30

```
180 CLS
240  SET(M—57,N)
290 CLS
320   SET(A—114,B)
```

Here is our attempt at animation.

TRS-80 Video Display Worksheet

TITLE _____ PROGRAMMER _____ COMMENTS _____ PAGE ____ OF ____



APPENDIX

This appendix is provided so that the programs in this book may be modified to run in LEVEL II BASIC. It is organized by chapters and references are made to individual pages in this book. Programs have not been optimized for length or speed. They are designed for ease in learning to use individual commands and statements.

CHAPTER 1—WHERE ARE WE GOING?

Reference is found on page 1 to the Level I PRINT AT statement. This statement has been changed to PRINT @ for Level II. Numerous programs in the book use this statement which *must* be changed if you are using LEVEL II.

CHAPTER 2—THE VIDEO DISPLAY

- Page 11 contains two statements using PRINT AT. For Level II these must be changed to:

```
50 PRINT @66, "THIS BEGINS AT 66"
70 PRINT @130,A
```

The screen at the bottom of page 11 will look the same.

- Page 12—The program at the bottom of the page has two lines which must be changed. They both contain PRINT AT statements. The changed lines for Level II are:

```
10 PRINT @263,"S"
20 PRINT @266,"<--S IS AT PRINT POSITION 263"
```

- Page 14—Change line 10 to:

```
10 PRINT @263,"S <-- S IS AT PRINT POSITION 263"
```

- Page 15—The program lines 12 and 14 (to be added to the previous program) must both be changed from PRINT AT TO PRINT @.

```
12 PRINT @327,"P     --P IS AT PRINT POSITION 327"
14 PRINT @391,"T     --IS AT PRINT POSITION 391"
```

  Bottom of page 15 refers to PRINT AT (PRINT @ for Level II).

- Page 18—A use of PRINT AT in suggested practice #4 should be PRINT @.

CHAPTER 3—THE PLOT IS EXPANDED

- Page 20—When negative values are used for X or Y in the statement SET(X,Y), an error occurs—the screen shows the following for either of the programs on this page.

```
?FC ERROR IN 30
READY
>_
```

Illegal function call error

- Page 21—The "wrap around" characteristic of Level I is not present in Level II BASIC. If the program at the top of page 21 is attempted without modification in Level II, the display will show:

```
?FC ERROR IN 30
READY
>_
```

AGAIN!
Illegal function call error

When SET(X,Y) is used in Level II, the X value must be *less than* 128 and *greater than* or *equal to* zero. The

Y value must be *less than* 48 and *greater than* or *equal to* zero.

- Page 22—Reference is made near the middle of the page to the HOW? error message of Level I. This message will be ?FC ERROR for Level II.

CHAPTER 4—GAMES AND ABSTRACT ART

The Random Rectangle Game on pages 31-35 has several changes necessary for Level II.

- Page 33—The following modifications are suggested.

```
230 PRINT @0,"                                ";
235 PRINT @0,"WHERE IS THE CENTER";
```

- Page 34—Here come many changes.

Part IV Final Results (completely re-written)

```
400  PRINT "YOU GUESSED IT IN";C;"GUESSES!"
410  IF (C=1)+(C=2) THEN PRINT "EXCELLENT***":
     GOTO 450
420  IF C<5 THEN PRINT "GOOD**": GOTO 450
430  IF C<9 THEN PRINT "NOT BAD*": GOTO 450
440  PRINT"YOU NEED PRACTICE!!!"
450  FOR W = 1 TO 1000
460  NEXT W
     Note: lines 470 and 480 deleted.
490  INPUT "DO YOU WANT TO PLAY AGAIN (Y
     OR N)";Q$
500  CLS
510  IF Q$ = "Y" THEN 100
520  INPUT "THANKS FOR PLAYING! ANYONE
     ELSE (Y OR N)";Q$
530  CLS
540  IF Q$ = "Y" THEN 100
550  END
```

Note: lines 470 and 480 deleted.

- Page 40—Additional Suggestions for Practice #2. PRINT AT should be PRINT @.

CHAPTER 5—WHAT'S BEHIND BARS

- Page 45—Once again we see the PRINT AT statements in lines 400-440. All should be changed to PRINT @ otherwise the lines are correct.
- Page 46—Again, change all PRINT AT to PRINT @ in lines 540-610. Each of these lines should also end with a semi-colon.

```
540 PRINT @975,"YEARS";
550 PRINT @798,"1";
560 PRINT @670,"2";
570 PRINT @542,"3";
580 PRINT @414,"4";
590 PRINT @132,"SALES IN MILLIONS FOR
    THE";
600 PRINT @197,"FIRST FIVE YEARS OF THE";
610 PRINT @261,"DELTA COMPANY'S
    EXISTENCE";
```

- Page 47—Change the PRINT AT in line 50 to PRINT @386,"1977"
- Page 48—Line 10 in Try Number Two should be changed to: 10 PRINT @386,"1977"

  Also references in center of page should be PRINT @. Line 50 suggested near the bottom of the page for a revision to Try Number One should be: 50 PRINT @386,"1977";
- Page 49—Lines 10 through 18 only need PRINT AT changed to PRINT @.

CHAPTER SIX—STRAIGHT LINES AT ODD ANGLES

- Page 59—Lines 10 through 28 need PRINT AT changed to PRINT @.
- Page 65—Exercise 15, line 26—change PRINT AT to PRINT @.

  Line 70—change PRINT AT to PRINT @. Line 72 should be:

```
72 PRINT @0,"                                    "
```

CHAPTER 7—SITTING DUCKS

- Page 68—In the discussion of the POINT statement, Level II is different than Level I. Level II will print a −1 if the point (25,14) is lighted when the PRINT POINT (25,14) is executed. The form of the point test in the sample line: IF POINT(X,Y) = 1 THEN 600 would be changed to: IF POINT(X,Y) THEN ¢)) when Level II is used.

- Page 70—Change line 250 to PRINT @ instead of PRINT AT—otherwise it is OK.

- Page 71—A significant change is needed for lines 400-410. They are shown here.

```
400 PRINT @0,"WHICH GUN DO YOU WISH TO
    FIRE (1-5)";: INPUT S
402 SET(X-2,4): SET(X-1,4): SET(X,4): SET(X,3)
404 SET(M-2,4): SET(M-1,4): SET(M,4): SET(M,3)
410 PRINT @0,"                                ";
```

  Also change line 530 to: 530 IF POINT (P,Y-2) THEN 600. And line 610 to: 610 PRINT @Z,"ZAP"

- Page 72—The Moving Target Subroutine must be entirely changed since Level II does not allow wrapping around from X position 127 to X position 0.

```
1000 IF X=127 GOTO 1060
1001 IF M=127 GOTO 1080
1005 SET(X+1,3): RESET(X,3)
1010 SET(M+1,3): RESET(M,3)
1020 SET(X+1,4): RESET(X-2,4)
1030 SET(M+1,4): RESET(M-2,4)
1040 X = X+1: M = M+1
1050 RETURN
1060 RESET(X,3): RESET(X,4): RESET(X-1,4):
     RESET(X-2,4): X=4
1070 SET(X+2,3): SET(X+2,4): SET (X+1,4):
     SET(X,4): GOTO 1005
1080 RESET(M,3): RESET(M,4): RESET(M-1,4):
     RESET(M-2,4): M=4
1090 SET(M+2,3): SET(M+2,4): SET(M+1,4):
     SET(M,4): GOTO 1005
```

CHAPTER 8—BENDING A STRAIGHT LINE

- Page 78—Since Level II provides for exponentiation, lines 5 and 30 of the Curve Drawing Program could be changed (although they will work as is).

```
 5  INPUT "WHAT VALUE FOR A IN Y = A*X12
30  Y = INT(47 - A*A12)
```

Also for a cleaner input you might add:

```
75  PRINT @53,"     ";
    (5 spaces)
```

- Page 79—A reference to the right of the X,Y table is made to pressing the ↑ key to freeze the computer. In Level II press the SHIFT and @ keys together to freeze the display. Pressing any key will start it again.
- Page 83—Change the PRINT AT in line 5 of the Bent Line Program I to PRINT @. Do the same for line 110 of Bent Line Program II.
- Page 84—Here are some proposed changes in addition to changing the PRINT AT to PRINT @ in lines 300 and 310.

```
300  PRINT @0,"WHAT LINE DO YOU WANT
     ERASED(1 TO";Q;")";
305  A$ = INKEY$: IF A$=" " THEN 305 ELSE A
     = ASC(A$) - 48
310  PRINT @0,"                                    ";
```

CHAPTER 9—THE ETERNAL TRIANGLE

- Page 95—Change lines 240 through 270.

```
240  PRINT @412,"B";
250  PRINT @604,"C"
260  PRINT @612,"A"
270  PRINT @730,"RIGHT TRIANGLE ABC"
```

- page 101—Change PRINT AT in line 240 Exercise 21 answers to PRINT @.

CHAPTER 11—FAR OUT IDEAS

- Page 123—The following lines of the Cat and Dog Program need to be changed.

```
110 A$ = "CAT"
120 B$ = "DOG"
130 A = 8
140 PRINT @A,A$
170 PRINT @A,"  "
190 PRINT @A-16,"  "  (three spaces)
200 PRINT @A,A$
210 PRINT @A-8, B$,A$
```

For the program which begins on the bottom of the page:

delete line 40

- Page 124—Change the following.

```
60 PRINT @320,A$
   delete line 70
   change lines 80 and 90 to PRINT @
95 PRINT @448,"                    ";
```

The GOSUBS can be changed to:

```
120 ON Q GOSUB 300,500,700,900
130 IF Q = 2 THEN 280 ELSE 250
delete lines 140, 160, and 180
```

Change: lines 250, 280, 310 and 320 from PRINT AT to PRINT @.
Line 350—change to: PRINT @263," "
Line 390—change to: PRINT @204," "
Line 430—change to: PRINT @145," "
(4 spaces)

Change lines 360, 400, and 440 from PRINT AT to PRINT @. Line 440 has another change which makes it look like this:

```
440 PRINT @90,"IS EATING THEIR FOOD.  "
    (5 spaces)
```

- Page 125—Lots of lines need to change only PRINT AT to PRINT @. They are 510, 520, 560, 600, 710, 720, 760, 800, 910, 920, 960, and 1000.

  Also, several lines that say PRINT AT XXX,"" should be changed to PRINT @XXX," " with four spaces between the quotes. XXX stands for whatever position is called for. These lines are: 550, 590, 630, 750, 790, 830, 950, and 990.

  The following two lines should be changed as shown.

```
640 PRINT @90,"ARE EATING THEIR FOOD.      "
    (4 spaces)
840 PRINT @90,"HAVE EATING THEIR FOOD.     "
    (3 spaces)
```

- Page 126—At the top of the page change two lines.

```
1030 PRINT @154,"      " (4 spaces)
1040 PRINT @90,"WILL EATING THEIR FOOD.  "
     (3 spaces)
```

- Page 130—Several changes.

  Line 160 and line 170—change PRINT AT to PRINT @ only. Lines 400, 430, and 600 move the quotation marks apart as:

```
400 PRINT @0,"                                ";
```

  Line 410—change PRINT AT to PRINT @
  Line 440—change PRINT AT to PRINT @
  Line 610—change PRINT AT to PRINT @

- Page 131—Just a few more.

  Line 630—change to PRINT @0,"   ";
  Line 640—change to: 640 PRINT @33,"SHOT #";D+1;"   "
  Line 770—change as shown below.

```
770 PRINT @0,"OUT OF AMMO!! PLAYER WITH
    FEWEST TARGETS LEFT WINS!!!    "
```

About the Author...

Don Inman, who graduated from University of Northern Iowa with Bachelor's Degree in Mathematics, is a long-time computer aficionado. When he's not working busily at the Editorial desk for CALCULATORS AND COMPUTERS Magazine, Don is usually tied up in a computer text or how-to book project. Don also presently has two more TRS-80 books on tap for dilithium Press.

About the Book...

For some time now, graphic displays on microcomputers have been out of the reach of hobbyists because of their complexity and high cost. In this book, author Don Inman will show you how, with a minimal knowledge of the BASIC computer language and your TRS-80 computer, you can create graphic displays that only a few years ago were the exclusive turf of the big computer owners. The book begins with the basics and works from line drawings through geometrics and right on up to moving figure animation and more advanced operations. A great handbook on computer graphics for microcomputer owners at all levels of experience.

Also from dilithium Press...

COUNTDOWN: Skydiver, Rocket and Satellite Motion on Programmable Calculators
Robert Eisberg and Wendell Hyde
ISBN 0-918398-26-6

QUIKTRAN: Quick FORTRAN Programming for Micros, Minis and Mainframes
C. Kevin McCabe
ISBN 0-918398-24-X

PEANUT BUTTER AND JELLY GUIDE TO COMPUTERS
Jerry Willis
ISBN 0-918398-13-4

YOUR HOME COMPUTER
James White
ISBN 0-918138-05-1

HOME COMPUTERS: A BEGINNER'S GLOSSARY AND GUIDE
Merl Miller and Charles Sippl
ISBN 0-918398-02-9

HOME COMPUTERS: 210 QUESTIONS & ANSWERS, VOLUME 1: HARDWARE
Rich Didday
ISBN 0-918398-00-2

HOME COMPUTERS: 210 QUESTIONS & ANSWERS, VOLUME 2: SOFTWARE
Rich Didday
ISBN 0-918398-01-0

UNDERSTANDING COMPUTERS
Paul Chirlian
ISBN 0-918398-15-0

32 BASIC PROGRAMS FOR THE PET COMPUTER
Tom Rugg and Phil Feldman
ISBN 0-918398-25-8

BEGINNING BASIC
Paul Chirlian
ISBN 0-918398-06-1

dilithium Press
P.O. Box 606
Beaverton, OR 97075

ISBN: 0-918398-18-5