micro COMPUTER books

David D. Thornburg

Computer Art and Animation

A User's Guide to

Atari Logo

Computer Art and Animation

A User's Guide to Atari Logo

David D. Thornburg

ADDISON-WESLEY PUBLISHING COMPANY
Reading, Massachusetts • Menlo Park, California
London • Amsterdam • Don Mills, Ontario • Sydney

This book is in the **Addison-Wesley Microcomputer Books** Popular Series

Marshall Henrichs, *Cover Design*
Ellen Klempner, *Interior Design*

Library of Congress Cataloging in Publication Data

Thornburg, David D.
Computer art and animation.

Includes index.
Summary: Explains how to use a computer to create graphic designs and animated sequences.
1. Computer graphics—Juvenile literature. 2. LOGO (Computer program language)—Juvenile literature. 3. Atari computer—Programming—Juvenile literature. [1. Computer graphics. 2. LOGO (Computer program language) 3. Atari computer—Programming. 4. Programming (Computers)] I. Title.
T385.T4958 1984 001.64'43 83-25771

ISBN 0-201-06515-0

ISBN 0-201-06515-0
ABCDEFGHIJ-HA-8987654

For Kathy . . .

with the hope that computer graphics will be just one more tool with which you express your ideas

Preface

This book about computer graphics was written for two types of people—artists who are interested in computers and computer enthusiasts who are interested in art. It is possible that you fall in neither of these categories, in which case this book may stimulate your interest in both areas.

Many artists feel that computers are outside their range of comprehension, and many computerists feel the same way about art. If you fit in either of these groupings, perhaps the rest of this preface will help you toward another viewpoint.

Artists Anxious about Computers

I've met many very creative artists who freeze up when they hear the word *computer*. The idea that a faceless, emotionless symbol-manipulation tool could be used for creative expression is met with disbelief. Of course, some people think that hammers and chisels are pretty cold and faceless, too, but that doesn't keep hammers and chisels from being useful tools for sculptors.

The point is that the computer is a tool, just as a paint brush or a silk screen frame is a tool—no more and no less. Denying its utility to the artist is as senseless as treating it as a "thinking machine." The recent interest in this expressive medium is traceable to the fact that the cost of computer graphics systems has dropped to the point that they are showing up in hundreds of thousands of homes. The computer has become an affordable expressive tool that can be used effectively by any artist who is willing to invest a little time in learning how to use it.

You may believe that computers are "number-crunching" machines that require the constant attention of people with exceptional technical skills. Although some computers fit this description, it doesn't mean that *all* computers are primarily mathematician's tools or that *you* have to be a wizard to run one.

In fact, the inexpensive Atari computer systems that use the Logo language are marvelously simple tools to use. Furthermore, you can use any of them to create graphic designs of exceptional beauty. All that is required on your part is that you be able to press keys on a keyboard and that you acquire the vocabulary of the Logo language. This book will provide the framework to allow you to learn this language easily and naturally.

Once you have mastered the computer and its Logo language, you will have opened the door to a new creative medium—computer graphics.

There is one characteristic of computer graphics you should know about before we start—it is different. It is different from watercolors, oils, pastels, block printing, weaving, pottery, and every other art form we have ever had. Because it is different, computer graphics will never replace any of these other art forms. Because it is different, computer graphics will let you express ideas in totally new ways—ways that are as far removed from your present art form as pen-and-ink drawing is from marble carving.

So don't think I'm going to ask that you give up any expressive tool in your kit—I'm not. What I hope will happen is that you will find the computer to be useful in expressing ideas that aren't readily expressed by the methods currently at your disposal.

Computerists Anxious about Art

I've met many people who feel that they can't express themselves artistically. That's unfortunate, because humans have expressed themselves artistically since our ancestors found that images could be scratched on a cave wall with a rock. Of course, not everyone *wants* to be an artist, and that's fine. Not everyone wants to be a brain surgeon, either. The problem is that many people who don't express themselves artistically think that they *can't* express themselves in this way. There is overwhelming evidence that this is not true.

Take a pen and a sheet of paper and draw a line on it. That wasn't hard, was it? Is the result "art"? Maybe not, but if you drew the "right" lines, it would be. How does one draw the right lines? Regardless of stylistic differences, there is one thing that artists have in common. They not only look at the world around them, they *see* the world around them. The key to good artwork, then, is often good *seeing*.

The mind's eye—the imagination—is a marvelous tool for seeing. Can you imagine? Can you imagine an ice cream cone the height of the Empire State Building? Can you imagine the taste of the color green? What is the sound of red? How about the color of love?

Once you start imagining and seeing, you are ready to start expressing yourself creatively. If you are new to art, try many tools—crayons, paints, clay, pens, whatever you have available—even computers. After all, computers are useful artistic tools, and you have something to express.

Acknowledgments

Books don't appear, unassisted, on an author's desk; they are influenced by the support of many people. This book is no exception. The long-time friendship and association of Bob Kahn at Atari has meant a great deal to me. His perseverance—and that of others at Atari and at LCSI—not only has made Atari Logo a reality but also has made it one of the finest implementations of the language ever to appear on inexpensive personal computers.

The freedom to create is the most powerful freedom of all. For their tolerance, acceptance, support, and encouragement in this task, I want to thank all those who helped me across the border from technology to art.

Finally, there is one for whom my thanks is greatest. May I always be helped to see the world through the eyes of love.

Mountain View, California D.T.
November 1983

Contents

I. Introduction

Computer Graphics—Here, There, and Everywhere

Computer-generated graphics are all around us. Businesses, long accustomed to typewritten documents, are now discovering that computer-generated plots of business data can be comprehended faster than data presented in a table. Educators, long-time fans of pictorial information, are discovering that computer graphics can enhance the teaching of subjects as diverse as logic and chemistry. And, of course, the entertainment applications of computer graphics are well known. To pick just one example, computer-driven video games illuminate millions of homes every day. The appeal of these games comes, in large part, from their colorful graphic images. Whether jumping frogs across a pond, jumping a man over a barrel, or smashing space bugs, the colorful animation holds the players' interest time and time again. Video game players are controlling their own cartoons!

Of course, games are far from the only entertainment application of computer graphics. The motion picture industry is now spending millions of dollars on massive computer systems to bring high-quality computer-generated images to the silver screen. In such films as *Star Wars, TRON,* and *Star Trek II,* the computer has established itself as a powerful ally for those who create video magic.

Beyond commercial applications, computer graphics systems have been used effectively as an expressive tool by many artists fortunate enough to have access to them.

Computers in the Home

Far from being the tools of a privileged few, computer graphics systems are available today for the cost of a clothes dryer. The personal computer industry grew from its modest beginnings a few short years ago to the point at which millions of computers are being purchased every year. The appearance of computers in homes means that millions of people now have access to technology that was once only found in industrial and academic environments. A graphic artist can now create computer-generated artwork in the privacy of a home or studio, instead of having to travel to a remote laboratory to gain access to this technology.

There are two major ways of generating computer graphics on a home computer system. One way involves the use of a *computer graphics program* that lets you use a graphics tablet or a joystick to "sketch" images on the screen. Using special commands in such a program, you might be able to fill outlines with color, rotate images, replicate parts of an image in other areas of the screen, and do many other powerful things. The only problem with these programs is that they do not provide access to additional features that you might like to incorporate in your artwork. Because these programs are generated by other people, you must be content with their concept of the graphic tools you need. Nonetheless, many of these programs are excellent and appropriate for artists who feel the need to create artwork by moving a pen over a surface.

The second way to create computer graphics is to work directly with a computer language that is richly endowed with graphics commands. Certain computer languages have the property of allowing you to define your own procedures, which may be recalled by simply typing the procedure's name. This approach to computer graphics, *graphics programming*, brings you in closer touch with the raw power of the computer. It even allows you, should you wish, to create programs that do the things done by existing graphics programs, with the important difference that *you* are the one who determines which features to incorporate.

Contrary to popular belief, computer languages are not hard to learn. With the appropriate language, you can be creating interesting graphics programs in the first ten minutes of exposure to the system. One of the most appropriate languages for the creation of computer graphics is called Logo. The feature of Logo that makes it so powerful is called "turtle graphics."

The "turtle" is an imaginary creature residing on the display screen whose sole function is to obey your commands. By instructing the turtle to move in its present direction or to turn by some angle, you can create any image you desire on the screen. The turtle can be instructed to leave a trace of its path in any of several colors, to erase a line, and to obey a predefined series of commands called a *procedure.*

Is the turtle easy to use? If you understand such words as FORWARD, BACK, RIGHT, and LEFT, you are well on the way to getting the turtle to create pictures on your television screen.

Using Atari Logo

This book will focus on the powerful graphics environment provided by the Atari personal computers, using the Atari Logo language cartridge. To make use of the material in this book, you will need the following equipment:

- Any Atari personal computer system
- Atari Logo cartridge
- Color (or black and white) television set or monitor
- Tape recorder or disk drive interface and disk drive (for recording your artwork)

Depending on your needs and desires, you may also wish to record your artwork on film or videotape. In this case, you will need a camera with a tripod or a video cassette recorder. Fortunately, the cost of a video cassette recorder has fallen to the point where it is now a reasonably priced accessory for your computer system.

What We Will Do

The remainder of this book will deal with the following topics:

- Becoming familiar with the equipment
- Turtle graphics and the creation of static images

- Projects in computer graphics
- The creation of animated shapes
- Designing animated sequences
- Projects in animation
- Recording your artwork on film or on a video cassette recorder

We will take things step by step, and before long you will have mastered computer graphics with Atari Logo. So set aside some time and prepare to enter the world of graphics programming. It is likely that you will find it fascinating.

II. Getting Started with Logo

Locating the Computer System

It is important that you find a good place to locate your computer system. Since you will be using it for creative applications, you should pick a place free from distractions. If you have a studio, find a quiet corner where the equipment won't be disturbed. At home, set up in a study or spare room away from heavy traffic flow. Your locations should be as dust-free as possible and should be indirectly lighted. Find a tabletop that is large enough to comfortably hold the computer system and any related equipment you may be using (such as a video cassette recorder). Be sure you can reach the keyboard easily. Your creative ideas won't flow well if your arms have to strain to reach the keys! The television set or monitor should be placed where the screen may be seen easily without any strain on your neck or eyes. Be sure that no room lights reflect off the screen.

Although your computer space needn't be dark, it shouldn't be so bright that the screen colors appear washed out. Most important, the computer should be in a place you like—a place that is conducive to the free flow of ideas.

Making the Right Connections

I wish we could start making pictures right away, but in this regard computers are different from pencils and paper. Computers have to be set up and turned on. Fortunately, unless you move your equipment around a lot, your system will only have to be set up once. The manuals that came with your Atari computer system show you how to set up all the equipment, so those instructions won't be repeated here. You should refer to the manual and turn the equipment on in the following sequence:

1. Turn on the disk drive (unless you are going to use a tape recorder to store your graphic procedures).
2. If you are using a disk drive, insert the Atari Master Diskette in the disk drive and close the door. You may also use a data diskette if it contains the disk operating system files.
3. Insert the Logo cartridge and turn on the computer.

Of course, you should also turn on your TV set, or you won't be able to see your pictures!

Starting Logo for the First Time

If you have used Atari Logo before, feel free to skip this section. What we will do here is show what happens—and what can go wrong—when starting the Logo system.

First, take out the Logo cartridge and turn on the power to your computer by itself. Do *not* turn on your disk drive power first (this time). Once the TV has warmed up, you may see the following display:

```
ATARI COMPUTER—MEMO PAD
```

This message lets you know that the Logo cartridge is not plugged in. Turn the computer off, insert the cartridge, and follow the startup procedure again. This time you should see the following message:

```
(C) 1983 LCSI ALL RIGHTS RESERVED
WELCOME TO ATARI LOGO
?_
```

This is much better.

If you look at the upper left corner of the display, you will see a solid white square. This is called the *cursor.* Its function is to let you know where a letter will appear when you type something on the keyboard. The question mark is a *prompt*. This symbol lets you know that the computer is waiting for you to give it an instruction.

You should take a few minutes to get familiar with the keyboard, since it is the principal tool through which you will convey your desires to the computer. Other ways include yelling at the system (not very effective), turning off the machine (effective, but final), and using joysticks or the KoalaPad graphics tablet (to be covered in a later chapter).

At first glance, the keyboard resembles that of a typewriter, at least so far as the placement of the letters and numbers is concerned. Some keys have two symbols on them. The comma key also has a left square bracket ([), and the period key has a right square bracket (]). These brackets are very important in Logo. To type a bracket, press the key marked SHIFT and hold it down while pressing the appropriate key.

Before using the keyboard, find the key marked RETURN. This key is pressed at the end of every line of instructions we give the computer. If we see an error before pressing this key, we can back up and fix it. Once RETURN is pressed, the computer will try to do the things you have entered in your commands. Because RETURN refers to a set of commands, this key should not be pressed at the end of each line of characters on the screen, unless the end of the line corresponds to the end of your instructions. If you are typing a long line of instructions, the computer will automatically continue displaying the instructions on the next line if you run out of room. You will see a right-facing arrow on the right side of the screen when you are entering more characters than can be displayed on one line. Only press RETURN when you are all done.

As an example, try typing the following:

```
THIS IS A TEST OF THE LOGO SYSTEM WITH LONG LINES
```

If you make a mistake while typing, just press the key marked DELETE/BS until you have erased the incorrect letter. Continue retyping to finish the line. You will see that as you typed

WITH the WI appeared on one line and the TH appeared automatically on the next line. You do not need to press RETURN until you are finished with your line of commands. If you press RETURN now, you will find that Logo doesn't know what to do with your sentence—but perhaps that isn't too surprising.

You should note that Logo only uses uppercase characters for its commands. You can display both uppercase and lowercase letters on the screen by pressing the CAPS/LOWR key. To get uppercase letters when this key has been pressed, just use the SHIFT key as you would on a typewriter. To restore the keyboard to display of uppercase letters only, hold down the SHIFT key and press the CAPS/LOWR key again.

A Brief Introduction to Logo

Now that we have come this far, let's get better acquainted with Logo. Type the following work and press the RETURN key:

```
HELLO
```

When you pressed RETURN, the computer responded with

```
I DON'T KNOW HOW TO HELLO
```

As you can see, our greeting was not too well received. Whenever you see the message I DON'T KNOW HOW TO, it means that you have used a word that is not in Logo's vocabulary. Logo starts out with a vocabulary of words that lets it do many useful things. Even better than that, however, Logo lets you create definitions for words of your own choosing (we'll see how later).

Perhaps this system is pretty friendly after all. You may not realize it, but your Atari computer will do another favor for you if you leave your computer on and unattended for a long time (several minutes). The screen will start to undergo automatic color changes to make sure your display phosphors don't get burned out. To stop this flashing display, simply press any key.

Introducing the Turtle

Of all the features of Logo, the turtle is central to computer graphics. As mentioned earlier, the turtle can be thought of as an object to which we send messages. These messages come either directly from the keyboard or as a result of letting Logo use a procedure we have defined.

The turtle draws its pictures on a portion of our display screen. Since the screen is entirely available for the display of text when Logo is started, we need to shift from the text display mode to the graphics display mode. To make this transition, we can send a message to the turtle. Type

```
CS
```

(which stands for *Clear Screen*) and press the RETURN key. Remember, if you make a mistake while typing, press the DELETE/BS key enough times to back up over the mistake; *then* retype the line.

As soon as you have entered this command you will see a small white turtle in the middle of the screen and the cursor at the left edge of the screen, roughly four-fifths of the way down. We have entered a mode in which we can see both pictures drawn by the turtle and a few lines of commands as we enter them.

The location of the turtle shows that it is in the center of the screen, and its direction shows that it is pointing toward the top of the screen. As soon as we entered the command CS, Logo was instructed to pass all relevant messages to the turtle for execution.

What kinds of messages can we send to the turtle? Let's find out!

III. Making Drawings with the Turtle

We are now at a point where we can start to create pictures. To do this, we need to learn the words in Logo that make the turtle draw lines on the display screen.

Some Basic Logo Commands

Before we can get any drawings on the screen, we must clear the display screen and make some space in which our turtle can move. We do this by entering the command

```
CS
```

To turn the graphics mode off and return to a full text display, we use the command

```
TS
```

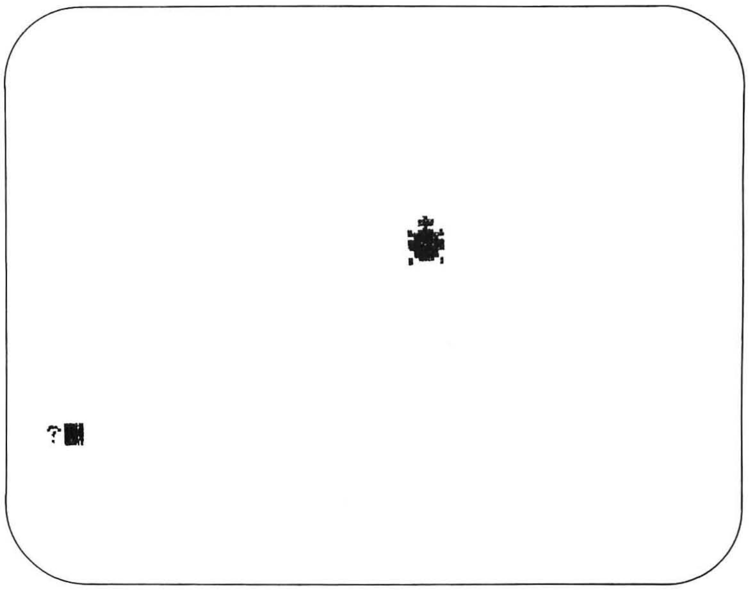
which lets us see the *Text Screen.* To see the turtle's full display area (but no text), we can type

```
FS
```

for *Full Screen.* To return to our original split screen (part graphics, part text), we can type

```
SS
```

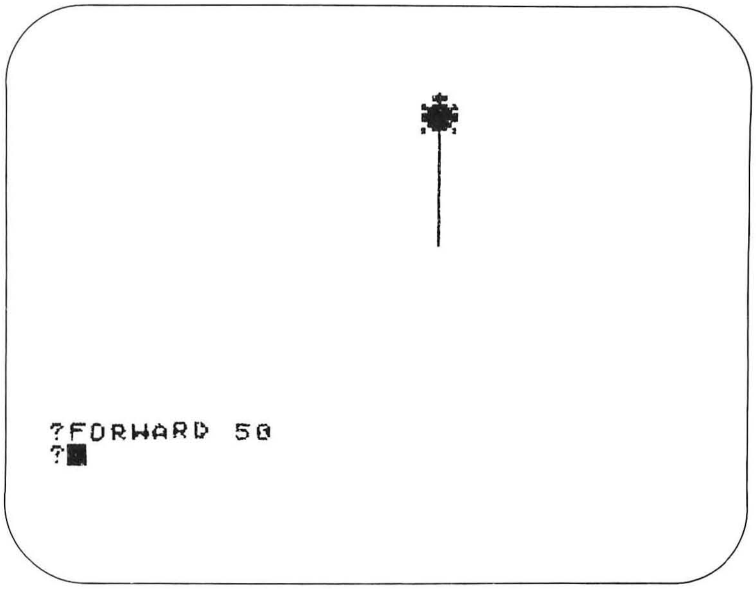
for *Split Screen.* When you type CS from either the text or the split screen mode, your screen is divided into a graphics and text window, and the turtle is placed in its "home" position and orientation.



Now let's find out how to move the turtle forward. We could try using the word FORWARD, but how much would the turtle move? This command must be followed by a specification of the number of screen units we want the turtle to move forward. Try entering

```
FORWARD 50
```

to see what happens.



```
?FORWARD 50
?
```

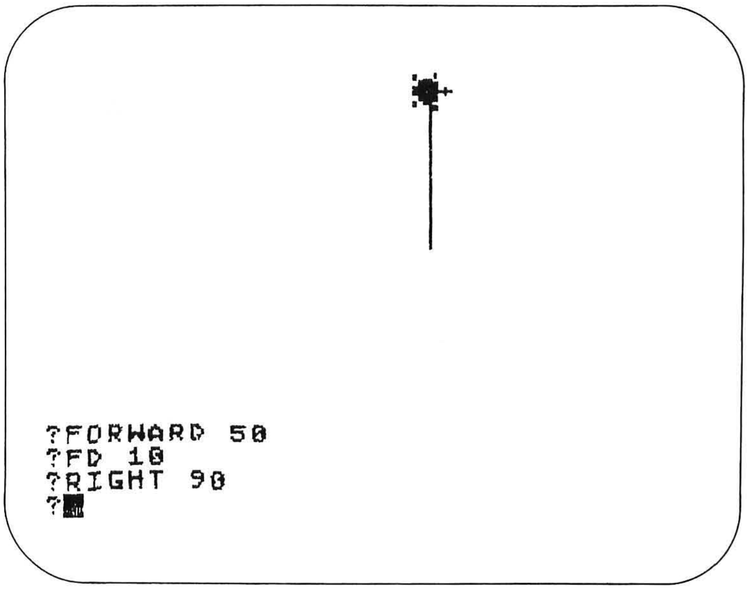
This command caused the turtle to move in its initial heading (toward the top of the screen) by 50 units, leaving a yellow trail behind. We can extend this line a little bit by giving the command

```
FD 10
```

The command FD is simply a shorthand form of FORWARD. There are short forms for many commonly used Logo words, and you will find that they save a lot of time and typing!

Unless we find some other commands soon, it is unlikely that the turtle will be very useful to us as a graphics tool. Fortunately, in addition to making the turtle move forward, we can also make the turtle change its orientation. To turn the turtle to the right by 90 degrees, for example, enter

```
RIGHT 90
```

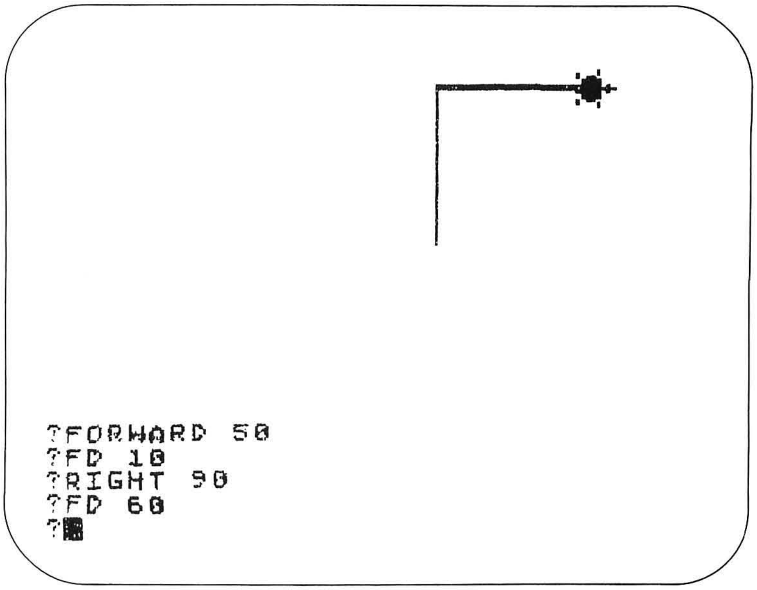


```
?FORWARD 50
?FD 10
?RIGHT 90
?
```

As you can see from the display, an advantage of having a visible turtle is that it is easy to tell in which direction it is pointing.

Let's see if we can use what we have drawn so far to create a square. Since our first line is now 60 units long, we should move forward by this amount again. Enter

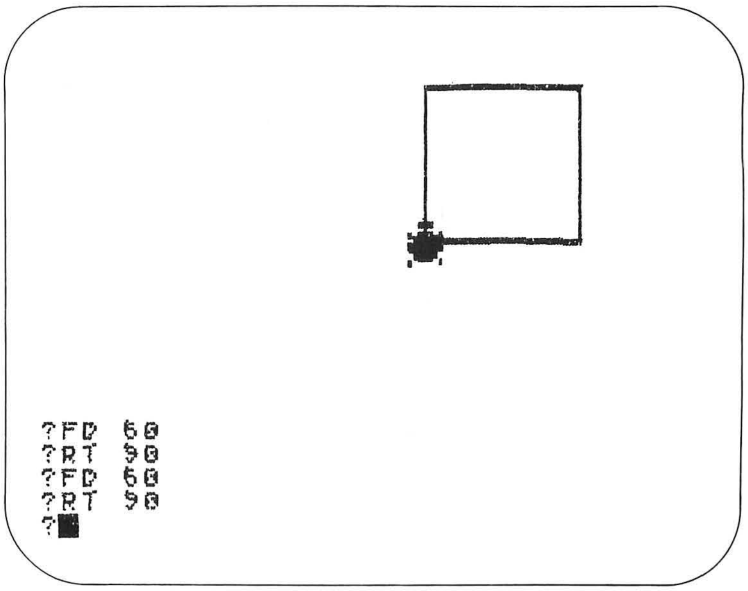
FD 60



```
?FORWARD 50
?FD 10
?RIGHT 90
?FD 60
?█
```

The shorthand form for RIGHT is RT, so the following commands should let us finish the square:

```
RT 90
FD 60
RT 90
FD 60
RT 90
```



```
?FD 60
?RT 90
?FD 60
?RT 90
?█
```

Notice, as you enter these commands, that the text you have already typed automatically moves up one line when your new text runs out of room at the bottom.

Why do you suppose we added the final RT 90 command, especially since the square appeared complete when we drew the last line? The final RT 90 command was needed to return the turtle to its exact starting position and orientation. As you gain more experience with turtle graphics, you will generally find it valuable to make sure that each figure ends up with the turtle back at its starting position and orientation. This is especially true if you make pictures using simple figures as general building blocks that are used over and over again.

Does the square we drew look square on the display? It is possible that it appears a bit rectangular. Every TV set and monitor seems to have a different scale for horizontal and vertical lines. If you want to make squares appear square, you should adjust the vertical size control (usually located at the back of the TV) until the figure looks correct to you. Alternatively, you can change the aspect ratio of the lines drawn on the screen with the .SETSCR (*SET SCReen*) command. Notice that this command starts with a period. Logo commands that start with periods are used infrequently and can change the properties of Logo if they aren't used carefully. Fortunately, .SETSCR is benign. When you first use Logo, the aspect ratio is set to 0.8 (the correct value for American television sets). If you draw a square that appears to be too wide, you might increase the aspect ratio a bit by entering

```
.SETSCR 0.9
```

and drawing another square to see if it looks better. After some trial and error, you will be able to set the aspect ratio perfectly for your TV set or monitor. Jot down the correct aspect ratio and enter it each time you start the computer. This value won't change unless you turn the computer power off.

So far, we have learned how to use two very basic commands: FORWARD and RIGHT. If you feel adventurous, you might want to examine the use of the commands BACK and LEFT (or BK and LT). You should be able to determine quickly that BACK and LEFT work in opposite ways from FORWARD and RIGHT.

Repeating Commands

It may seem that you have to do a lot of typing to make the turtle draw pictures. Fortunately, this is not the case. Logo has a command that makes it very easy to create figures that are generated from the repetition of commands over and over again (such as a square). This command consists of the word REPEAT, followed by the number of times the instructions are to be repeated, followed by a list of the instructions.

For example, the eight steps we used to create a square can be replaced with this single command:

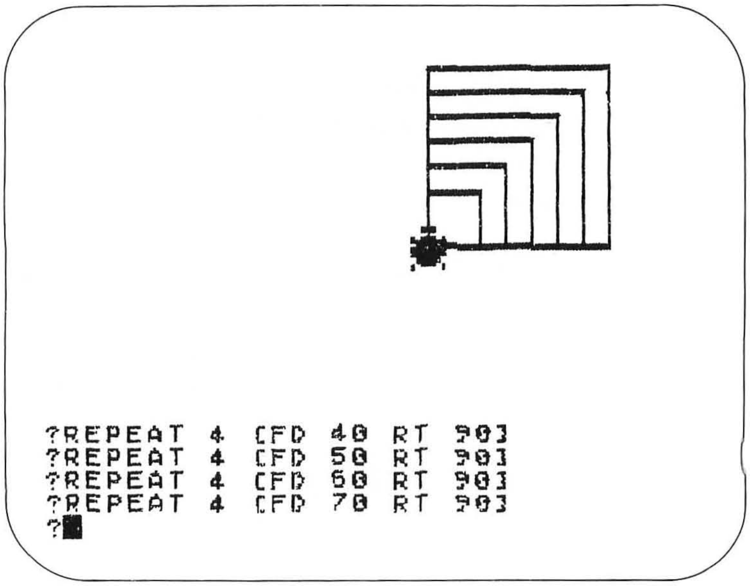
```
REPEAT 4 [FD 60 RT 90]
```

This command says, in essence, "Repeat, four times, the instructions 'move forward 60 units and turn right 90 degrees.' " To type this command, you need to use the square brackets ([ ]), which are found on the comma (,) and period (.) keys and are obtained by holding down the SHIFT key while typing the appropriate character.

Logo uses square brackets to enclose a *list*. Lists are collections of words that may be Logo commands or any of several other objects used by Logo. As we shall see later, the ability of Logo to work with lists is one of its strong points.

To try out our easy technique for drawing squares, enter the following:

```
CS
REPEAT 4 [FD 20 RT 90]
REPEAT 4 [FD 30 RT 90]
REPEAT 4 [FD 40 RT 90]
REPEAT 4 [FD 50 RT 90]
REPEAT 4 [FD 60 RT 90]
REPEAT 4 [FD 70 RT 90]
```



```
?REPEAT 4 [FD 40 RT 90]
?REPEAT 4 [FD 50 RT 90]
?REPEAT 4 [FD 60 RT 90]
?REPEAT 4 [FD 70 RT 90]
?
```

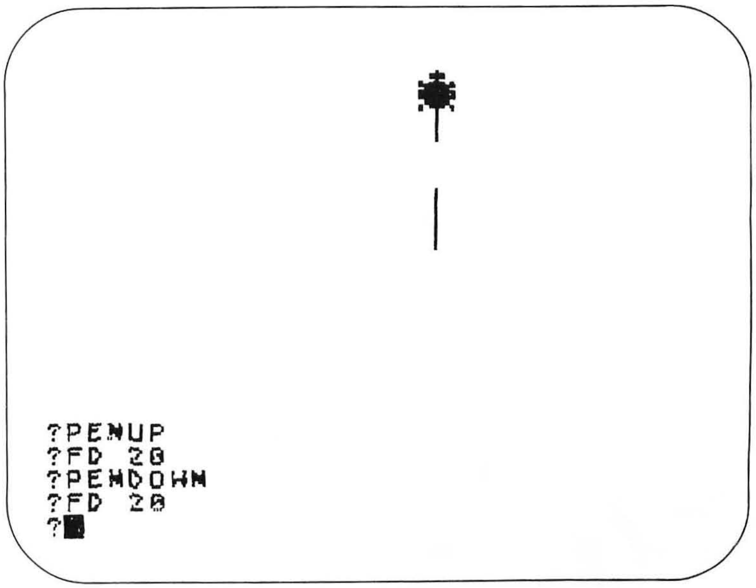
This set of nested squares was created with far fewer commands than we would have needed if we didn't have the REPEAT command at our disposal. We will see more creative uses of this powerful instruction as we progress.

Before generating any more pictures, we should learn more about things the turtle can do.

The Turtle's Pen

So far, we have seen that the turtle starts out with a yellow "pen" that draws lines as the turtle moves. There will be times when we will want the turtle to move without drawing a line. For such times, it will be important to have a command that "picks the pen up." To see how Logo accommodates us in this regard, enter

```
CS
FD 20
PENUP
FD 20
PENDOWN
FD 20
```



```
?PENUP
?FD 20
?PENDOWN
?FD 20
?
```

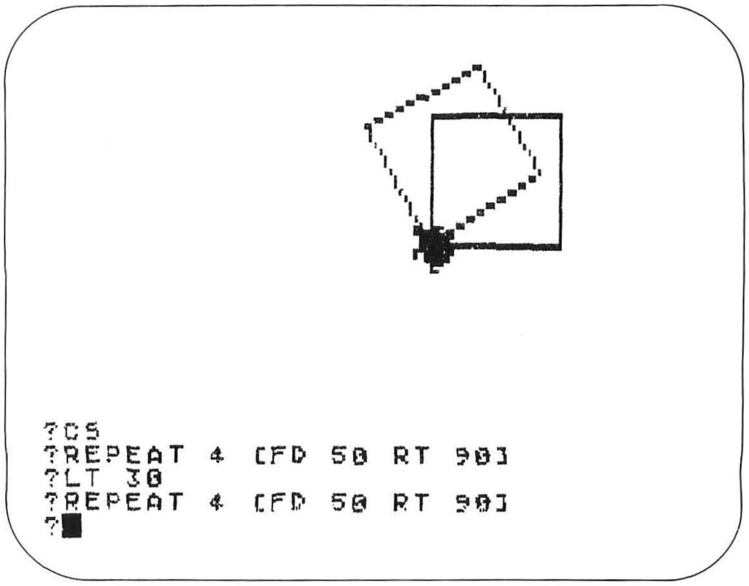
As you entered these commands, the turtle obeyed the instructions to lift the pen up and set it down again. The abbreviations for PENUP and PENDOWN are PU and PD, respectively.

Now that you know how to move the turtle and how to lift and lower the pen, you might want to know how to erase a line you have drawn by mistake. To erase a line, we need to have the turtle carry an eraser instead of a pen. Once again, we will use an example to show how this is done. Let's start by drawing a square:

```
CS
REPEAT 4 [FD 50 RT 90]
```

Next, let's add a second square, rotated to the left of the first one by 30 degrees:

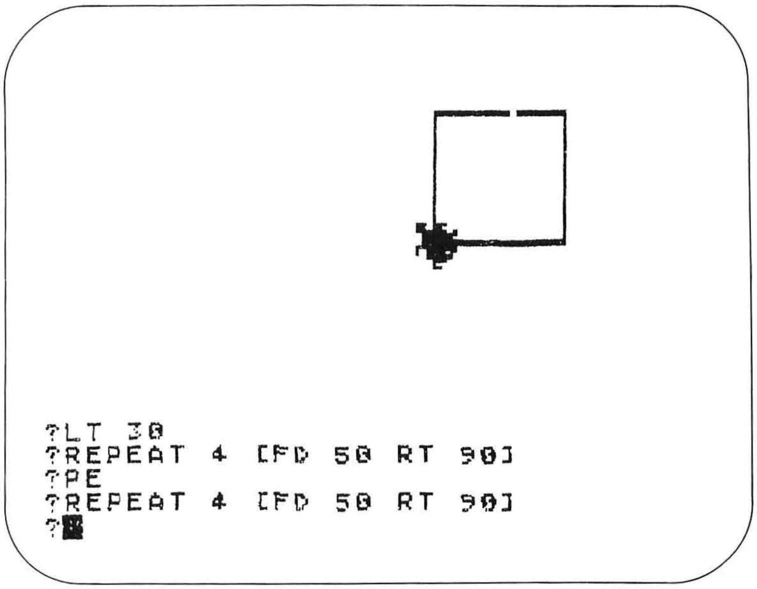
```
LT 30
REPEAT 4 [FD 50 RT 90]
```



```
?CS
?REPEAT 4 [FD 50 RT 90]
?LT 30
?REPEAT 4 [FD 50 RT 90]
?█
```

If we decide to erase this second square without disturbing the first one, we need only enter

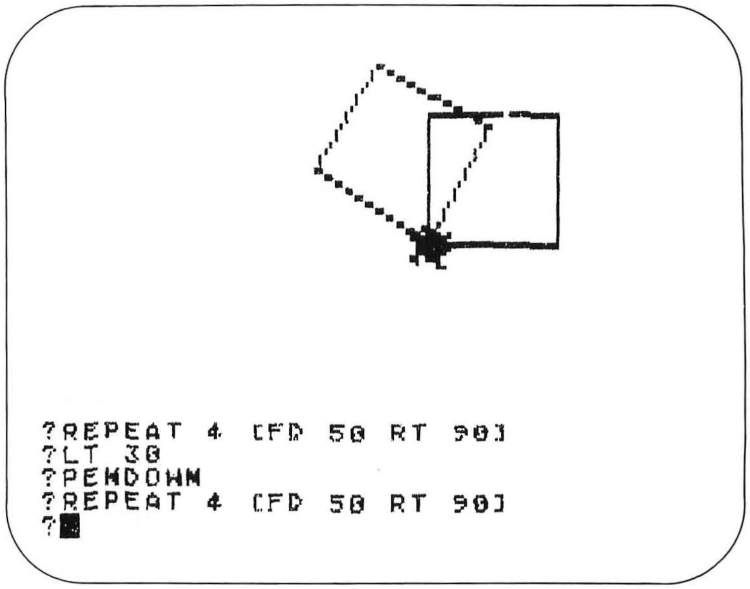
```
PE
REPEAT 4 [FD 50 RT 90]
```



```
?LT 30
?REPEAT 4 [FD 50 RT 90]
?PE
?REPEAT 4 [FD 50 RT 90]
?
```

To draw a new square rotated by 30 more degrees to the left, enter

```
LT 30
PENDOWN
REPEAT 4 [FD 50 RT 90]
```



```
?REPEAT 4 [FD 50 RT 90]
?LT 30
?PENDOWN
?REPEAT 4 [FD 50 RT 90]
?
```

The *Pen Erase* command is entered as PE. To continue drawing after erasing some lines, just enter PD for *Pen Down*.

In addition to having an eraser, the turtle also has 128 combinations of color (*hue*) and brightness (*luminance*) from which you can choose. Each hue and luminance level has a number associated with it, as shown in the following tables:

HUE	NUMBER
Gray	0
Gold	8
Orange	16
Red-orange	24
Pink	32
Purple	40
Purple-blue	48
Dark blue	56
Blue	64
Light blue	72
Turquoise	80
Green-blue	88
Green	96
Yellow-green	104
Orange-green	112
Light orange	120

LUMINANCE	NUMBER
Dark	0
	1
	2
	3
	4
	5
	6
Light	7

Atari Logo lets the turtle use any of several pens, each of which can have its color changed by the user. If you change the color of a pen with which you have already drawn some lines, all the lines you have previously drawn with that pen will be changed to the new color. In other words, you can change the color of lines after you have drawn them! To set the color of the pen you are using, enter the *SET Pen Color* command (entered as SETPC). The command

```
SETPC 0 85
```

will set the starting pen (pen 0) to a medium-brightness turquoise. It does this because the color setting 85 is equal to the value for turquoise (80) plus the medium luminance value (5). To choose a color, add the hue value (from the first table) to the luminance value (from the second table) and use the result in the SETPC command. The 0 in the SETPC command assigns the color change to pen 0—the pen carried by the turtle when it is given the CS command.

The color black is the darkest form of gray. Since gray has the hue value 0 and the darkest luminance also has the value 0, you might want to experiment to determine that the command

```
SETPC 0 0
```

will cause the turtle to draw black lines.

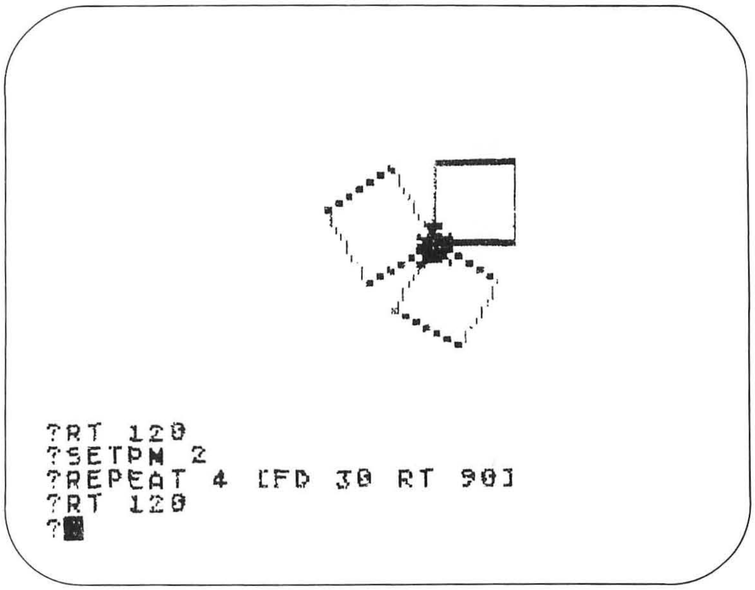
To have more than one color on the screen at a time, the turtle can use any of three pens. It starts out with pen 0 but can be assigned any pen with the SETPN (*SET PeN*) command. The command

```
SETPN 2
```

lets the turtle draw lines with pen 2. When Logo first starts out, each of the three pens is automatically assigned a color value. Pen 0 has color 15 (light gold), pen 1 has color 47 (light purple), and pen 2 has color 121 (orange). Remember, if you don't like these colors, you can change them to anything you want. This freedom to change colors so easily is a unique property of Atari Logo that you will probably make great use of!

To see the effect of different pen colors, enter

```
CS
SETPN 0
REPEAT 4 [FD 30 RT 90]
RT 120
SETPN 1
REPEAT 4 [FD 30 RT 90]
RT 120
SETPN 2
REPEAT 4 [FD 30 RT 90]
RT 120
```



```
?RT 120
?SETPN 2
?REPEAT 4 [FD 30 RT 90]
?RT 120
?
```

Now that you have pictures drawn with all three pens on the screen, experiment with SETPC to see the effect of color changes on your pictures.

Atari Logo not only lets you change the color of the turtle's pen, it also lets you change the background color of the screen. To do this, we use the SETBG (*SET BackGround*) command. This command, which uses the same numbers as are used for the pen colors, starts out with color 74 (light blue). For example, the command

```
SETBG 0
```

creates a black background, and the command

```
SETBG 74
```

restores the background to its original value (74).

Experiment with different combinations of background and pen colors. The Atari palette is quite rich and should be explored. One interesting visual effect you can create is to make a figure appear to vanish by changing its pen color to the same color as the background. To make the figure reappear, just change the pen color to another value!

The Turtle's Limits

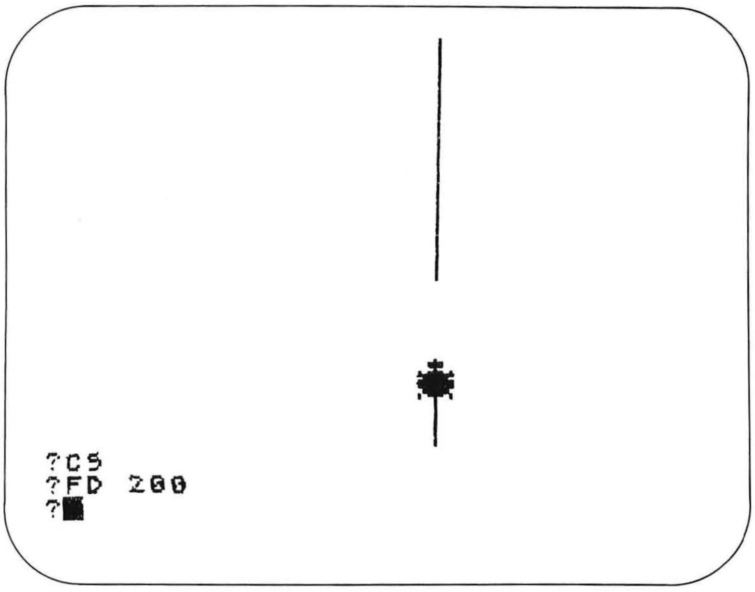
Now that we know as much about the turtle as we do, let's see how much room is available to us for drawing pictures. Enter

```
CS
```

and let's see what happens when we send the turtle on a long trip. If we enter

```
FD 200
```

we see a vertical line running off the top of the screen and reappearing at the bottom of the graphics window.



```
?CS
?FD 200
?█
```

What happened?

When we tried to run the turtle off the screen, it "wrapped around" to the other side. This wraparound feature guarantees that you will never lose the turtle. If you don't want to use this feature, it can be turned off with the command

```
WINDOW
```

and turned on again with the command

```
WRAP
```

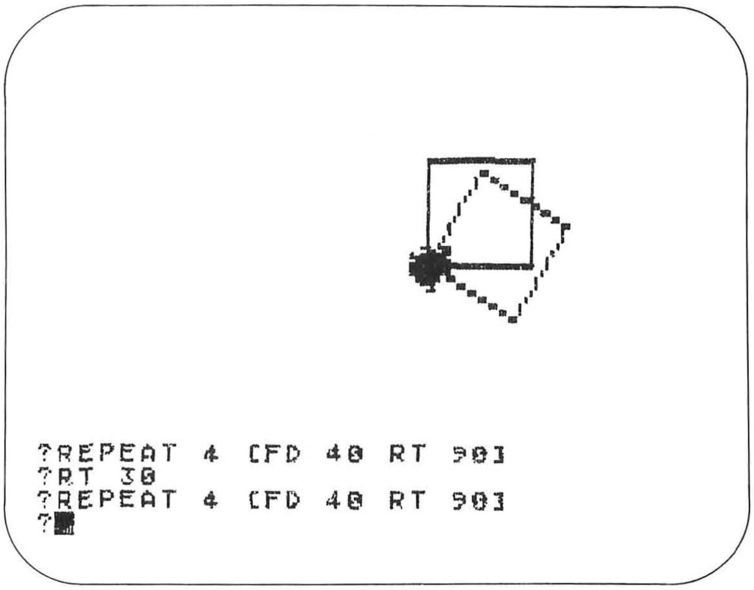
Devise an experiment to see how far you can go from the home position without wrapping around. If you measure the extent of the turtle's world, you will find it can move as much as 161 units to the right and 158 units to the left of the home position, as much as 120 units up from this position, and as much as 119 units below the home position. The vertical space available will depend on the value of the .SETSCR setting; these numbers apply when this setting is 0.8.

If we call movements to the right or left of the home position movements along the *x*-coordinate, and movements above and below the home position movements along the *y*-coordinate, we can see that any point on the screen can be specified by a pair of numbers that give the *x*- and *y*-coordinates of the point.

Relocating the Turtle

Logo lets us pick the turtle up and place it at any location on the screen. This is tremendously valuable if you want to start a picture somewhere other than at the center of the screen. Depending on the kind of motion you desire, you can use one of three relocation commands, SETX, SETY, and SETPOS. These commands set the location of the turtle anywhere on the screen. To see how they work, enter

```
CS
REPEAT 4 [FD 40 RT 90]
RT 30
REPEAT 4 [FD 40 RT 90]
```



```
?REPEAT 4 [FD 40 RT 90]
?RT 30
?REPEAT 4 [FD 40 RT 90]
?
```

These commands drew two squares, starting from the home position. The second square is tilted from the first by *30* degrees. Next, enter

```
PU
SETX -60
PD
```

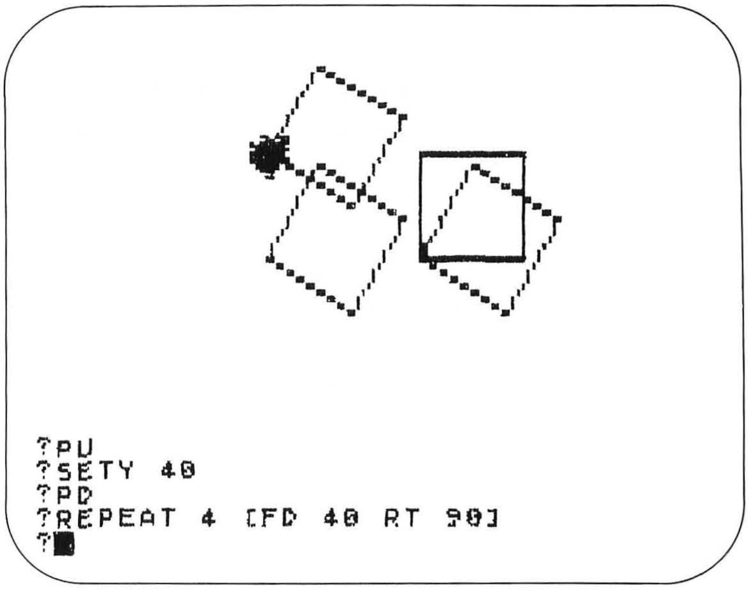
The minus sign (−) is just to the right of the P key. Notice that the turtle has moved to the left by 60 units. If you now enter

```
REPEAT 4 [FD 40 RT 90]
```

you will get a tilted square, starting at this new location. This shows that as the turtle is moved, its orientation is not changed in the slightest. Next, enter

```
PU
SETY 40
PD
REPEAT 4 [FD 40 RT 90]
```

The turtle moved directly up from its previous location by 40 units and drew another square.

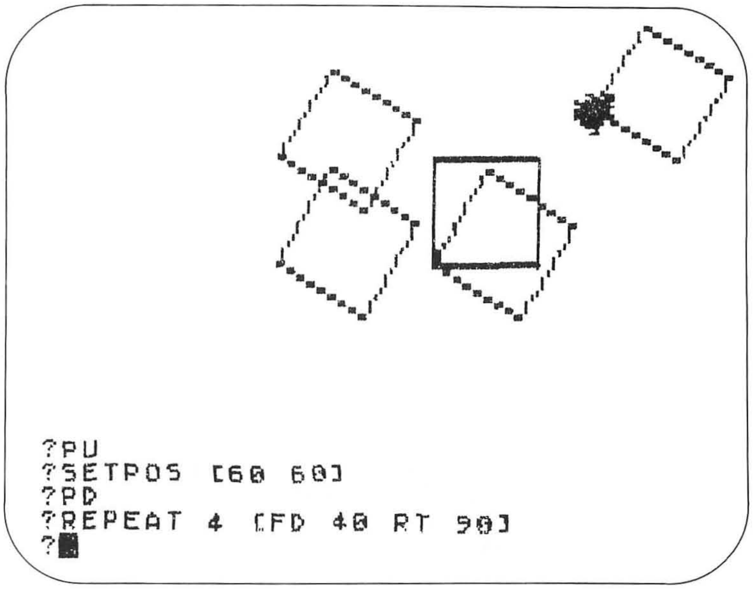


```
?PU
?SETY 40
?PD
?REPEAT 4 [FD 40 RT 90]
?
```

Now enter

```
PU
SETPOS [60 60]
PD
REPEAT 4 [FD 40 RT 90]
```

The turtle moved to a point 60 units up and to the right of its home position before drawing another tilted square. Notice that when you use SETPOS, the coordinate values form a list, which must be enclosed in square brackets.



```
?PU
?SETPOS [60 60]
?PD
?REPEAT 4 [FD 40 RT 90]
?
```

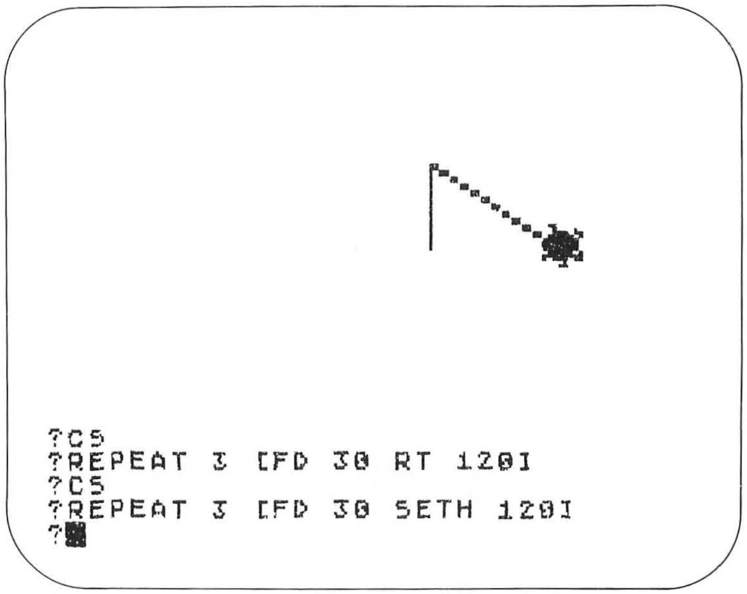
Finally, enter

SETPOS [60 −20]

Oops—this drew a line on the screen because we forgot to lift the pen first! This is a good reminder that we must always lift the pen before moving the turtle (and remember to set it down afterwards). Now enter

REPEAT 4 [FD 40 RT 90]

to complete our picture.


```
?CS
?REPEAT 3 [FD 30 RT 120]
?CS
?REPEAT 3 [FD 30 SETH 120]
?
```

This draws the first two sides of the triangle and then continues drawing the third side at an angle of 120 degrees. The commands RIGHT and LEFT turn the turtle by a specified amount from their previous heading. They are *relative* commands. The command SETH turns the turtle *to* an angle measured from 0 degrees (pointing straight up). It is an *absolute* command.

Before finishing this section, you should know that you can make the turtle invisible by typing HT (*Hide Turtle*). This is useful if you don't want the image of the turtle to be part of your artwork. To see the turtle again, just type ST (*Show Turtle*).

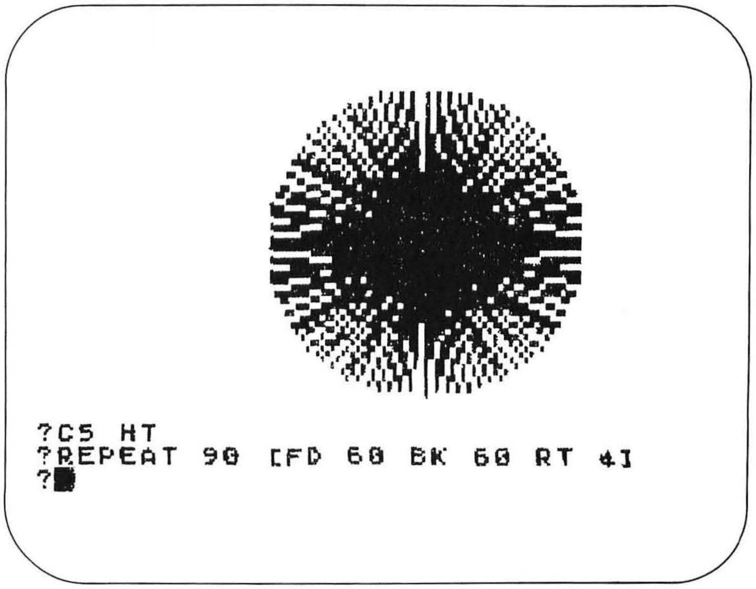
Some Art Projects

If you have worked straight through to this point, you deserve a rest. This chapter has given you many of the mechanical details of the Logo turtle, and it is now time to spend some effort using what you have learned. Before progressing to the next level, create some works of art to try out the skills you have obtained thus far.

As an example, if you enter

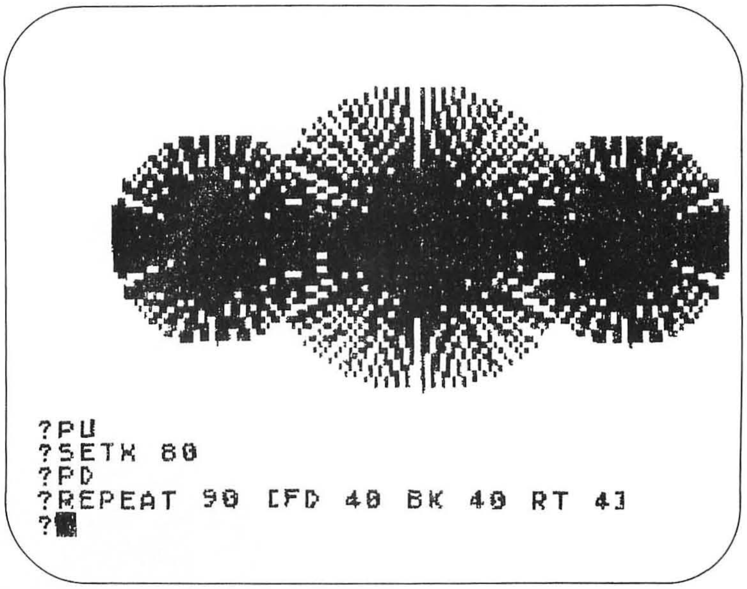
```
CS HT
REPEAT 90 [FD 60 BK 60 RT 4]
```

you will get this pleasant pattern.



```
?CS HT
?REPEAT 90 [FD 60 BK 60 RT 4]
?
```

How would you go about creating the following pattern?



```
?PU
?SETX 80
?PD
?REPEAT 90 [FD 40 BK 40 RT 4]
?
```

One way to make this picture is to start with the previous one and add the following commands:

```
PU
SETX -80
PD
REPEAT 90 [FD 40 BK 40 RT 4]
PU
SETX 80
PD
REPEAT 90 [FD 40 BK 40 RT 4]
```

Now you should experiment on your own. Try creating pictures that fit nicely on the screen. Then try creating pictures that use long lines (such as FD 300) to see the effects of wraparound. Set the turtle to some angle, such as 47 degrees (with the screen in wraparound mode), and send the turtle on a *very* long trip!

Play with pen colors. Play with background colors. Can you make the background flash from one color to the next?

As you experiment, think about the texture of the graphics medium:

- Is computer graphics smooth or grainy?
- Is it hard or soft?
- Is it crisp or fuzzy?
- Which colors do you like?
- Does it seem that the computer lets you paint with light itself?
- Does the graphics screen seem like an electric loom, with the patterns woven in its surface?

You might want to keep a journal of your art projects and jot down your feelings about each picture you are creating. Later, it will be beneficial for you to reexamine your answers to these questions to see if your viewpoint has changed.

You should be pleased with your progress at this point. You are well on your way to mastering Logo turtle graphics!

IV. Teaching the Turtle New Tricks

You have already learned enough about turtle graphics to create quite elaborate pictures. Even some simple patterns still require a lot of typing, however, and all this typing can be bothersome, especially if you find that you want to use certain figures over and over again. Also, you have not yet learned how to save your handiwork for later viewing. We should learn how to create our pictures in a form that lets us save them for viewing on another day.

As in the preceding chapter, this chapter is primarily devoted to developing more skill with Logo. We will do a few art projects at the end of this chapter; so if you are already familiar with the techniques of defining procedures, saving and loading procedures on a diskette or cassette tape, and using variables, you may want to skip briefly through the first part of the chapter and then do some of the projects. If you are not already familiar with these topics, this chapter will help you learn the remaining basic techniques needed to create stationary (as opposed to animated) works of computer art.

The Power of Procedures

Logo has a feature that lets us assign any collection of Logo commands to a word. For example, instead of drawing a 50-unit square by entering

```
REPEAT 4 [FORWARD 50 RIGHT 90]
```

each time, we can create a *procedure* that draws a square each time we enter the word

```
SQUARE
```

When we create a procedure, the procedure's name (SQUARE, for example) joins the library of words that Logo knows about. The process of creating procedures lets you extend the Logo language to suit your own needs. Tailoring the language by adding your own procedures, each with a name of your choosing, lets all aspects of this computer graphics medium fit your own expressive style. Logo starts with just enough building blocks to get you started. From there on, you are on your own!

Defining Procedures

Let's see if Logo can draw a 50-unit square by using a single word. Enter

```
CS
SQUARE
```

The error message

```
I DON'T KNOW HOW TO SQUARE
```

lets us know that SQUARE is not currently part of Logo's vocabulary. To add this word to the vocabulary, enter the words

```
TO SQUARE
```

As soon as you press the RETURN key, the question mark prompt will be replaced by a carat (>):

```
TO SQUARE
>_
```

Notice that the cursor is located to the right of the carat.

By typing the words TO SQUARE, you let Logo know that you are defining a procedure. Every procedure definition has at least three lines. For the type of procedure we will create now, the first line consists of the word TO followed by the procedure name. The last line is the word END. The lines in between are the commands that make up the procedure. (We don't have any of these yet.) These commands can use standard Logo words, such as FORWARD and the like, or they can use words for other Logo procedures you have created.

To define our SQUARE procedure, type

```
REPEAT 5 [FD 80 RT 144]
```

(Yes, I know this isn't going to give us a square. There is a method to my madness, however, so please bear with me.)

If you make a mistake while typing this line, you can fix it in the usual way by erasing the mistake with the DELETE/BS key. When you have finished entering this line, you will want to return to that part of Logo that lets us draw pictures. To finish the procedure, type

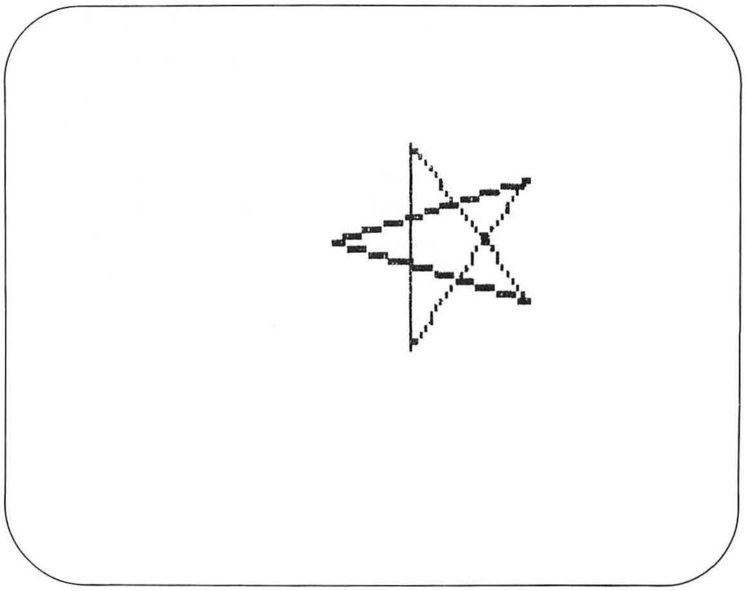
```
END
```

and press RETURN. Logo next types a message to let you know that the definition of the word SQUARE has been added to Logo's dictionary of commands.

Let's see what we have accomplished. Enter

```
CS
SQUARE
```

This draws a beautiful—star!



Why did I have you intentionally create a procedure to draw a star instead of a square? There are two reasons. First, this exercise shows that Logo does not "understand" the meanings of your words. It just treats the words as arbitrary labels. This procedure would have drawn a star if it had been named CIRCLE, RECTANGLE, FRED, or A25X. So long as you don't try to use words that are already in Logo's vocabulary, the names you choose for your procedures are totally up to you. The second reason I had you misdefine SQUARE is that it allows us to learn how to use the editor to fix the procedure so that we will get a square when we type SQUARE.

To edit this procedure, type

```
EDIT "SQUARE
```

(Note that Logo uses a quotation mark, as in "SQUARE, when referring to a name. In this case, we are referring to the name of a procedure.) When you press the RETURN key, you will see that we have reentered the Logo editor and that the

SQUARE procedure is listed on the screen. In order to fix our procedure so that it will draw a 50-unit square, we must be able to move the cursor to the places we need to change and to insert and delete characters. The Logo editor has several keys to help us in this task. Each of these keys is activated by holding down the CTRL key and pressing the appropriate key, as shown in the following table:

KEY	FUNCTION
A	Move the cursor to the beginning of the line
E	Move the cursor to the end of the line
X	Move the cursor to the beginning of the editor
Z	Move the cursor to the end of the editor
Up arrow	Move the cursor up one line
Down arrow	Move the cursor down one line
Left arrow	Move the cursor to the left by one line
Right arrow	Move the cursor to the right by one line

Additional editor operations include the following

KEY	FUNCTION
DELETE/BS	Delete the character to the left of the cursor
CTRL and DELETE/BS	Delete the character at the cursor
CTRL and CLEAR	Delete all characters from the cursor to the end of the line and save them in a special "delete" buffer
CTRL and Y	Insert the text that is currently in the delete buffer
ESC	Return Logo from the edit mode
BREAK	Stop the editor without making any changes and return to Logo

To insert a character, you just type it in, and the editor automatically shifts everything to make space for it.

Now, let's fix our procedure. First, move the cursor to the beginning of this line:

```
REPEAT 5 [FD 80 RT 144]
```

Next, move it (using the right arrow key) until it is under the space to the right of the 5:

```
REPEAT 5_[FD 80 RT 144]
```

Press the DELETE/BS key:

```
REPEAT_[FD 80 RT 144]
```

and press 4:

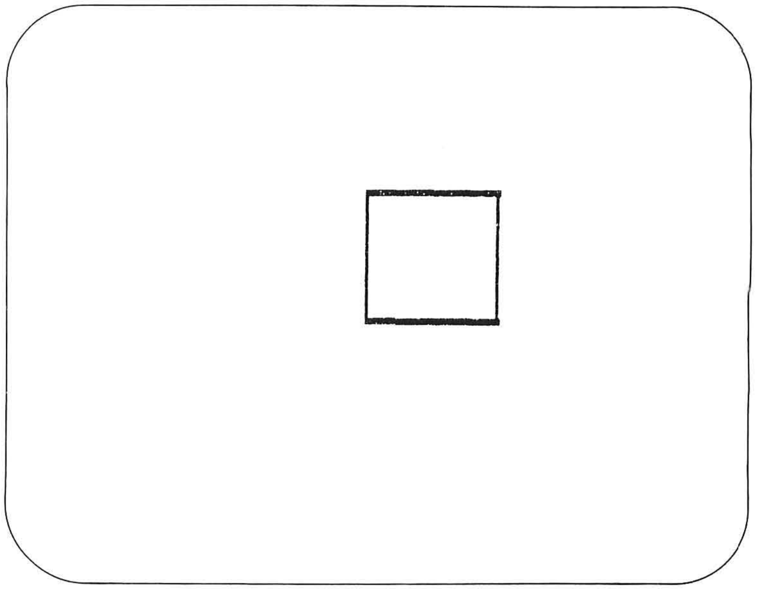
```
REPEAT 4_[FD 80 RT 144]
```

So far, so good. Now, continue to edit this line to change the 80 to a 50 and the 144 to a 90. When you are done, the line should read

```
REPEAT 4 [FD 50 RT 90]
```

At this point, press the ESC key, and Logo will have entered the modified definition of SQUARE into its library. Let's test our handiwork by entering

```
CS
SQUARE
```

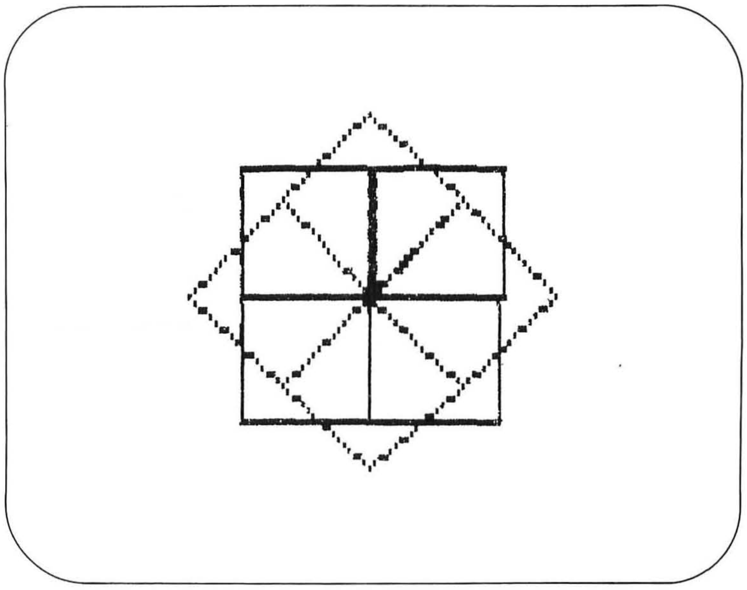


This time we see a square on the screen instead of a star.

Next, let's see if Logo treats SQUARE as it would one of its native words by entering the following:

```
REPEAT 8 [SQUARE RT 45]
```

When you press RETURN, you will see a pattern made from eight squares, repeated at equal angles around the center.



Now that you have learned how to define words in Logo, you can create complete works of art that can be displayed by typing only one word!

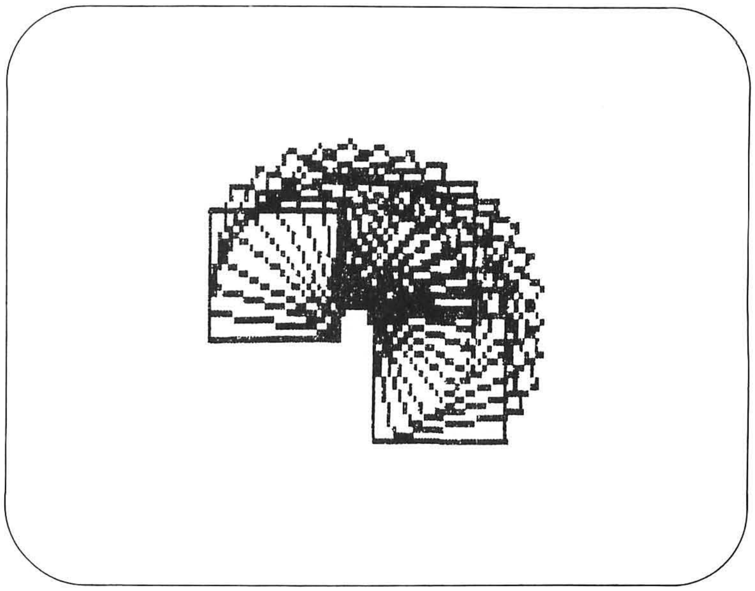
As an example, enter the following procedure:

```
TO PICTURE
CS HT
SETX -50
REPEAT 19 [SQUARE FD 10 RT 10]
END
```

To draw this picture on the display, just enter the word

```
PICTURE
```

and press RETURN.



Saving and Recalling Your Logo Workspace

Now that we have had a glimpse of how procedures can be useful to us, let's see how to save and recall procedures from the tape or disk memory. The advantage of saving your procedures is that you can recall them whenever you want without having to enter them again by hand. You may also find that some of your procedures are powerful building blocks that you will want to use in many pictures. These building block procedures can be saved as a separate extension of Logo, which you might want to recall each time you turn on the computer to work on a new picture. We will soon see some examples of useful building blocks.

As you create new procedures, you are adding new words to Logo's vocabulary. Logo allows you to save these new words (and their meanings) in something called a *file* on cassette tape or on a flexible disk. In case you have forgotten the names of the procedures you have created, you can see their names by entering

```
TS
POTS
```

The command POTS *Prints Out TitleS* of procedures on the screen. Your screen should now show the following procedure names:

```
TO SQUARE
TO PICTURE
```

To see the contents of any procedure, just enter PO (*Print Out*), followed by the procedure name. (The procedure name must be preceded by a quotation mark—for example, PO "PICTURE.) If you want to erase a procedure from your Logo workspace, just type the word ERASE, followed by the procedure name with a quotation mark in front of it (but don't do this yet).

When you save the procedures you have defined, you save all of them at once. Your file will contain the procedures for as many words as you have defined. Once your workspace has been saved on tape or disk, you may turn the computer off without fear of losing all your hard work. Every procedure you have defined (but not erased) will be saved on the tape or disk, ready to be recalled the next time you use the computer.

If you are using a disk system to save your procedures, you must first initialize a blank diskette, using the instructions that came with your disk system. If you are using cassette tape, you should save your procedures on high-quality cassettes (not the three-for-a-dollar variety). Follow the instructions that came with your computer system for connecting the cassette tape unit.

To save your files on a floppy disk, insert the diskette in the disk drive, close the door, and type the command

```
SAVE "D:filename
```

The "filename" should be a word of eight characters or less that describes your procedures. The reason for choosing a descriptive name is that your diskette ultimately will hold many of your procedure files. You could pick a name like SQUARES, for example, since both of your procedures use squares, or you could also call it CHAPT4 to remind you that the procedures came from the fourth chapter of this book.

Let's type

```
SAVE "D:CHAPT4
```

and press RETURN. The light on the disk drive will turn on, and you will hear a whirring sound as the procedures are being saved. As soon as the computer has finished this task, the question mark prompt will return to the screen, showing that Logo is ready for your next command.

One cautionary note is important if you are using the disk memory. You should always remember to remove the diskette when the power to the computer is being turned on or off. As a matter of general practice, you want to treat diskettes with great care and make sure that they don't get damaged by accidental "glitches" caused by power surges.

Now that you have saved your procedures, let's be bold and return Logo to its starting conditions. There are two ways to do this. You could turn off the computer and start over, or you could press the SYSTEM RESET key. The latter method has the same result as turning off the computer and turning it on again, but it is more gentle to the computer's electronic circuitry.

Once you have restarted Logo, type

```
POTS
```

You will notice that there are no procedures in the Logo workspace. To recall the procedures you generated, type

```
LOAD "D:CHAPT4
```

The disk light will turn on, and you will hear a whirring sound as the computer locates the filename and the procedures in the file CHAPT4 are loaded into your Logo workspace. To see that the procedures have been recalled, enter POTS. You will see that the procedures SQUARE and PICTURE are now available for your use.

If you are using a cassette recorder instead of a disk drive, you should save and recall procedures with the following commands:

```
SAVE "C:
```

or

```
LOAD "C:
```

Cassette files do not use filenames. When you save your procedures on the cassette, you will hear two beeps from the computer. This is your signal to press the RECORD and PLAY buttons on the recorder simultaneously. After the procedures have been saved, you will see the familiar Logo prompt on the screen, and you should press the STOP button on the recorder. To recall your procedures (with the LOAD "C: command), rewind the tape before entering the command. When you press RETURN, you will hear a single beep. This is your signal to press the PLAY button on the recorder. After the procedures have been loaded, you will again see the Logo prompt. Press the STOP button on the recorder, and you are finished!

Because of the importance of saving and recalling procedures, you may want to review this section before proceeding.

Variables

Although procedures such as SQUARE are quite useful, they have a limitation. Suppose that you want to make squares of different sizes—perhaps one of 25, one of 37, and one of 43 units. We could create new procedures for each of these—

SQUARE1, SQUARE2, and SQUARE3, for example. It would be far more powerful, however, if we could create one generalized procedure that would let us create squares of any size. The key to doing this in Logo is something called a *variable.* Before showing Logo's use of variables in procedures, we will make a slight digression to learn just what variables are.

A Logo variable can be thought of as a box that has a label and some contents. The box is named with a word, just as procedures are. When referring to the label for the box, the word is always preceded by a quotation mark ("). (We used the quotation mark in referring to a procedure's name when using commands like EDIT.) The contents of the box can be referred to by preceding the label name with a colon (:). Thus, the quotation mark denotes the name of the box and the colon denotes the contents of the box.

What kinds of things can be stored in Logo variables? Variables can contain numbers, words, or lists. To explore this a bit, we will introduce two new commands: MAKE and PRINT. MAKE lets us create a variable and provide it with some contents. PRINT lets us print text on the display screen; we can use this command to examine the things inside Logo variables.

To try out these commands, enter

```
TS CT
```

to clear the display screen (CT stands for *Clear Text*). Next, enter

```
MAKE "HUEY 6
```

This command instructs Logo to create a variable whose name is HUEY and whose contents is the number 6. To see the contents of this variable, type

```
PRINT :HUEY
```

When you press RETURN, you will see the number 6 on the display screen.

Next, enter

```
MAKE "HUEY "HORSEFEATHERS
```

If you now enter

```
PRINT :HUEY
```

the computer will display the word HORSEFEATHERS.

As a last example, enter

```
MAKE "HUEY [COW EAGLE DOG CAT]
```

and type

```
PRINT :HUEY
```

The computer will display the list COW EAGLE DOG CAT.

As you can see, we can make Logo variables hold all sorts of things.

Suppose that we want to increase the value of a number that is stored in a Logo variable. How can we do this? Let's look at an example. First, create a variable that contains a number:

```
MAKE "FRED 23
```

If you type

```
PRINT :FRED
```

you will see the number 23 on the screen. Next, enter

```
MAKE "FRED :FRED + 1
```

What do you think this means? The best way to translate this command may be: "Make the variable whose name is FRED contain the contents of FRED plus one." We can see if we get the desired result by entering

```
PRINT :FRED
```

You will see that the number printed on the screen is 24, just as we expected. By repeating this last MAKE command, we can increase the number in FRED to 25, 26, 27, 28, and so on. We have made a *counter*! Counters are very useful tools, as we will see later.

Now we are ready to examine the use of variables in procedures. As mentioned earlier, it would be very useful for us to be able to define a procedure that lets us create squares of any size. By typing SQUARE 27, for example, we could draw a 27-unit square.

To define this procedure, first enter

```
ERASE "SQUARE
```

to eliminate the old procedure, and then use the procedure editor to enter

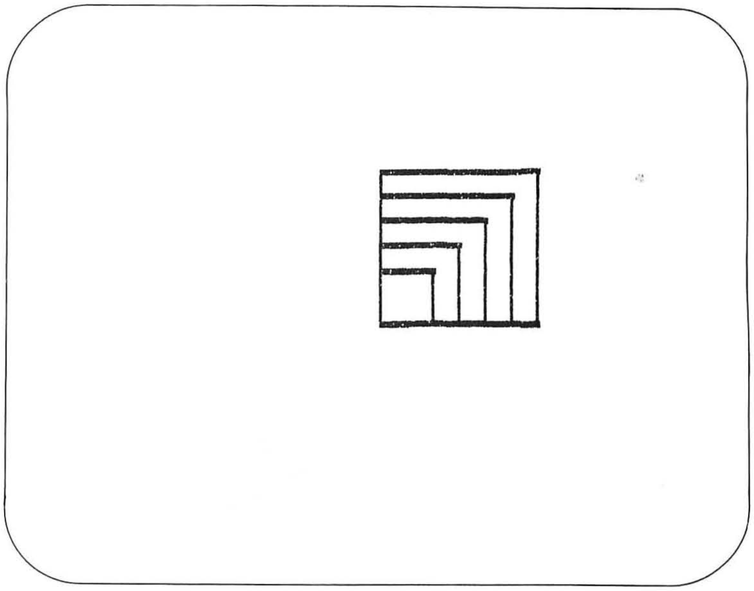
```
TO SQUARE :SIZE
REPEAT 4 [FD :SIZE RT 90]
END
```

To create this procedure, type EDIT SQUARE. When you press RETURN, you will see that you have entered the Logo procedure editor, with the cursor at the T in TO. Using the right arrow key (remember to hold the CTRL key down), move the cursor to the end of the line. Press the space bar once and enter :SIZE. Now press the RETURN key. Type the next two lines of the procedure, and press the ESC key to return to Logo.

This procedure uses the variable SIZE to contain the length of each side of the square. This variable is uniquely identified by its name and by the procedure in which it appears. This means that you could use the word SIZE as a variable in another procedure, and Logo would never be confused about which variable you were using. If you give a command such as SQUARE 20, the local value for SIZE (20) will be "passed" inside the procedure to be used with the FD command.

Let's see how our new procedure works. Enter

```
CS HT
SQUARE 20
SQUARE 30
SQUARE 40
SQUARE 50
SQUARE 60
```



This is a much easier way to create nested boxes than the way we did it in the last chapter!

We can pass the contents of variables as well as numbers with our procedures. To see an example of this, enter the procedure

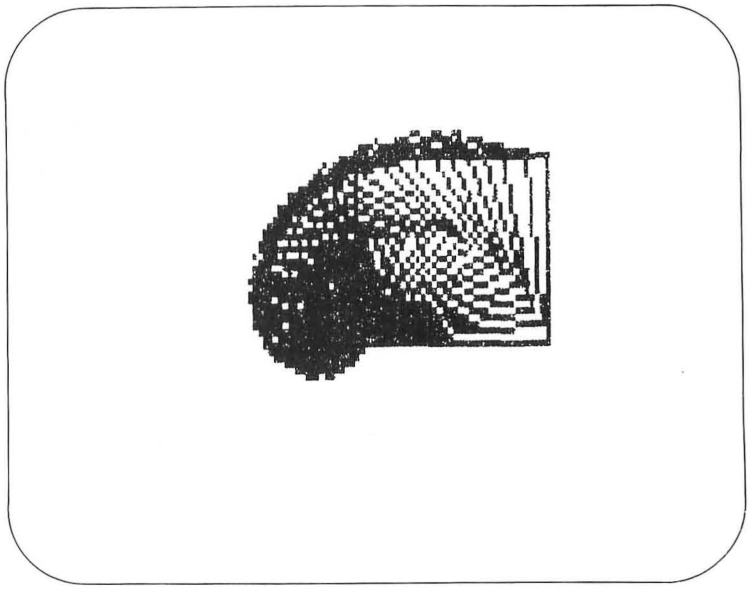
```
TO PICTURE2
CS HT SETH 180
MAKE "COUNT 1
REPEAT 37 [SQUARE :COUNT RT 5 MAKE "COUNT :COUNT + 2]
END
```

After pressing the ESC key to leave the editor, enter

```
PICTURE2
```

This procedure creates a series of tilting squares with increasing sizes. Each square's size is given by the contents of the variable

COUNT, and this value is increased by two each time the command sequence is repeated.



Each time you create a new picture, you should save your workspace so that nothing will be lost if you accidentally turn off the computer.

Projects That Use Procedures

Recent reports from the distant planet Spork indicate that their artists enjoy using a computer graphics language called GOLLI. The golli is actually a Sporkian animal that can frizzle in a straight line or can rizzle by up to 360 drigs before heading in the direction it started from. We found that one could draw a 40-unit square (called a *snarp* in GOLLI) by giving the command

```
SPLEGO 4 [FRIZZLE 40 RIZZLE 90]
```

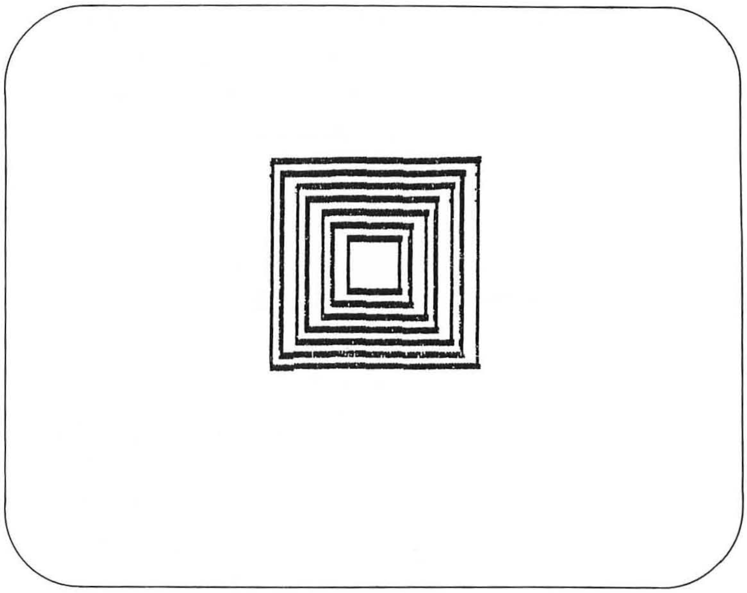
Your task is to create the procedures in Logo that let you enter this Sporkian command and have it work. Next, you should write the GOLLI commands to draw a five-pointed star and demonstrate that it works. Here is one procedure to help you get started:

```
TO SPLEGO :NUMBER :LIST
REPEAT :NUMBER :LIST
END
```

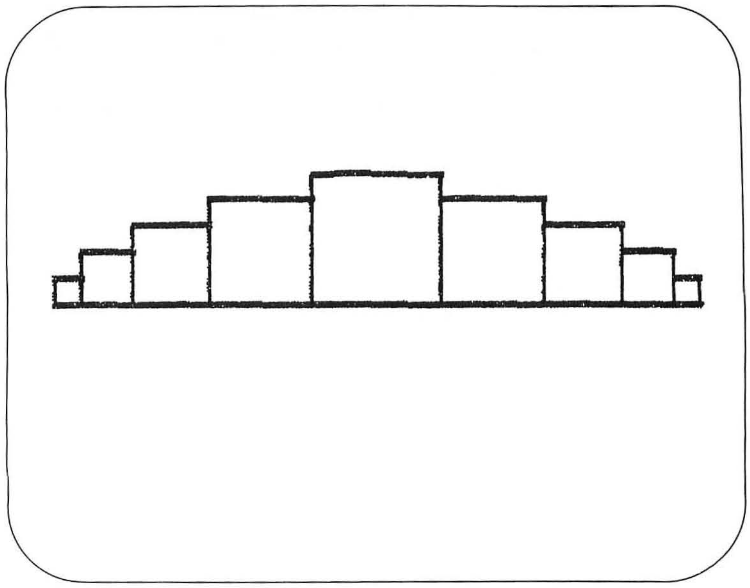
You should be able to create the other procedures you need without much trouble.

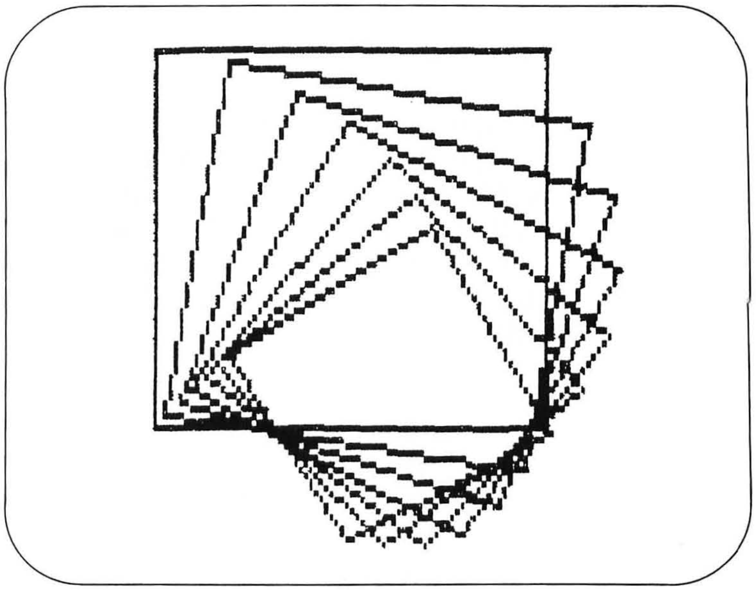
Moving back to more familiar terrain, you can create many interesting pictures with figures as simple as the square. For example, the following figure was created with the procedure

```
TO PICTURE3
CS HT
PU SETPOS [ -40 -40] PD SQUARE 80
PU SETPOS [ -35 -35] PD SQUARE 70
PU SETPOS [ -30 -30] PD SQUARE 60
PU SETPOS [ -25 -25] PD SQUARE 50
PU SETPOS [ -20 -20] PD SQUARE 40
PU SETPOS [ -15 -15] PD SQUARE 30
PU SETPOS [ -10 -10] PD SQUARE 20
END
```



See if you can create the procedures to generate pictures that look like those in the following figures.





The following procedures show one way to create these pictures:

```
TO PICTURE4
CS HT
SETX -125
MAKE "SIZE 10
REPEAT 5 [SQUARE :SIZE RT 90 FD :SIZE LT 90 MAKE
  "SIZE :SIZE + 10]
MAKE "SIZE :SIZE - 20
REPEAT 4 [SQUARE :SIZE RT 90 FD :SIZE LT 90 MAKE
  "SIZE :SIZE - 10]
END
```

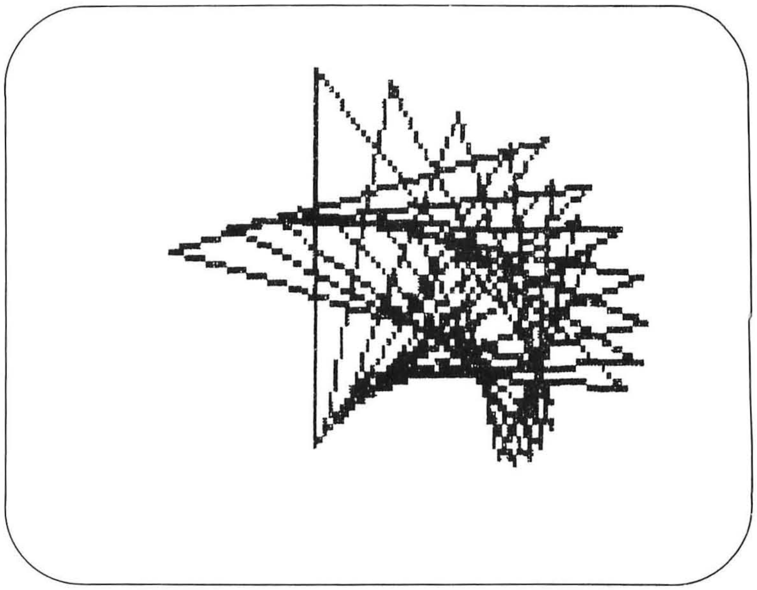
```
TO PICTURE5
CS HT FS
PU SETPOS [ -40 -30] PD SQUARE 150 RT 10
PU SETPOS [ -35 -25] PD SQUARE 140 RT 10
PU SETPOS [ -30 -20] PD SQUARE 130 RT 10
PU SETPOS [ -25 -15] PD SQUARE 120 RT 10
PU SETPOS [ -20 -10] PD SQUARE 110 RT 10
PU SETPOS [ -15 -5] PD SQUARE 100 RT 10
PU SETPOS [ -10 0] PD SQUARE 90 RT 10
END
```

Now you should create some pictures of your own that use squares. Try using different colors for the lines and background. Make pictures with a lot of symmetry; then make pictures that use random sizes, orientations, and starting positions. Which patterns do you like better? Write down your feelings about each picture in your journal. Save all your artwork for later viewing.

Finally, modify the procedure SQUARE so it draws a five-pointed star. You will want to edit it to look like this:

```
To SQUARE :SIZE
REPEAT 5 [FD :SIZE RT 144]
END
```

Now repeat the various PICTURE procedures we have created. Here is what PICTURE5 looks like with stars instead of squares.



Does the star change the nature of the overall picture by a little or a lot? Which patterns do you like better?

In the next chapter we will use our knowledge of Logo turtle graphics to create many more interesting patterns. You have now mastered the fundamental technique of turtle graphics, so let's move on to the next stage!

V. The Four Little Polygons and How They Grew

Thus far, we have been able to create pictures using two kinds of building blocks—lines and squares. In the next few chapters, we will explore many classes of building block procedures that may be powerful tools in your hands. Some of the building blocks are quite primitive, and others are very detailed. Some of the detailed ones can stand on their own as designs, yet any of them may be used in combination with others to create even more beautiful patterns.

As in previous chapters, we will finish with a few projects to let you try out your skills. Before working on this chapter, you should clean out your workspace so that it contains no procedures. The easiest way to do this is to restart Logo by pressing the SYSTEM RESET button.

Polygons

In the last chapter we used this procedure:

```
TO SQUARE :SIZE
REPEAT 4 [FD :SIZE RT 90]
END
```

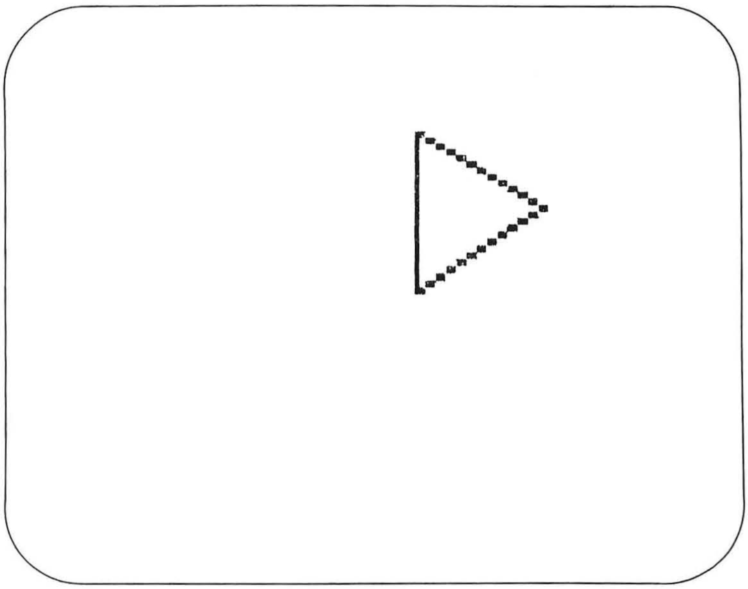
The square generated by this procedure is an example of a regular polygon. Although squares can form the basis for many interesting pictures, there are also many other pretty polygons to use. To see one example, let's create a procedure that generates an equilateral triangle:

```
TO TRI :SIZE
REPEAT 3 [FD :SIZE RT 120]
END
```

If you now enter

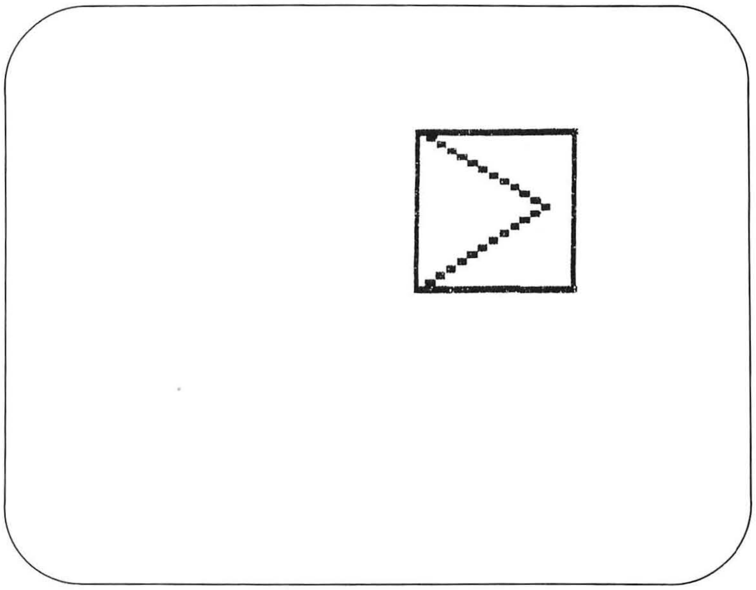
```
CS HT
TRI 60
```

you will see a triangle on the screen.



You can draw a box around the triangle by entering

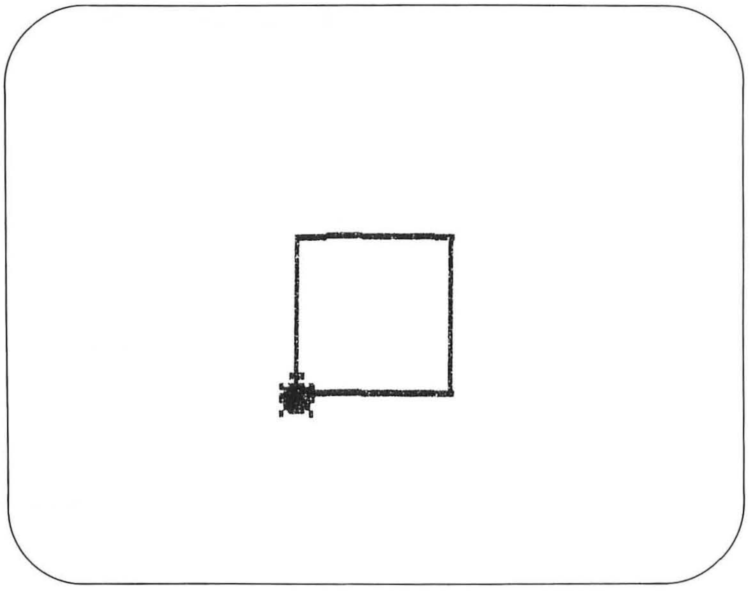
```
SQUARE 60
```



Suppose that you wanted to draw a simple picture of a house. You could use a square for the front of the house and a triangle for the roof. Experiment a bit to see how to put the roof on the house.

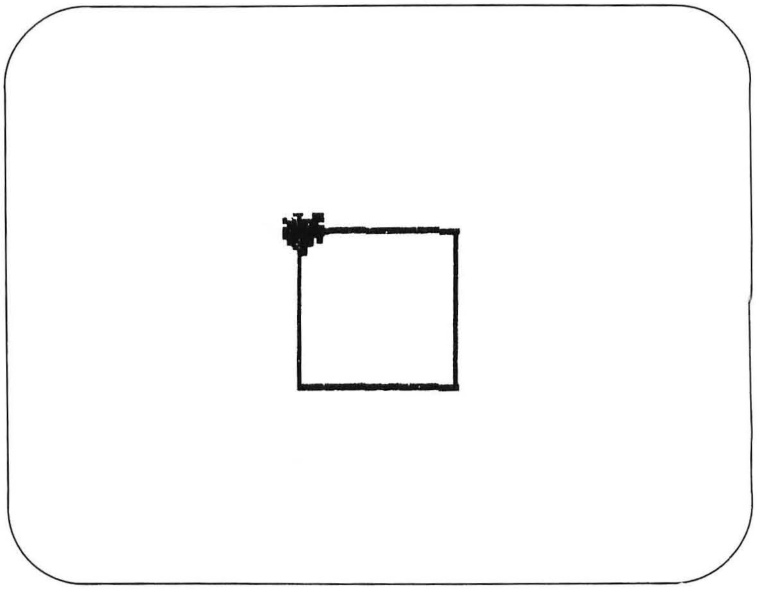
One way to create the house is to start with the bottom part:

```
CS ST
SQUARE 60
```



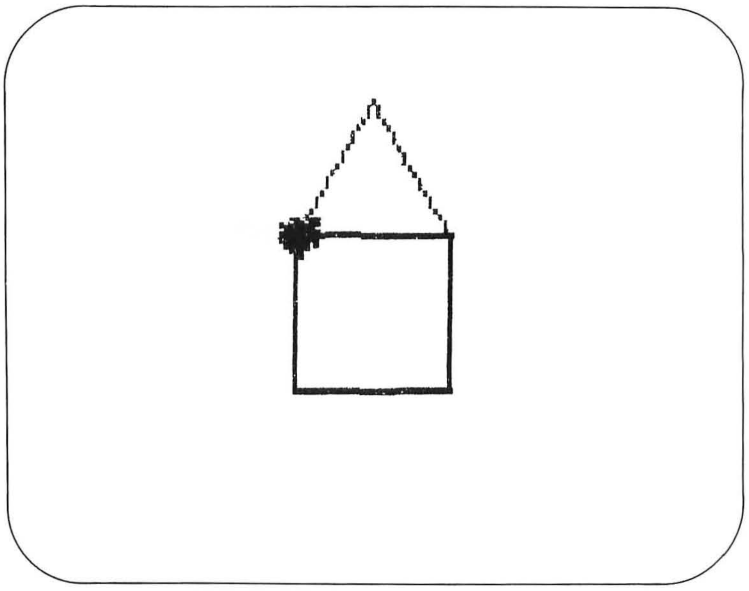
and then move the turtle to the upper left corner of the box and tilt it to the right by 30 degrees:

```
FD 60 RT 30
```



Finally, you can draw the roof by entering

```
TRI 60
```



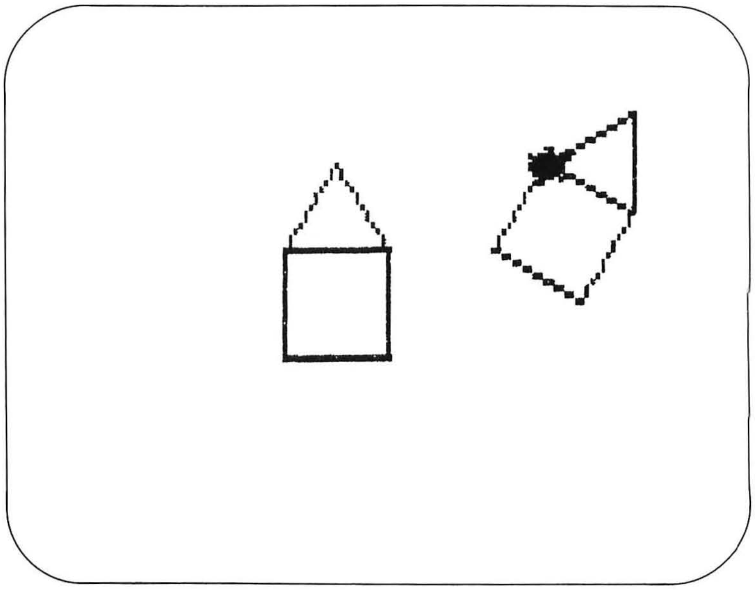
(As you can see in this exercise, the visible turtle can be of great help in checking on the placement of figures in your pictures.)

You could easily create a procedure for drawing houses of any size; for example:

```
TO HOUSE :SIZE
SQUARE :SIZE
FD :SIZE RT 30
TRI :SIZE
END
```

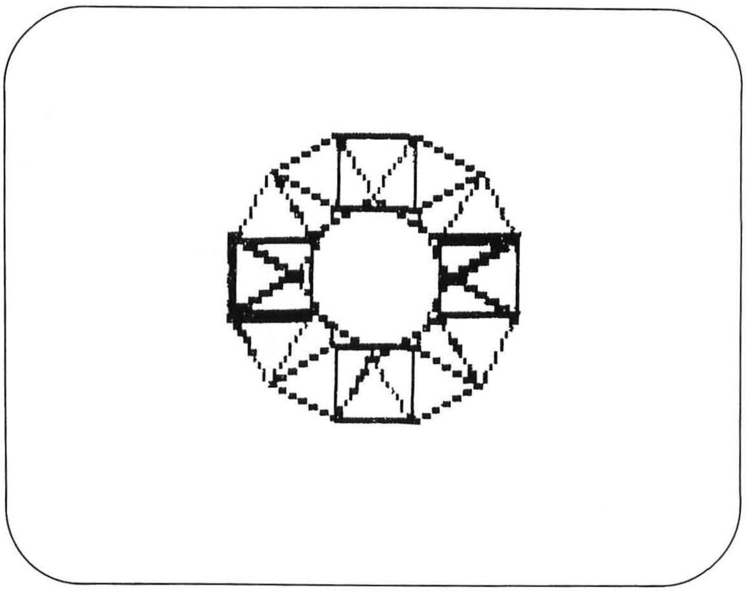
Let's use this procedure to create two houses side by side:

```
CS
PU SETX -50 PD
HOUSE 40
PU SETX 30 PD
HOUSE 40
```



Oops—our second house is tilted and offset from the first. The reason this happened is because the procedure HOUSE does not leave the turtle in the same position and orientation it had when it started. Before we fix this problem, let's experiment with HOUSE some more. Enter

```
CS HT
REPEAT 12 [HOUSE 30]
```



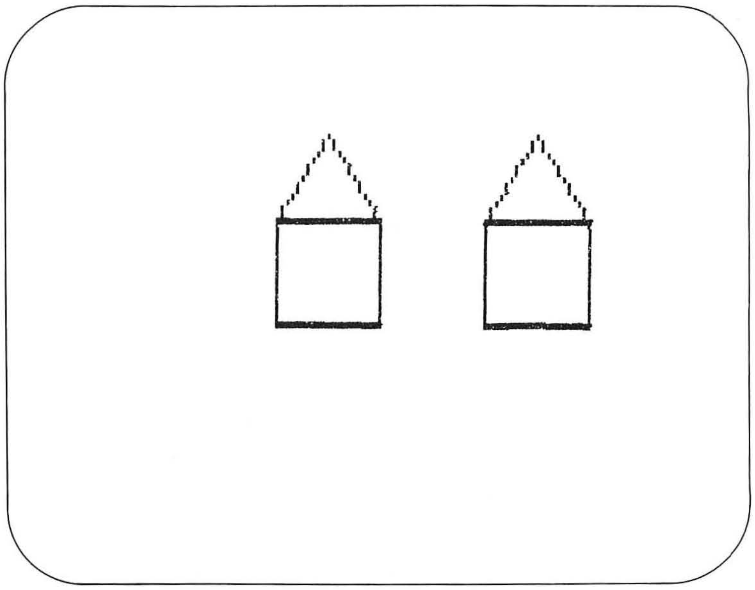
This is a pretty pattern—one you might wish to develop into a more detailed picture using different sizes and colors for HOUSE.

To fix our problem with HOUSE, we can return the turtle to its starting position and orientation by adding one line to the procedure. Modify the HOUSE procedure so that it looks like this:

```
TO HOUSE :SIZE
SQUARE :SIZE
FD :SIZE RT 30
TRI :SIZE
LT 30 BK :SIZE
END
```

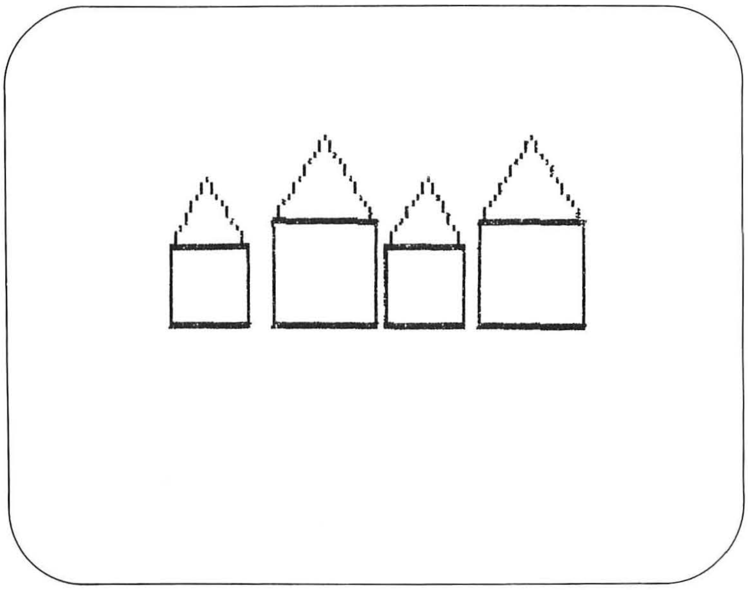
Now, if we repeat our previous experiment, we will get two houses side by side:

```
CS
PU SETX -50 PD
HOUSE 40
PU SETX 30 PD
HOUSE 40
```



We can add more houses of different sizes to this picture. Here are two more to get you started:

```
PU SETX -90 PD HOUSE 30
PU SETX -5 PD HOUSE 30
```



Blossoms

Triangles and squares are useful for much more than drawing houses. We can also create patterns by repeating polygons after turning the turtle by a chosen angle. If we keep repeating this process, we will get a flower blossom. For example, in the last chapter we created a pattern by entering

```
REPEAT 8 [SQUARE 30 RT 45]
```

Suppose that we want to make a blossom-drawing procedure that lets us use any polygon we want as the repeated unit in the blossom. To do this, we need to create a list, such as [SQUARE 20 RT 45] or [TRI 30 RT 45]. We can put the polygon procedure name in a variable, such as LIST, but we need to find a way to append RT 45 to this name. Logo lets you combine two lists with the SE (*SEntence*) command.

See if you can figure out what SE does in the following procedure:

```
TO BLOSSOM :LIST
REPEAT 8 SE :LIST [RT 45]
END
```

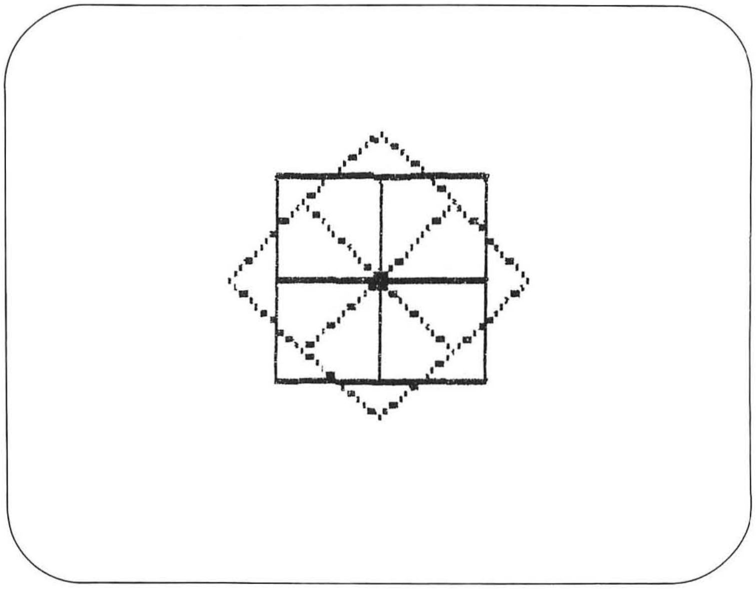
If the contents of LIST is [TRI 37], for example, SE creates a new list that looks like this:

```
[TRI 37 RT 45].
```

This is exactly what we want!

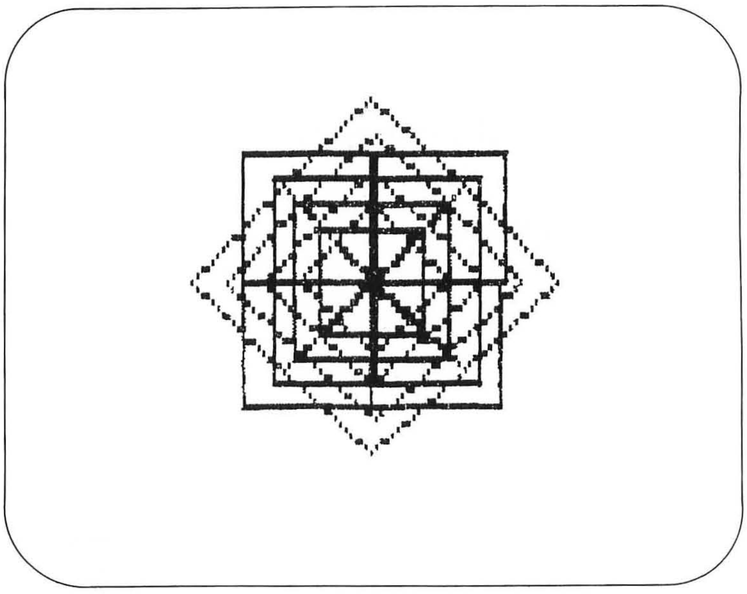
Try entering

```
CS HT
BLOSSOM [SQUARE 40]
```



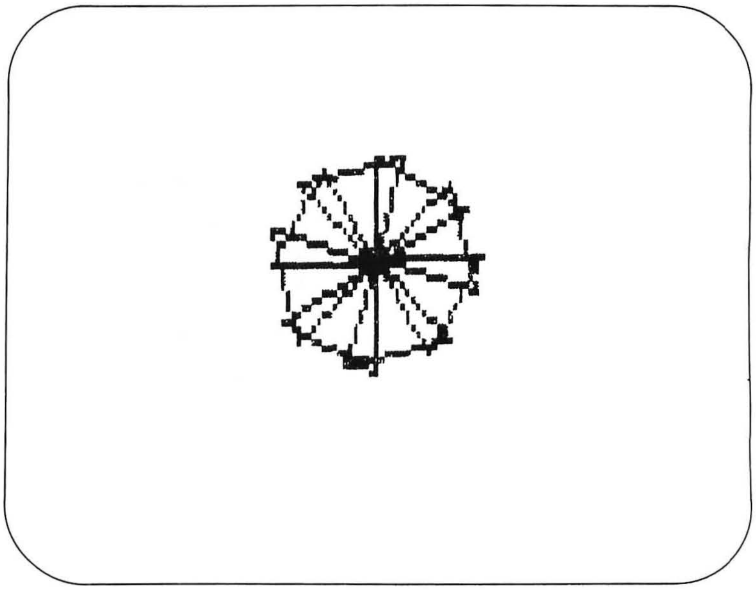
This gives us the same figure we saw in the last chapter. Now let's add to this picture by entering

```
BLOSSOM [SQUARE 50]
BLOSSOM [SQUARE 30]
BLOSSOM [SQUARE 20]
```



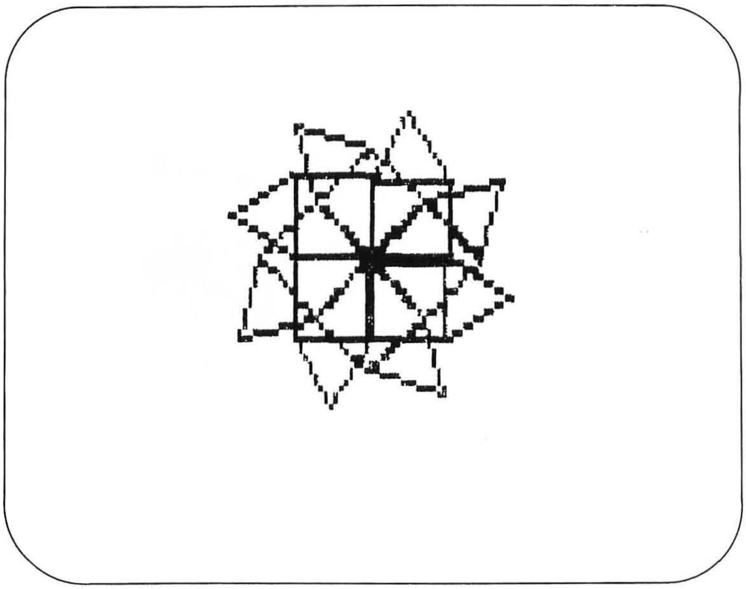
BLOSSOM can create many interesting patterns. Try

```
CS
BLOSSOM [TRI 40]
```



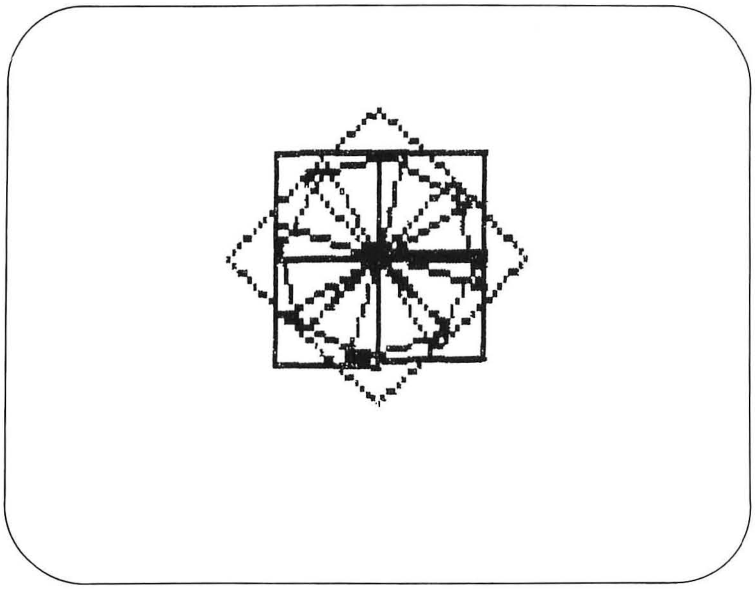
And try

```
CS
BLOSSOM [HOUSE 30]
```



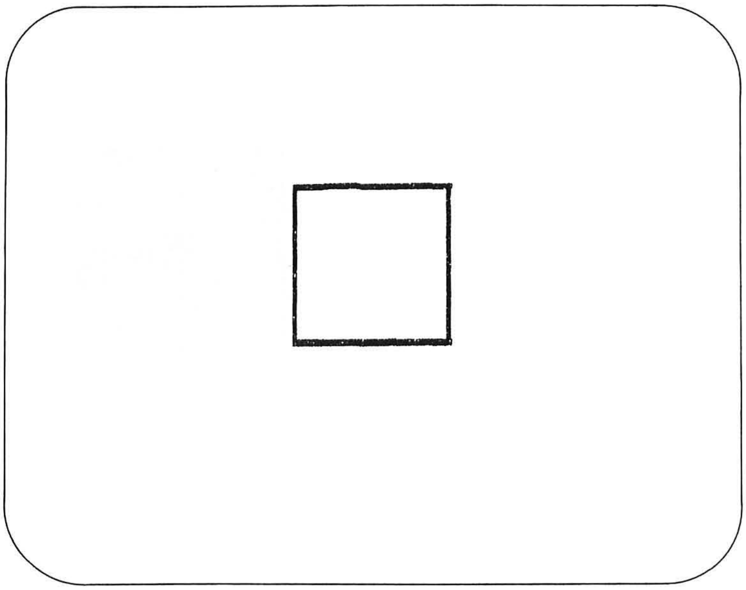
What do you think will happen if you try

```
CS
BLOSSOM [SQUARE 40 TRI 40]
```



Were you surprised? If not, try to guess what will happen if you enter

```
CS
BLOSSOM [SQUARE 60 LT 45]
```



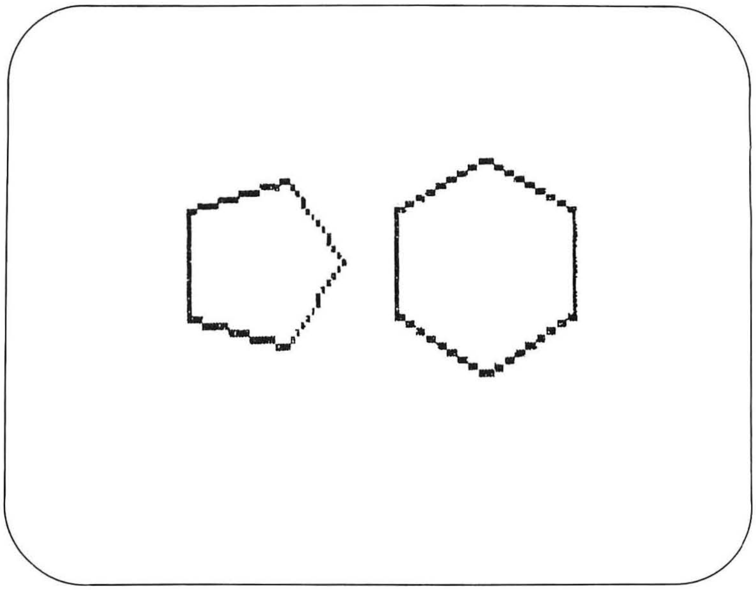
All we see is a single square! The reason for this is fairly simple—our command LT 45 acted to counterbalance the RT 45 command in the BLOSSOM procedure.

Now that we know about squares and triangles, what about other polygons? The following two procedures let you create regular pentagons and hexagons:

```
TO PENT :SIZE
REPEAT 5 [FD :SIZE RT 72]
END

TO HEX :SIZE
REPEAT 6 [FD :SIZE RT 60]
END
```

To see the figures generated by these two procedures, enter

```
CS HT
PU SETX -50 PD PENT 40
PU SETX 30 PD HEX 40
```

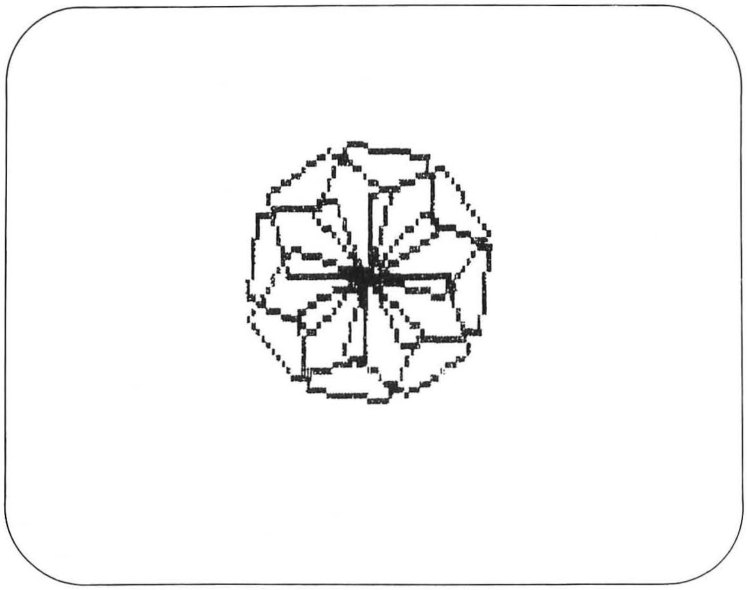


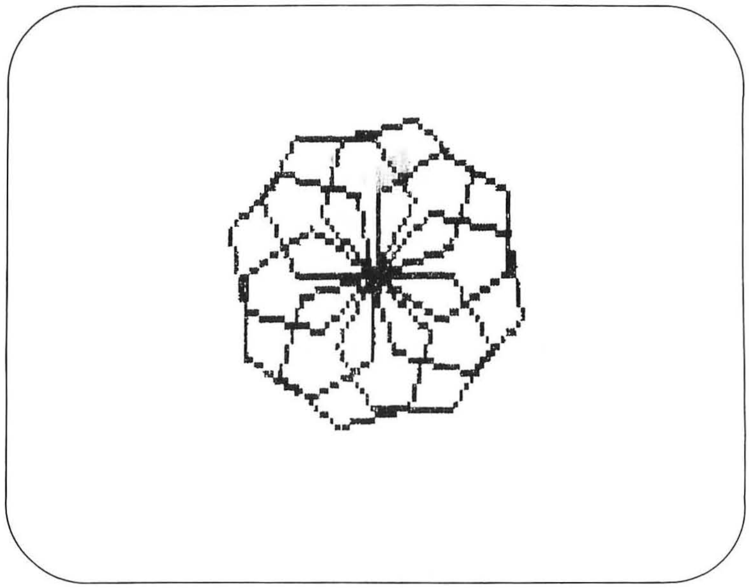
BLOSSOM makes nice patterns with pentagons and hexagons, as you can see by entering

```
CS
BLOSSOM [PENT 30]
```

and

```
CS
BLOSSOM [HEX 30]
```





POLY and the Single Procedure

Because polygons are such useful building blocks, it would be a great benefit to have a single procedure that would let us create regular polygons with any number of sides. Before we create such a procedure, we should find out what properties are shared by all polygon procedures and see if this helps us design a general procedure.

We can start by looking at the commands that *draw* triangles, squares, pentagons, and hexagons:

```
REPEAT 3 [FD :SIZE RT 120]
REPEAT 4 [FD :SIZE RT 90]
REPEAT 5 [FD :SIZE RT 72]
REPEAT 6 [FD :SIZE RT 60]
```

These commands appear to be similar except for the two numbers. The first number is simply the number of sides in the polygon. The second number is the angle through which the turtle turns before drawing the next side.

As the number of sides increases, the turning angle decreases. Let's examine the *total* angle turned by the turtle while drawing each polygon. For the triangle, this number is 3×120, or 360 degrees. For a square, it is 4×90, or 360 degrees. You can easily prove to yourself that this result is also true for pentagons and hexagons. In fact, any closed turtle path that doesn't cross over itself has a total turning angle of 360 degrees. (For the mathematically inclined reader, this is called the *Turtle Total Trip Theorem.* For the nonmathematically inclined reader, feel free simply to enjoy the result—coming soon—without needing to understand its derivation.)

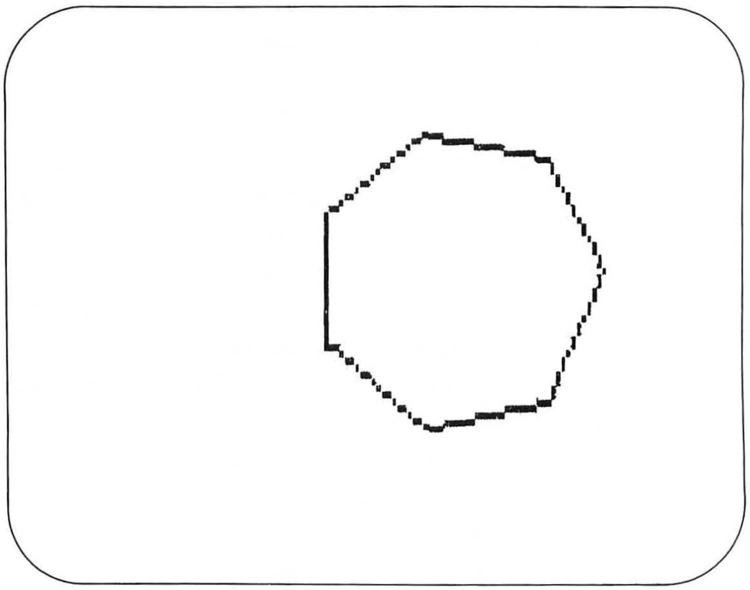
The point of this exercise is that we now know that the turning angle needed for any regular polygon is given by 360 divided by the number of sides. The Logo symbol for division is the solidus (/). See if you can determine that the following procedure, POLY, wil let us create any regular polygon. Enter

```
TO POLY :SIDES :SIZE
REPEAT :SIDES [FD :SIZE RT (360 / :SIDES)]
END
```

(We used parentheses to group the calculation 360 / :SIDES as a reminder that this should be done before the turtle turns to the right.)

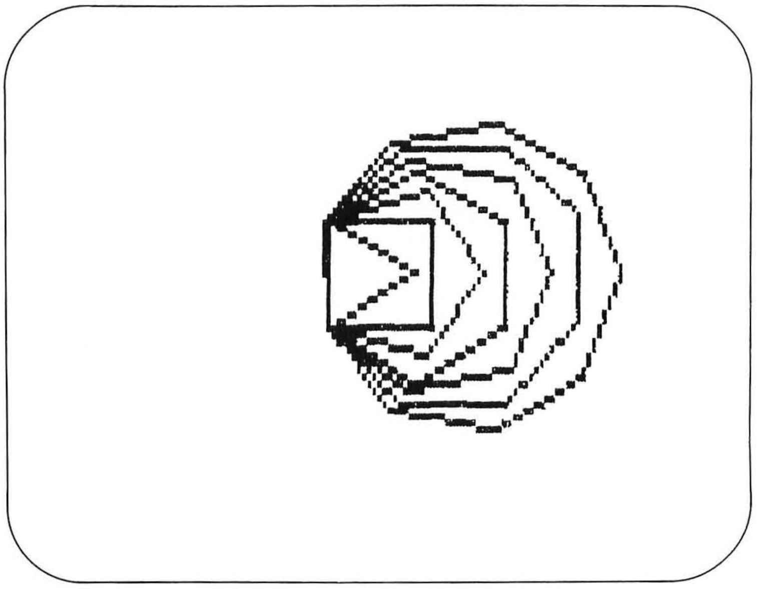
How does POLY work? Try POLY 4 30 to see if you get a square. So far, so good. Next, enter

```
CS
POLY 7 50
```



To see a collection of polygons, enter

```
CS
POLY 3 40
POLY 4 40
POLY 5 40
POLY 6 40
POLY 7 40
POLY 8 40
POLY 9 40
```



Procedures such as POLY are quite valuable because they are compact and yet let us create a large number of different figures with very simple commands. Compactness is important because the memory of your computer is fixed in size; there is a limit to the number of procedures it can hold. The procedure POLY is the same size as SQUARE, yet it is far more powerful because it lets us create many different polygons. If we erase TRI, SQUARE, PENT, and HEX, we will be able to have more procedures in our workspace without giving up our ability to create triangles, squares, pentagons, or hexagons.

Of course, POLY doesn't let us make all polygons. For polygons with unequal sides or unequal angles, we will have to create new procedures. Let's look at one example.

A Pentagonal Tile

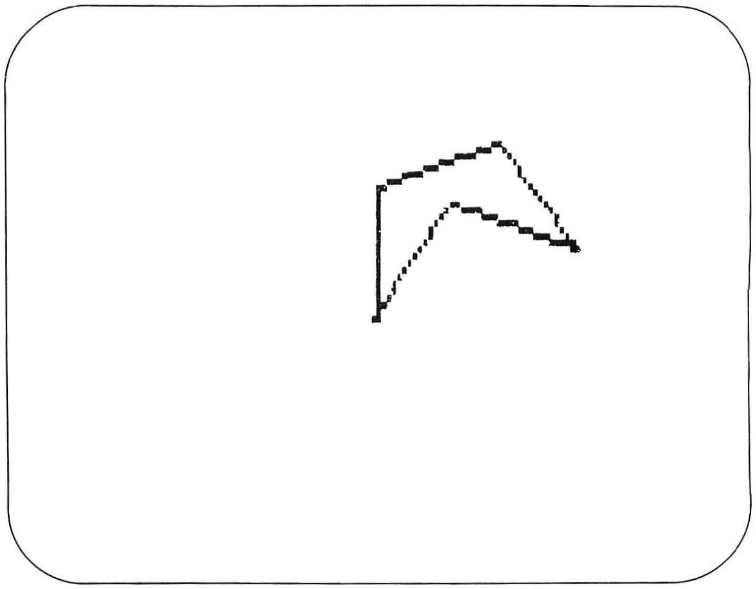
Many different polygons can be combined to create pictures. Sometimes the pictures are representational (such as our use of a square and a triangle to make a house), or they might be abstract (such as some of the BLOSSOM patterns). It is interesting to find polygons that can be arranged with other copies of themselves to generate hundreds of different patterns. One such polygon is generated by the PENTILE procedure:

```
TO PENTILE :SIZE
REPEAT 2 [FD :SIZE RT 72]
FD :SIZE RT 144
FD :SIZE LT 72
FD :SIZE RT 144
END
```

(Note that if you add up the angles turned to the right and subtract the angle turned to the left, the result is 360 degrees.)

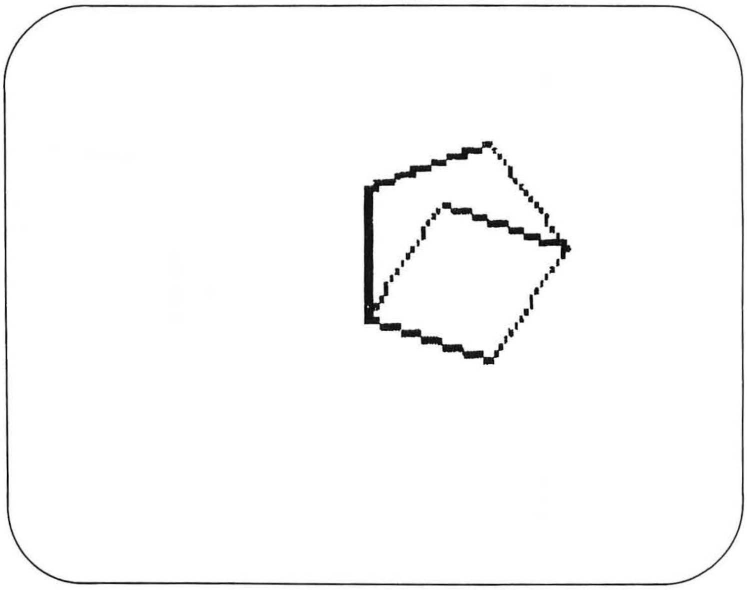
To see a picture of this tile, enter

```
CS
PENTILE 50
```



This figure is a pentagon because it has five sides, but it looks very different from the pentagon created by POLY. To see the relation between the two, enter

```
POLY 5 50
```



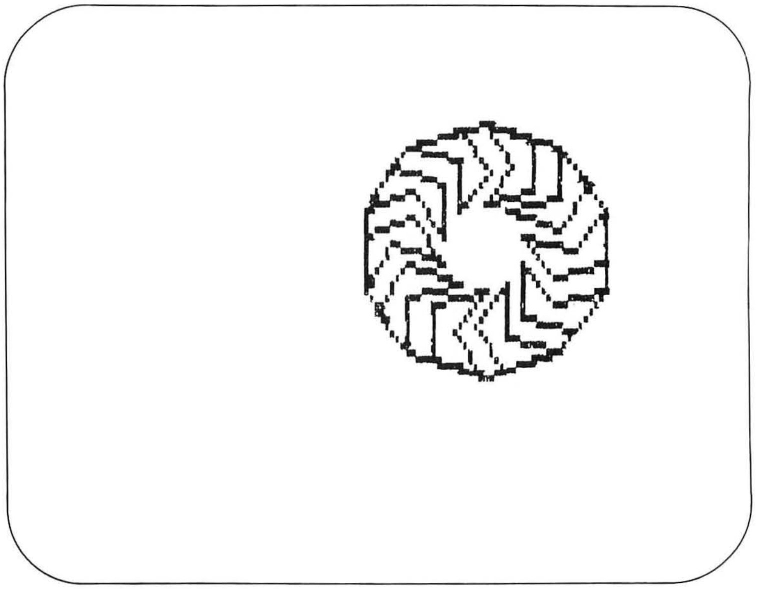
The only difference is that PENTILE was made by snapping in one of the corners of our regular pentagon.

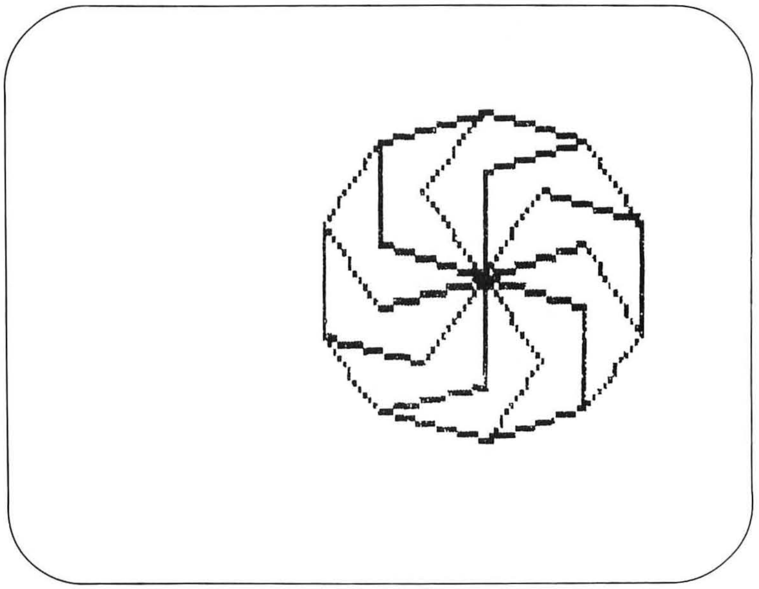
PENTILE polygons can be arranged in many interesting patterns. You may even want to cut some out of cardboard so that you can experiment when you are away from the computer. (A beautiful wooden set of tiles based on this figure is available in toy stores under the name PENTALBI—trademark, Kurt Naef.) The following two figures show just a few of the many patterns you can create with this polygon. These figures were created with the commands

```
CS
REPEAT 10 [PENTILE 20 FD 30 RT 36]
```

and

```
CS
REPEAT 10 [PENTILE 40 RT 36]
```





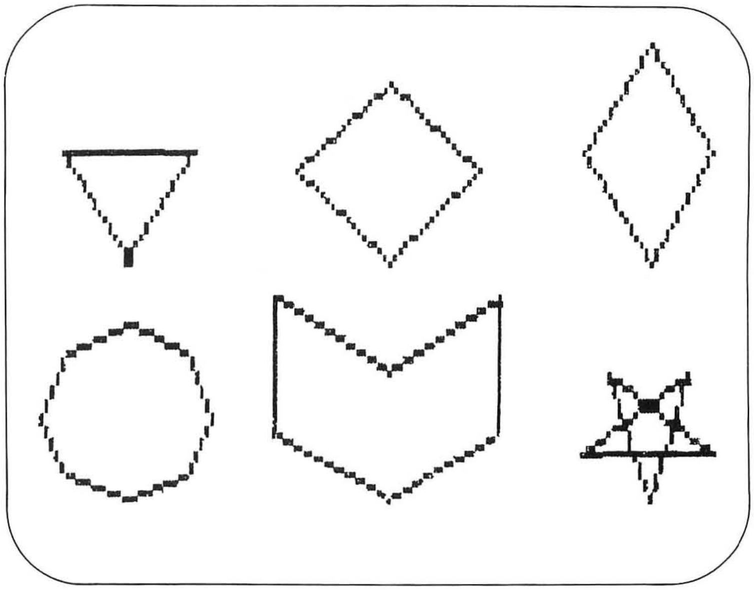
Kaleidoscopes

Simple polygonal patterns can be used to make very attractive pictures—for example, the patterns generated by a kaleidoscope. If you have ever taken a real kaleidoscope apart, you must have wondered how such a simple apparatus could generate such beautiful images. Most kaleidoscopes consist of a set of mirrors and some small pieces of colored plastic that can be shaken to take random positions on a flat surface. When you look through the eyepiece, the mirrors generate multiple images of the arrangement of plastic pieces to produce beautifully symmetrical pictures. Because Logo's turtle graphics allows us to create images that act like the pieces of plastic, it is possible to create kaleidoscopic images on the computer screen with a simple set of procedures.

The Logo kaleidoscope operates in the following manner. The system contains a set of graphic procedures to draw the fundamental picture elements (squares, triangles, stars, and so on). There can be as many of these elements as you desire (subject to the memory limitations of your system, of course), and each of the elements can be drawn in any size. This gives the effect of having even more patterns to choose from.

We use Logo's random-number generator to select a shape, the shape's size and color, and the distance from the center of the screen at which the shape will be drawn. Finally, these data are used by another procedure, which places a copy of the chosen shape at several equally spaced angles around the center of the screen. Once one shape has been drawn, the process can be repeated for other shapes until the final image meets with your approval.

Our kaleidoscope will start out with six shapes.



The procedures for these shapes are as follows:

```
TO TRIANGLE :SIZE
LT 30
POLY 3 :SIZE
RT 30
END

TO DIAMOND :SIZE
LT 45
POLY 4 :SIZE
RT 45
END

TO PATT1 :SIZE
LT 30
REPEAT 2 [FD :SIZE RT 60 FD :SIZE RT 120]
RT 30
END
```

```
TO OCT :SIZE
LT 67.5
POLY 8 (:SIZE / 2)
RT 67.5
END

TO PATT2 :SIZE
LT 60
FD :SIZE RT 60 FD :SIZE RT 120
FD :SIZE LT 60 FD :SIZE RT 120
FD :SIZE RT 60 FD :SIZE RT 120
END

TO STAR :SIZE
LT 18
REPEAT 5 [FD :SIZE RT 144]
RT 18
END
```

Each of these figures has been defined to have mirror symmetry on the vertical axis by adjustment of the angle at which the pattern is drawn. This is not a requirement, and you may wish to experiment with other orientations. The octagon was drawn at half the specified size to keep it in balance with the other figures.

Key elements of our kaleidoscope procedures are the Logo functions RANDOM and OUTPUT. RANDOM randomly picks a number between 0 and one less than the number specified. For example, RANDOM 50 will pick a number between 0 and 49. To see how RANDOM works, enter

```
TS CT
REPEAT 10 [PRINT RANDOM 50]
```

Ten randomly chosen numbers will appear on the screen.

OUTPUT is a Logo primitive that lets us pass values back out of a procedure. For example, if we wanted to create a Logo procedure that calculated the square of a number, we could enter

```
TO SQ :N
OUTPUT :N * :N
END
```

If we then typed

```
PRINT SQ 9
```

The screen would show 81. OUTPUT is different from PRINT in that it produces a result that is ready to be used as input by another Logo procedure.

To make the kaleidoscopic image, we need a procedure that creates a list of basic patterns, chooses a pattern at random from this list, and selects an appropriate size (for example, between 20 and 50 units). Next, it should pick a random distance from the center (less than 60 units, to keep the images on the screen). Once these steps have been completed, copies of the chosen image should be stamped symmetrically around the screen. Then the procedure should wait for you to tell it if you want another element added to the image. When you press any key, the process will be repeated. The following procedure performs these tasks for us:

```
TO IMAGE
MAKE "LIST [STAR DIAMOND OCT PATT1 PATT2 TRIANGLE]
MAKE "NAME SE PICKRANDOM :LIST (20 + RANDOM 30)
MAKE "DIST RANDOM 60
SETPN RANDOM 3
PENUP
WINDMILL :DIST :NAME
MAKE "NAME RC
IMAGE
END
```

This procedure uses two other procedures that have to be defined: PICKRANDOM and WINDMILL. The function of PICK-RANDOM is to choose an element of a list randomly. The following procedure does this for us:

```
TO PICKRANDOM :LIST
OUTPUT PICK (1 + RANDOM (LENGTH :LIST) ) :LIST
END
```

The procedure PICK selects a given element from a list, and LENGTH measures the number of elements in a list:

```
TO PICK :NUM :LIST
IF :NUM = 1 [OUTPUT FIRST :LIST]
OUTPUT PICK (:NUM - 1) (BUTFIRST :LIST)
END

TO LENGTH :LIST
IF :LIST = [] [OUTPUT 0]
OUTPUT 1 + LENGTH BUTFIRST :LIST
END
```

The IF command in these procedures operates in the following way. If the statement following the IF command is true (for example, if the value of NUM is equal to 1), then the instructions enclosed in the list (for example, OUTPUT FIRST :LIST) will be executed; otherwise, they will not be.

Another subtlety associated with these procedures is that they operate *recursively*. If you have difficulty understanding how they work, you may want to read about them in *Logo for the Apple II*, by H. Abelson (McGraw-Hill), or read the chapter on recursion in *Discovering Apple Logo*, by D. Thornburg (Addison-Wesley). It is also all right just to view them as handy tools without going into the details of their operation.

The only procedure we have left to define for our kaleidoscope is WINDMILL. The function of this procedure is to draw a chosen pattern at equally spaced angular increments around the center of the screen. You may want to experiment with different numbers of images. I have tried using six images spaced at 60-degree increments and eight images spaced at 45-degree increments. Both work fine, but other angles are worth exploring as well. The number of copies of a pattern times the angle increment must equal 360 for the pattern to be symmetrical. That is why we turn 60 degrees for six copies (6 × 60 = 360) and 45 degrees for eight copies (8 × 45 = 360).

```
TO WINDMILL :DIST :LIST
REPEAT 6 [FD :DIST PENDOWN RUN :LIST PENUP BACK :DIST
  RT 60]
END
```

Now, to generate a kaleidoscopic pattern, hide the turtle and enter

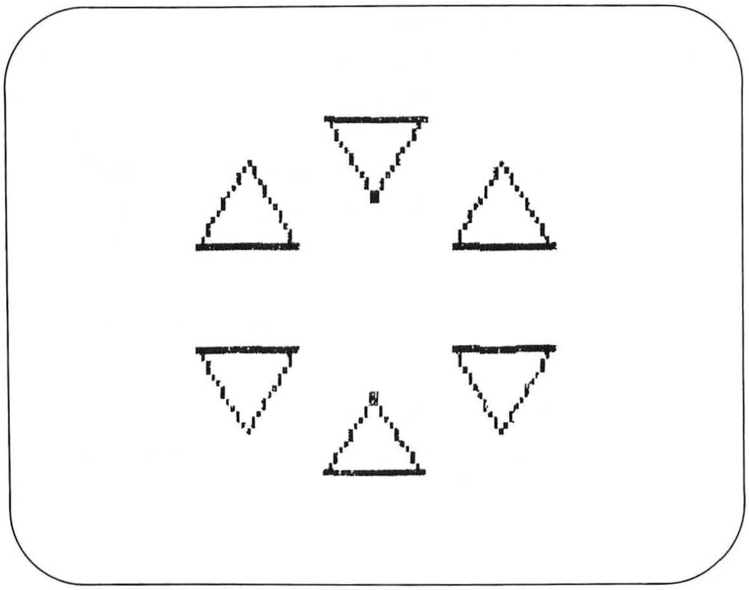
```
FS
IMAGE
```

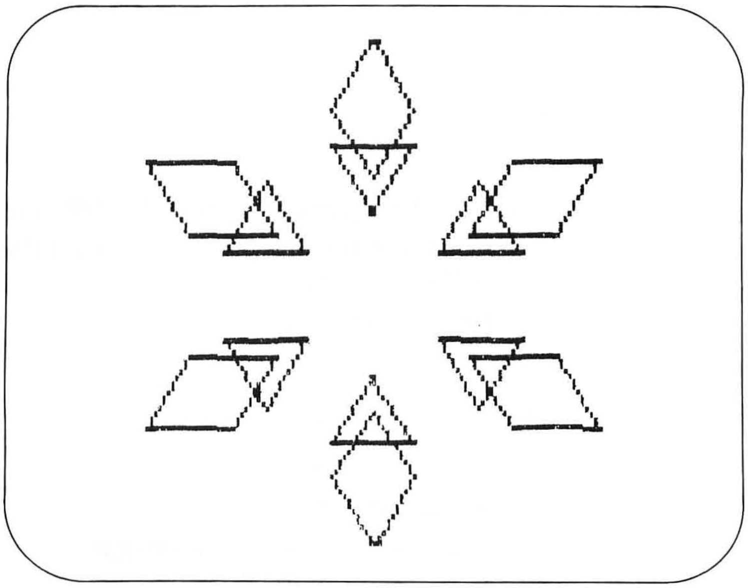
After the first pattern is drawn, press any letter key (or RETURN) to get the next pattern. When the pattern has reached the desired complexity, you may want to stop and admire your handiwork. By adding the command

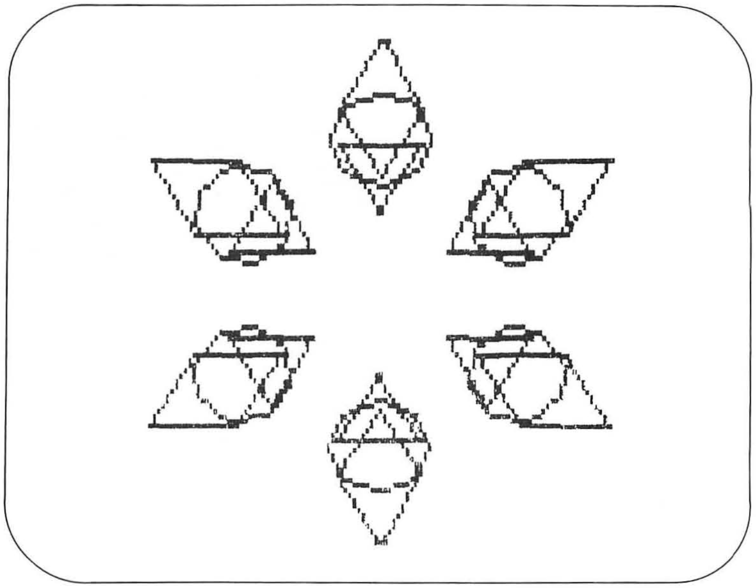
```
PRINT SE :DIST :NAME
```

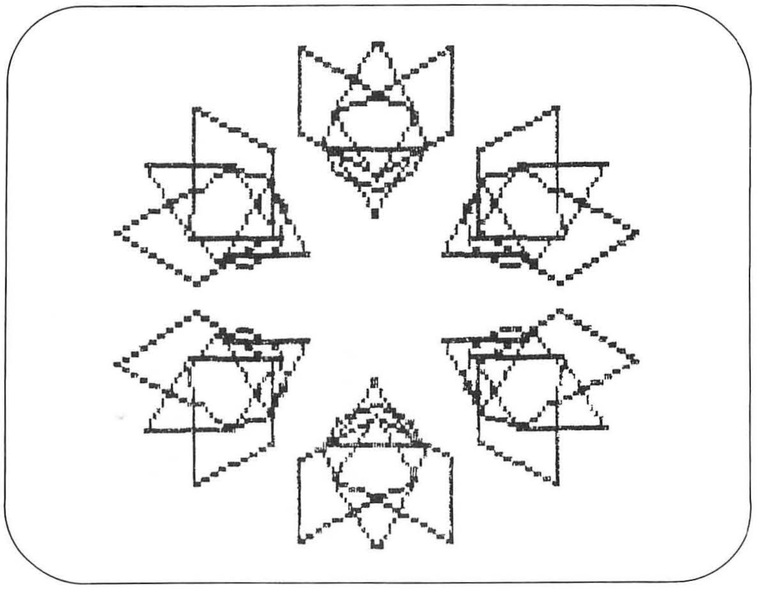
before the WINDMILL command in IMAGE you can print out the values for the inputs to WINDMILL in case you want to reconstruct the patterns later.

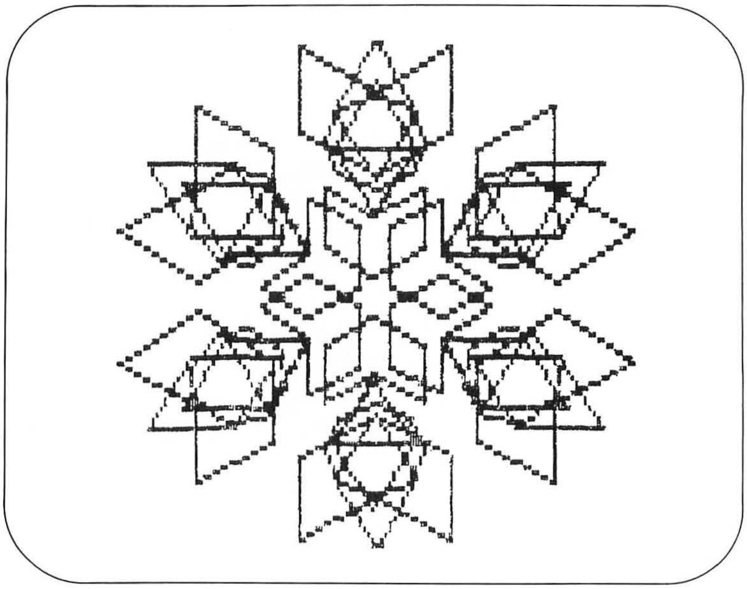
The following five figures show the successive development of one pattern.



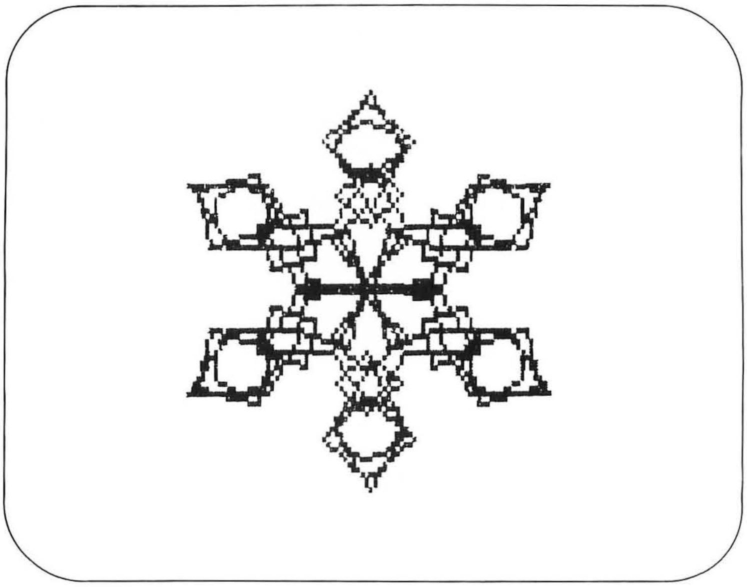


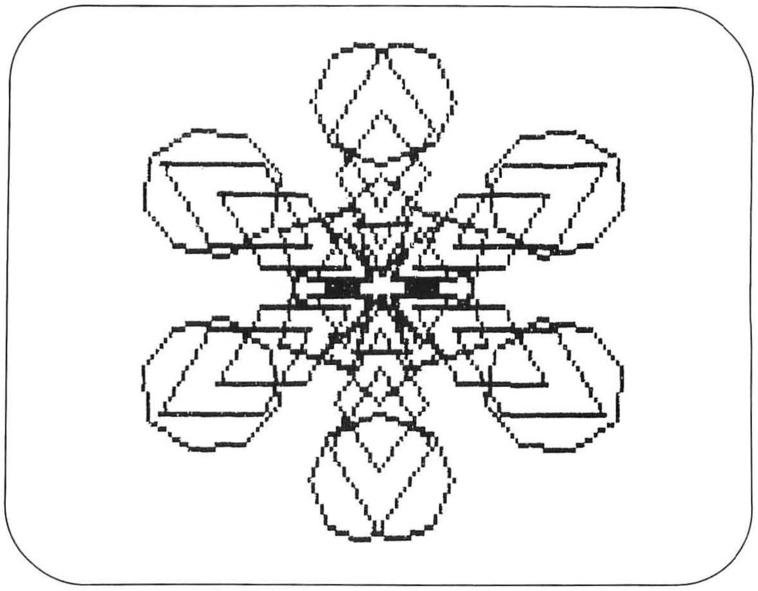


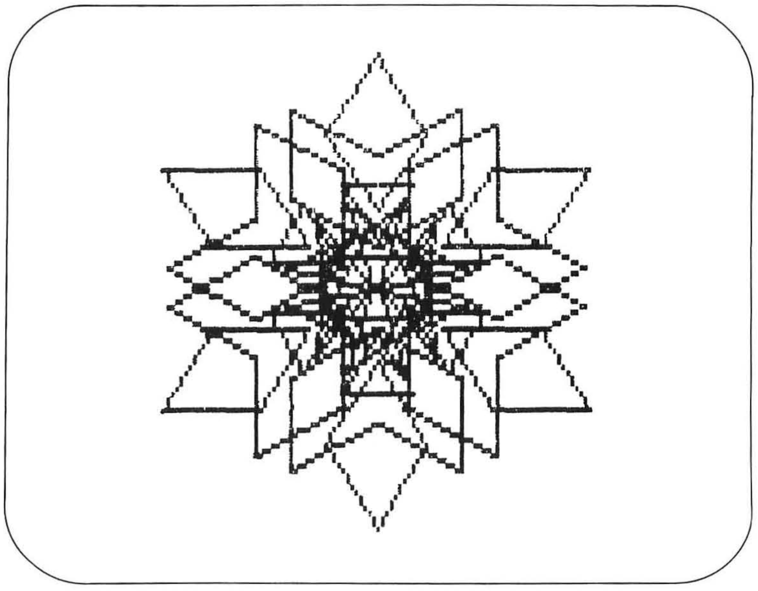


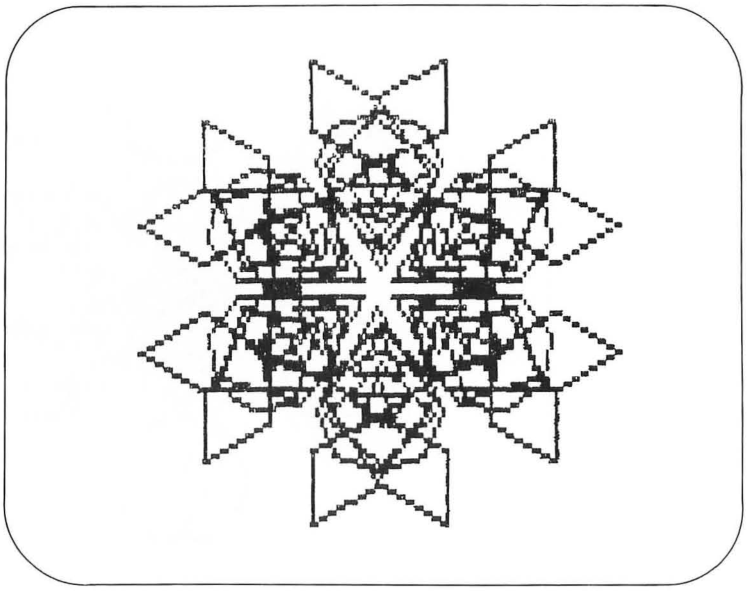


The following figures illustrate some other kaleidoscopic patterns that were generated with this set of procedures.









The patterns we have generated here resemble snowflakes. You might also want to change WINDMILL to create eight equally spaced patterns to see how you like them.

Some Projects with Polygons

You may find that polygons are among your favorite building blocks. You should be able to create many pictures using polygons. Using PENTILE and POLY, for example, make a simple BOAT procedure. Can you next create a regatta? Now use POLY to make a fanciful steam engine or some other mechanical contraption.

After you have tried a few pictures like these, use POLY (and PENTILE) to create some abstract pictures. Use many different pen colors. When you have completed your pictures, change the background colors to see which ones you like best. As always, save your pictures for later viewing.

VI. Circles, Arcs, and Stars

The preceding chapter provided a simple way to generate regular polygons. Although these polygons are valuable building blocks for graphic creations, they are insufficient for the creation of many figures. This chapter will show ways to create other geometric shapes of great power—circles, arcs, and stars.

Before proceeding with this chapter, you should erase all the procedures in your workspace except POLY and BLOSSOM.

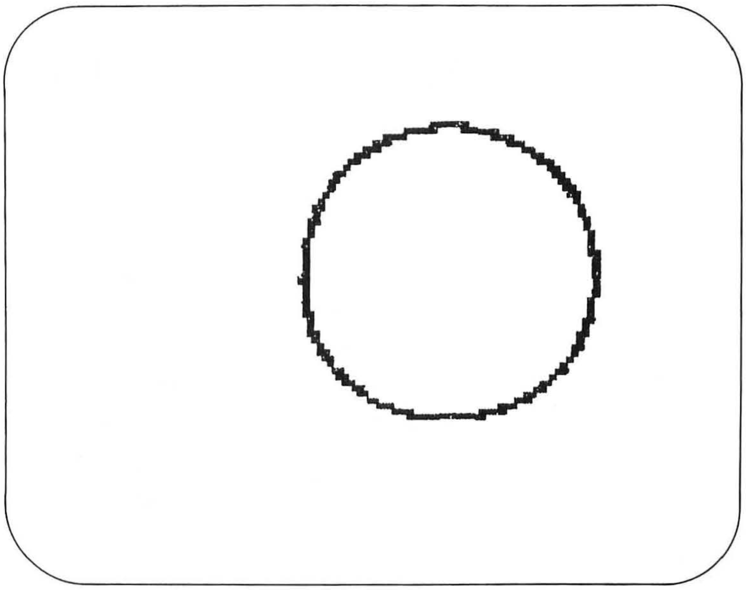
Circles

Circles are among the most powerful geometric symbols. They convey perfection, wholeness, and a sense of peace that does not exist in any other simple figure.

To have the turtle generate a circle, we must find a way to send it on a circular path. This may seem challenging, since the turtle can either turn or move in a straight line but not both simultaneously. We can create a useful approximation to a circle, however, by having the turtle take a small step and then turn a slight amount before moving again. As this process is repeated, the turtle will move in a circular path. The smallest step the turtle can take is one unit, and the smallest turning angle is one degree, so we should try this combination to see what it generates. From the Turtle Total Trip Theorem, we know that we will need 360 such steps to complete the figure.

To see our circle, enter

```
CS
REPEAT 360 [FD 1 RT 1]
```



Technically, this figure is not a circle—it is a 360-sided polygon; but the granularity of the display screen's resolution wouldn't let us generate any more detail, even if we could increase the number of sides. (*Note:* If your circle looks elliptical, your display's aspect ratio is not set properly. As was mentioned earlier, you may be able to fix this if your TV set has a vertical size control. Alternatively, you can adjust the aspect ratio with the Logo .SETSCR command.)

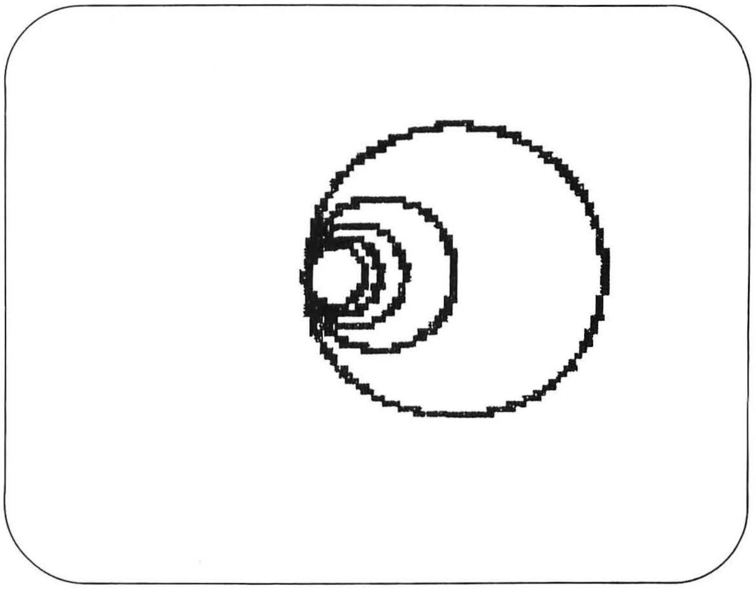
Now that we have created a circle, we should find out how to create circles of different sizes. If we increase the size of each step the turtle takes, we will get a larger circle; if we increase the angle turned at the end of each step, we will get a smaller circle. The following procedure lets us experiment with different circles:

```
TO CIRCLE :SIZE :ANGLE
REPEAT (360 / :ANGLE) [FD :SIZE RT :ANGLE]
END
```

First, let's experiment with different values of ANGLE by entering

```
CS HT
CIRCLE 1 1
CIRCLE 1 2
CIRCLE 1 3
CIRCLE 1 4
CIRCLE 1 5
```

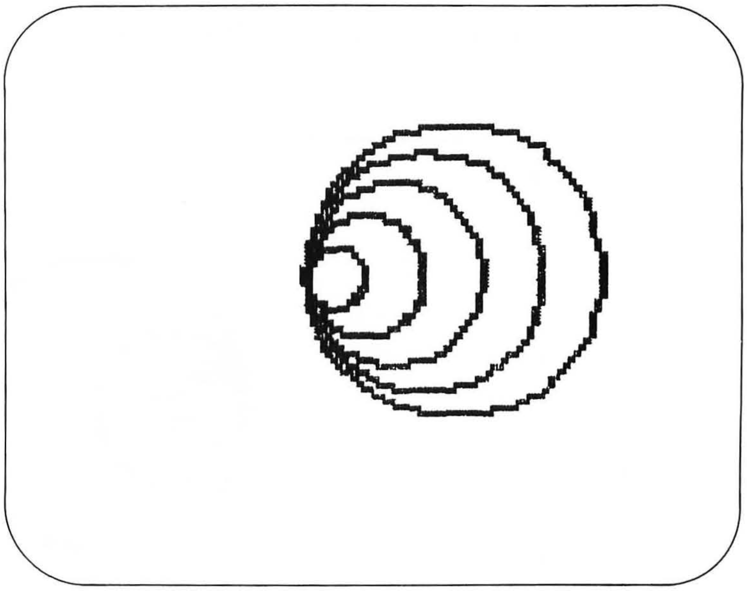
This gives us ever smaller circles.



Next, let's experiment with different values of SIZE by entering

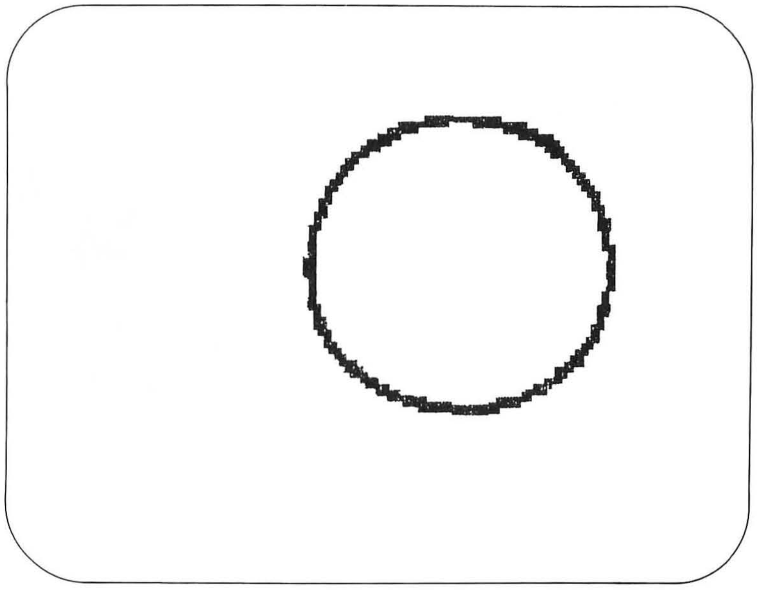
```
CS HT
CIRCLE 1 5
CIRCLE 2 5
CIRCLE 3 5
CIRCLE 4 5
CIRCLE 5 5
```

As you see, this gives us ever larger circles.



Next, let's change SIZE and ANGLE at the same time to see what that gives us. Enter

```
CS HT
CIRCLE 1 1
CIRCLE 2 2
CIRCLE 3 3
CIRCLE 4 4
CIRCLE 5 5
```

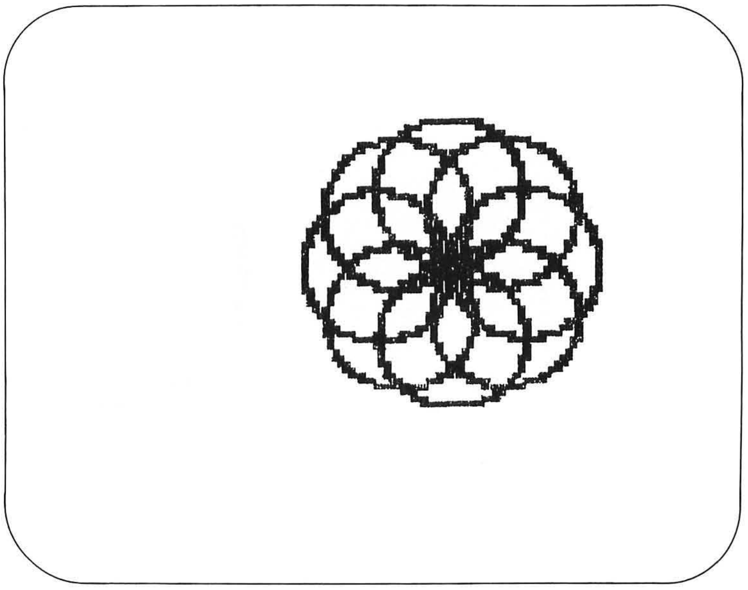


All these circles are nearly the same size. The difference in their shapes comes from the fact that we are really drawing polygons with different numbers of sides.

As you experiment with CIRCLE, you will find values of SIZE and ANGLE that let you create any size circle you want.

Next, try

```
CS
BLOSSOM [CIRCLE 1 2]
```



Now that we know how to make circles, we should learn how to make parts of circles.

Arcs

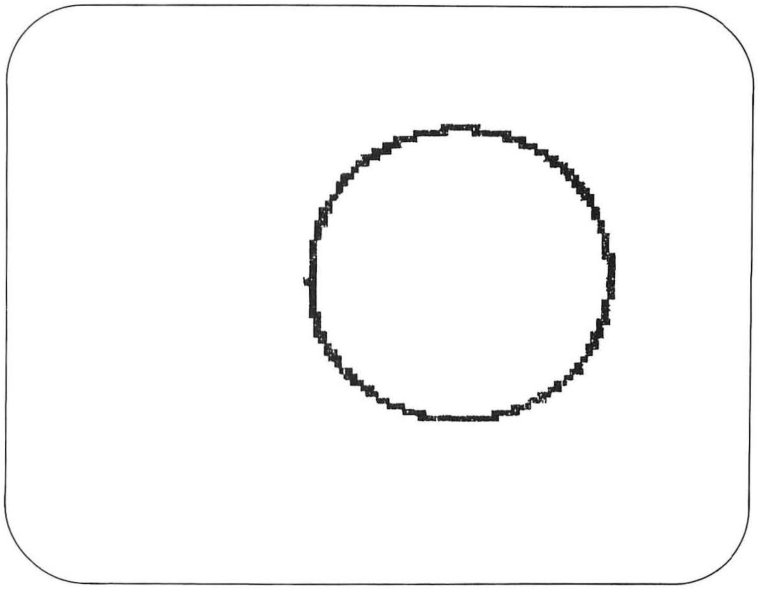
If we want to draw only part of a circle, we must modify the circle procedure so that it can be stopped before we have turned 360 degrees. If we do travel 360 degrees, the turtle will trace a closed path.

Let's experiment by using the following procedure:

```
TO ARC :SIZE :ANGLE :AMOUNT
REPEAT (:AMOUNT / :ANGLE) [FD :SIZE RT :ANGLE]
END
```

We can test this ARC procedure by creating a circle from three 120-degree arcs. Enter

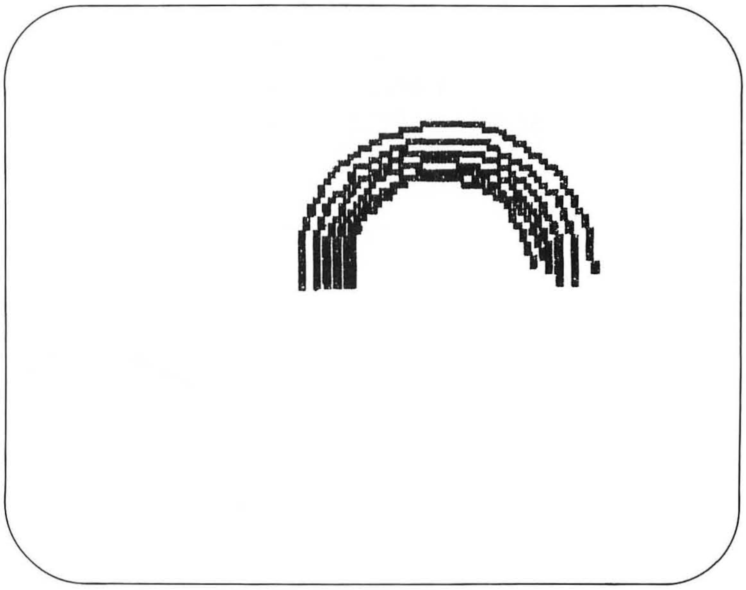
```
CS
SETPN 0 ARC 1 1 120
SETPN 1 ARC 1 1 120
SETPN 2 ARC 1 1 120
```



By adjusting the values for SIZE, ANGLE, and AMOUNT, you will be able to create arcs that are long and graceful as well as arcs that are small and tightly turned.

As an experiment, create a rainbow using arcs. Here is one procedure that will do it for us:

```
TO RAINBOW
CS
PU SETPN 0 SETPOS [-80 0] PD SETH 0 ARC 8 8 185
PU SETPN 1 SETPOS [-74 0] PD SETH 0 ARC 8 9 185
PU SETPN 2 SETPOS [-70 0] PD SETH 0 ARC 8 10 185
PU SETPN 0 SETPOS [-66 0] PD SETH 0 ARC 8 11 185
PU SETPN 1 SETPOS [-62 0] PD SETH 0 ARC 8 12 185
PU SETPN 2 SETPOS [-60 0] PD SETH 0 ARC 8 13 185
END
```

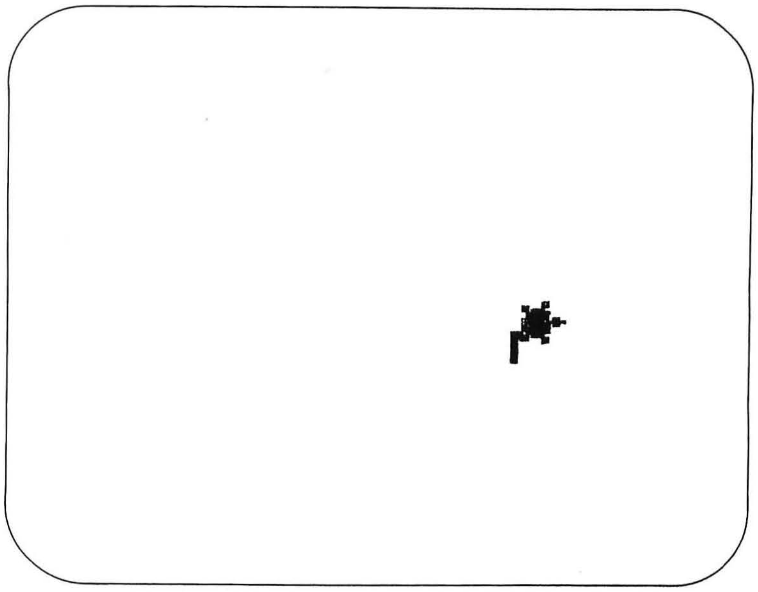


A Flower

Using the procedures we have created, we can draw a pretty picture of a flower. Since we know how to use BLOSSOM for the flower itself, and we know that ARC can be used to create the stem, let's start with a procedure for creating a leaf.

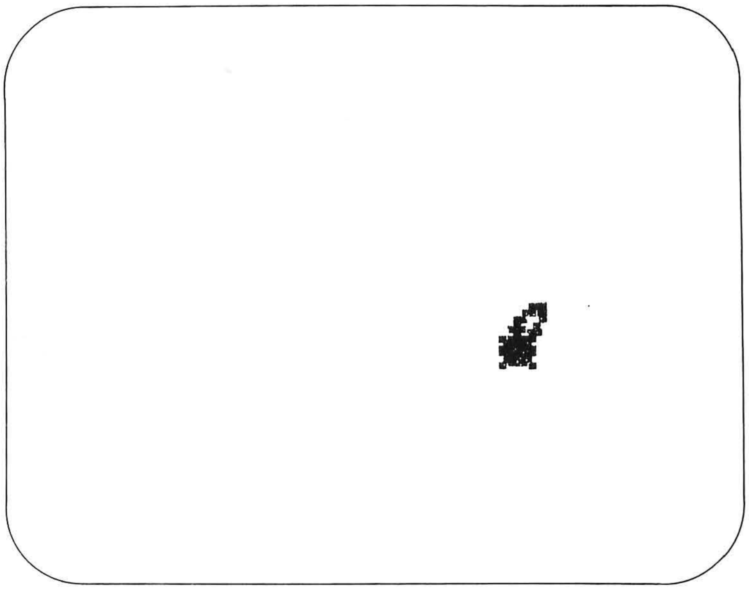
A leaf can be made from two small arcs. For example, enter

```
CS ST SETPN 0
ARC 2 9 90
```



This draws half of the leaf. To draw the other half, we must first turn the turtle, or we will draw a semicircle instead. Since the turtle needs to turn by 360 degrees overall, and since the two arcs for the leaf sides account for 180 degrees, we must turn an additional 180 degrees. We want to have the turtle back at its starting position when we are done, so we should turn the turtle by 90 degrees after each use of ARC. To try this, enter

```
RT 90
ARC 2 9 90
RT 90
```



This generates a nice leaf for us. We can now define the LEAF procedure as follows:

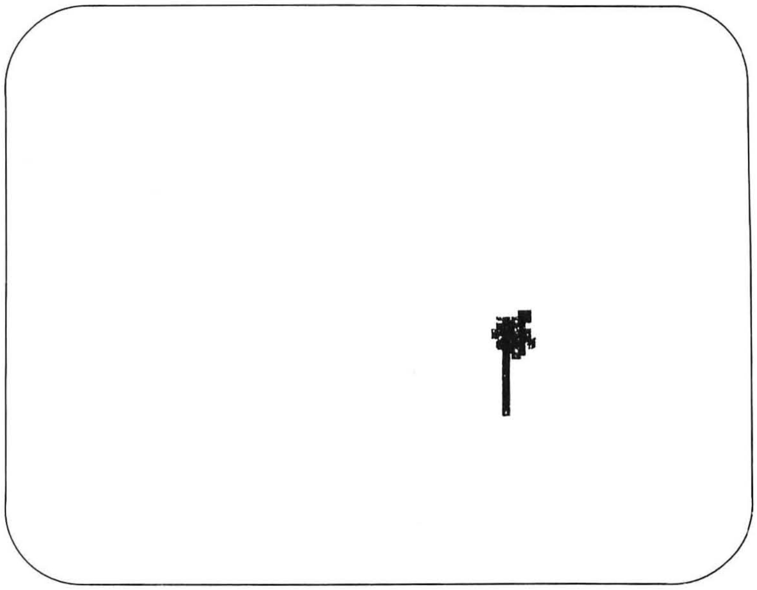
```
TO LEAF
REPEAT 2 [ARC 2 9 90 RT 90]
END
```

Next, we need a procedure that draws a portion of the stem. To draw the stem, let's use a gradual arc. For the STEM procedure, enter

```
TO STEM
ARC 3 1 10
END
```

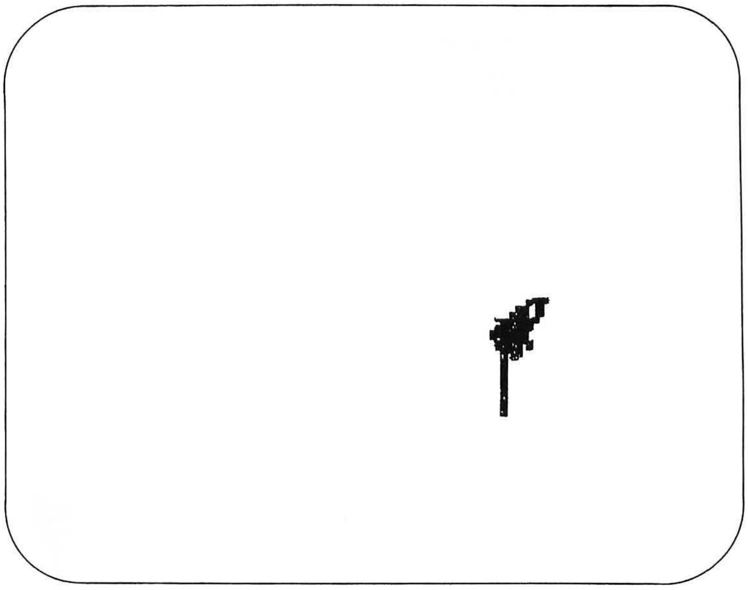
Now, to draw our flower, let's start by drawing the stem. Enter

```
CS
SETPN 0
STEM
```



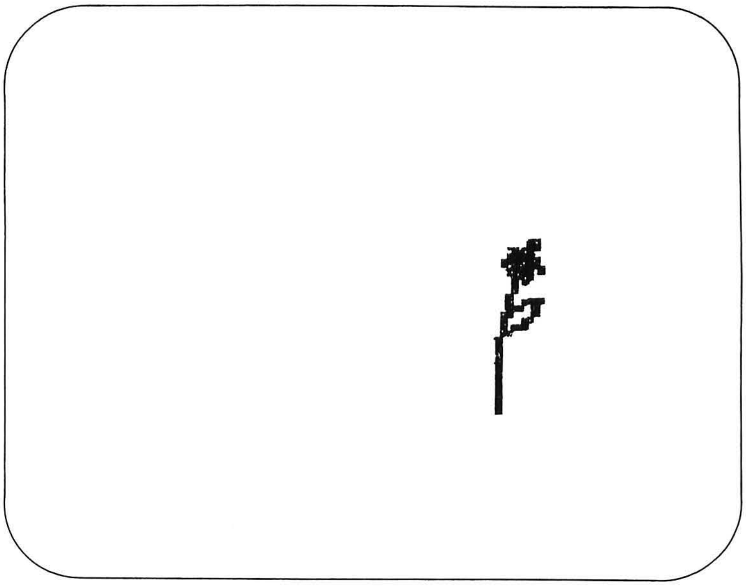
Next, let's draw a leaf by entering

```
LEAF
```



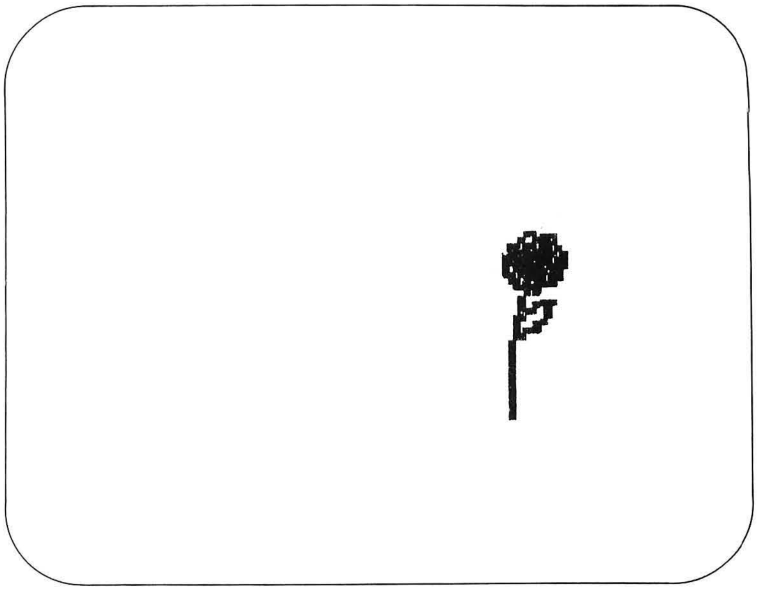
We can extend the stem some more by entering

```
STEM
```



Now we can make the blossom of our flower by entering

```
BLOSSOM [POLY 6 6]
```



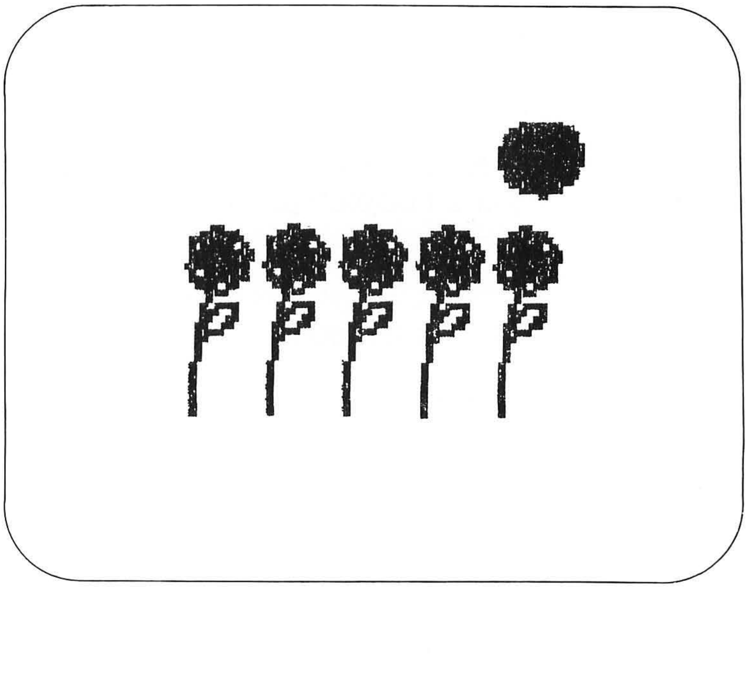
Now that all the pieces work together properly, we can create a FLOWER procedure that lets us draw flowers of different colors:

```
TO FLOWER :COLOR
HT
SETPN 1
STEM
LEAF
SETPN :COLOR
STEM
BLOSSOM [POLY 6 6]
END
```

When you start using FLOWER, you will notice that the turtle does not end at its beginning position. Keep this in mind as you use this procedure.

Let's make a picture using FLOWER. The following PICTURE procedure will draw a small row of flowers and put a yellow sun in the sky:

```
TO PICTURE
CS HT
PU SETPOS [-75 -40] SETH 0 PD FLOWER 0
PU SETPOS [-45 -40] SETH 0 PD FLOWER 1
PU SETPOS [-15 -40] SETH 0 PD FLOWER 2
PU SETPOS [15 -40] SETH 0 PD FLOWER 0
PU SETPOS [45 -40] SETH 0 PD FLOWER 1
PU SETPOS [60 60] PD SETPN 0
REPEAT 90 [FD 15 BK 15 RT 4]
END
```



The last command you typed gave us a yellow sun. If we had wanted a night scene, we could have darkened the background and put stars in the sky instead.

Stars

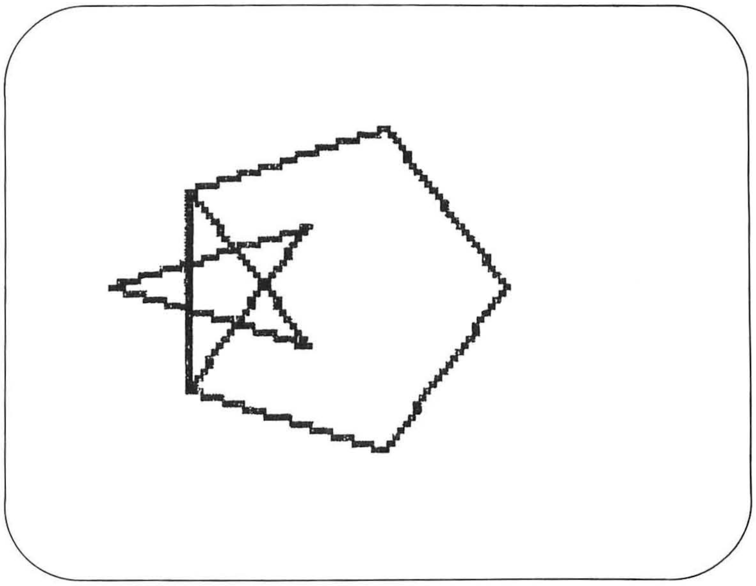
We already know that a five-pointed star can be drawn with a command such as

```
REPEAT 5 [FD 80 RT 144]
```

In this section, we will explore this and other stars, and we will learn how stars are related to the simple polygons we created in the last chapter.

For starters, let's compare the five-pointed star with a pentagon. Enter

```
CS
REPEAT 5 [FD 80 RT 144]
REPEAT 5 [FD 80 RT 72]
```



The only difference between the commands for these two figures is the turning angle. You might have noticed that 144 is twice 72. This is an interesting result, suggesting a relationship between stars and simple polygons. As we will see, however, not all polygons can be turned into stars by doubling the turning angle. Also, some polygons can be turned into different stars by doubling or tripling the turning angle or increasing it by other multipliers.

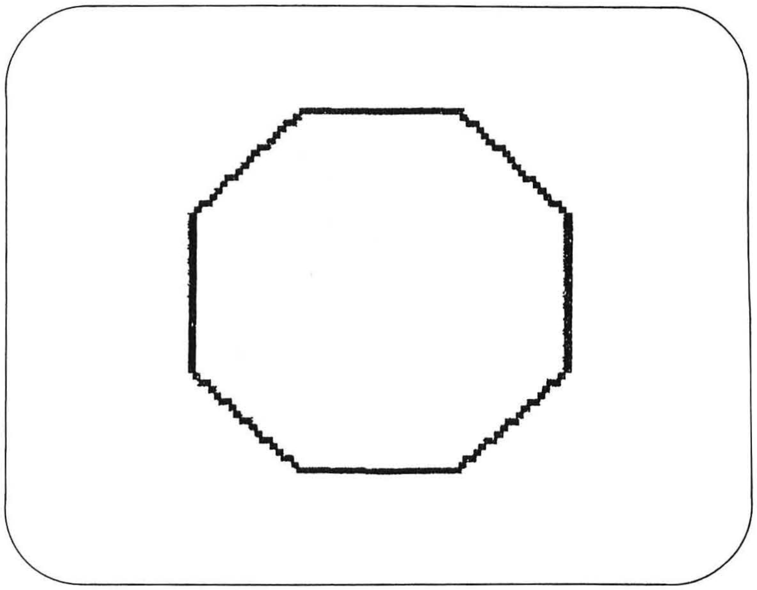
We can create a generalized star procedure that is similar in structure to POLY by entering

```
TO STAR :SIDES :MULT :SIZE
REPEAT :SIDES [FD :SIZE RT (:MULT * 360 / :SIDES)]
END
```

Note that Logo uses the asterisk (*) as the symbol for multiplication.

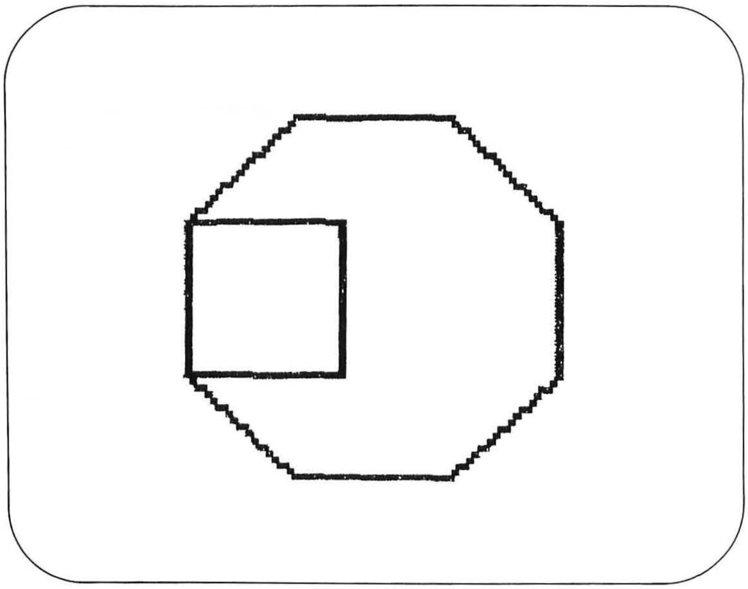
Now that you have the STAR procedure, let's test it. Enter

```
CS
STAR 8 1 60
```



This generated an octagon—the expected result for a multiplier of one. Next, enter

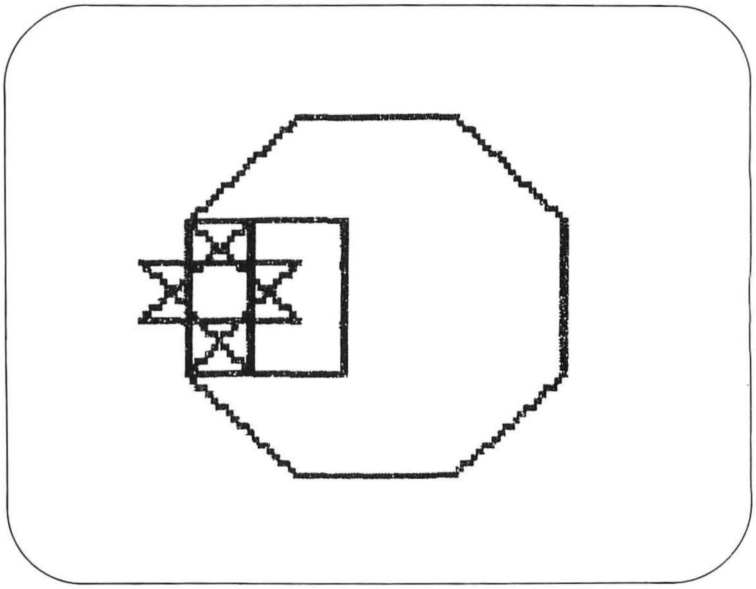
```
STAR 8 2 60
```



This added a square to the figure! The square results from a turning angle of 2 × 360/8, or 90 degrees. If we now enter

```
STAR 8 3 60
```

We finally get an eight-pointed star!



You should experiment with different values for SIDES and MULT to see which combinations give stars and which do not.

Some Projects

Now that circles, arcs, and stars have been added to your graphics tool kit, you should be ready to create some very interesting pictures. As a starter project, try to modify PICTURE to generate a night scene instead of the day scene we created. To do this, change the background to black and replace the sun with small stars in the sky.

Try using different star patterns with BLOSSOM. Are these more interesting than the patterns BLOSSOM creates with simple polygons?

Can you create a diamond that fits inside a circle so that its corners just touch the circle's boundary?

Using your knowledge of lines, polygons, circles, and arcs, can you create the signs of the zodiac?

Using the counter technique described in Chapter IV, create a growing star on the screen. As the figure is being drawn, do you sense the illusion of motion? As always, save your artwork for later viewing.

VII. Squirals and Spirals

Thus far, we have explored graphic procedures that produce static images. Their finished form offers no clue to the process that created them. In this chapter, we will explore some figures that are formed by a sequential growth process. As with polygons, we will be dealing with a set of commands that are repeated to form the object. By increasing the size of the lines drawn with each repetition of the command sequence, an object is caused to grow on the display screen.

Squirals

Living things sometimes leave traces of their growth patterns that can be studied without watching the object grow. Seasonal cycles, for example, produce a series of concentric rings in trees. By counting the rings, we can deduce the age of a tree. A more common growth pattern, found in both plants and animals, is the spiral. The effect of spirals on the eye is so strong that they almost appear to be in motion.

The first spiral figures we will explore are made with straight lines. These spirals are called *squirals* (from *square spirals*). Before entering the procedures used in this chapter, you should clear your workspace.

To start, let's examine a spiral made from straight lines and square (90-degree) corners. To draw a square, we draw the same length line after each turn of 90 degrees. To draw a square spiral, we need to increase the length of each side over its previous value. The following procedure will allow us to experiment with squirals containing various turning angles. The increment by which each side grows is chosen to be 2, although you may want to change this value as you experiment.

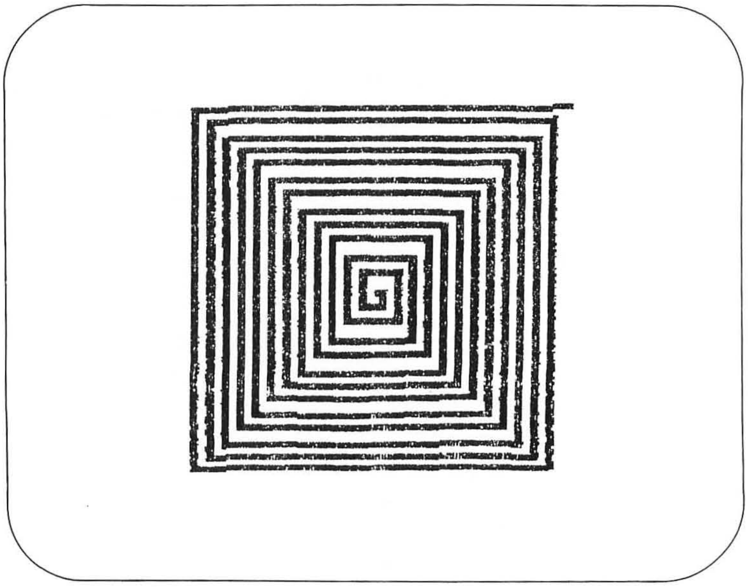
To create the SQUIRAL procedure, enter

```
TO SQUIRAL :ANGLE :STEPS
MAKE "SIDE 0
REPEAT :STEPS [FD :SIDE RT :ANGLE MAKE "SIDE
  :SIDE + 2]
END
```

Next, enter

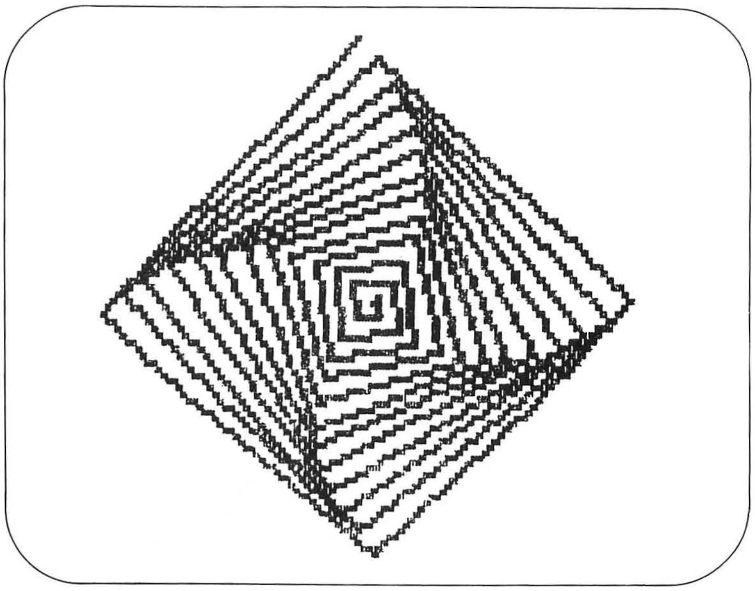
```
CS HT
SQUIRAL 90 50
```

This will generate a squiral pattern on the display screen.



On examination, we can see how closely this figure relates to the square. Next, suppose we were to use an angle close to 90 degrees—89 degrees, for example. Before trying this angle, try to visualize the result in your mind's eye. Will the change be small or large? Once you have made your decision, enter

```
CS
SQUIRAL 89 50
```



This figure looks very different from the figure generated by SQUIRAL 90 50. Why is this?

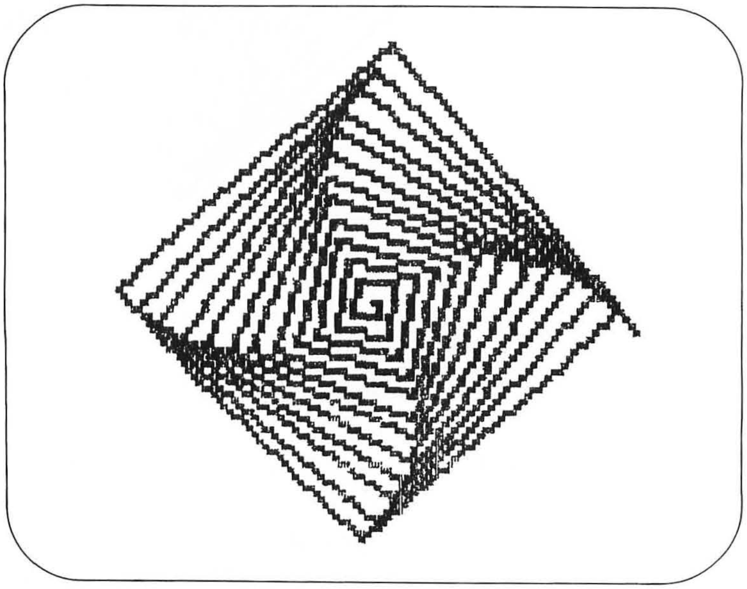
As the figure for SQUIRAL 89 50 is drawn, each turning angle differs from that for a square by only 1 degree, but the impact of this difference is cumulative so far as the overall figure is concerned. By the time the turtle has made one circuit around the path, the difference is 4 degrees. By the time the procedure stops, the turning angle is 100 degrees in variance from its value for the truly square squiral.

One of the attractions of this particular figure is the four arching branches twisting to the left. These branches are formed as an interference pattern of corners that bump into each other because of the angular mismatch (compared to a square). Such interferences, called moiré patterns, are quite common. You can see moiré patterns by holding two window screens together and rotating one of them slightly with respect to the other. When the screens are adjusted so that there is no interference pattern, the screens are perfectly aligned.

We can see a similar effect for squirals. For example, if we enter

```
CS
SQUIRAL 91 50
```

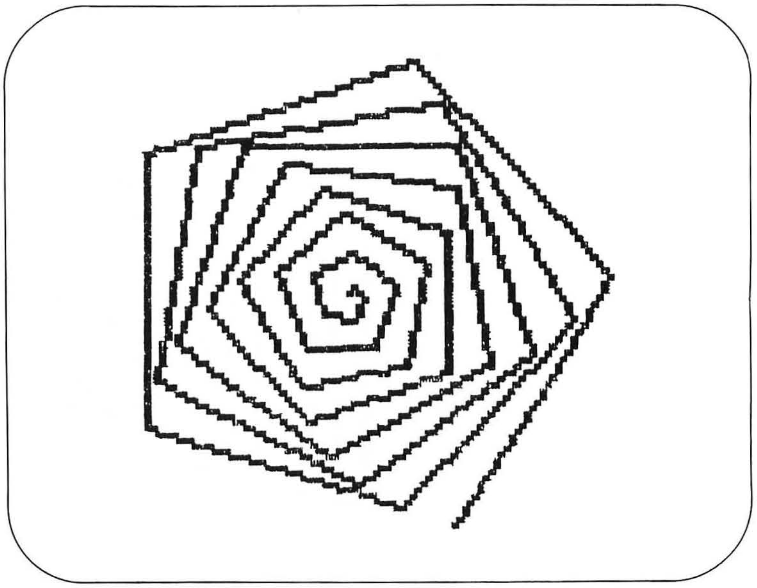
we will generate a squiral pattern that has arms branching off to the right.

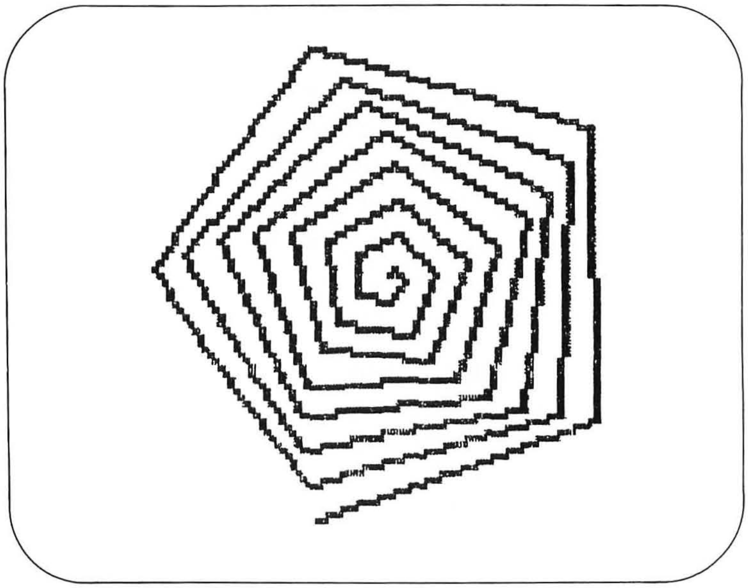


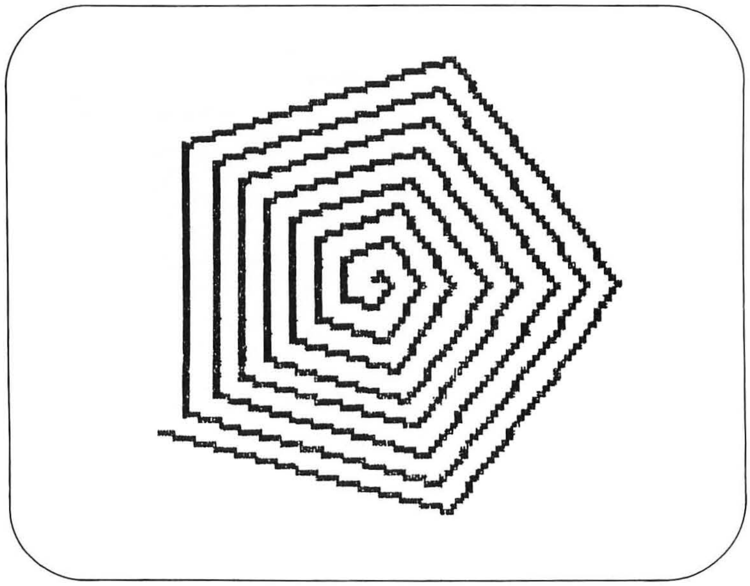
Whenever we see interferences of this type, it is a clue that we are close to a regular pattern whose alignment of sides creates no interference. If the squirals arch to the left, the angle is too small; if they arch to the right, the angle is too large.

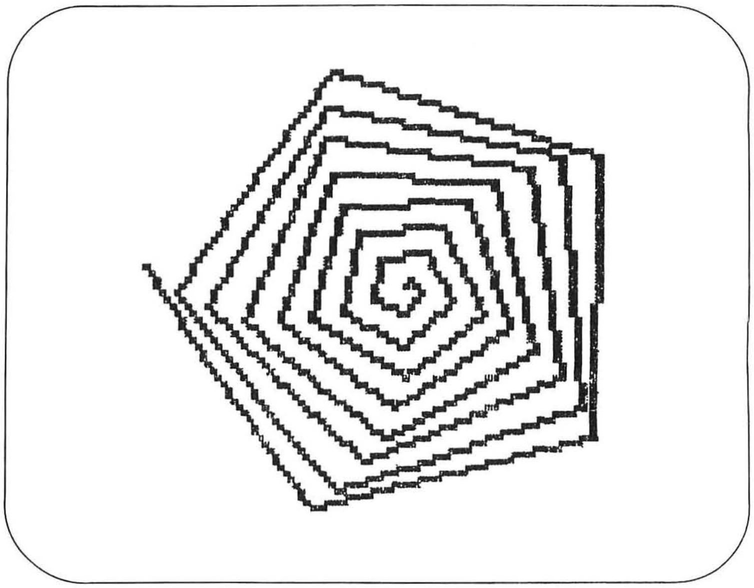
How many squirals are there with no interference patterns? The following figures are examples of squiral patterns based on the pentagon. (You will want to clear the screen before drawing each squiral.)

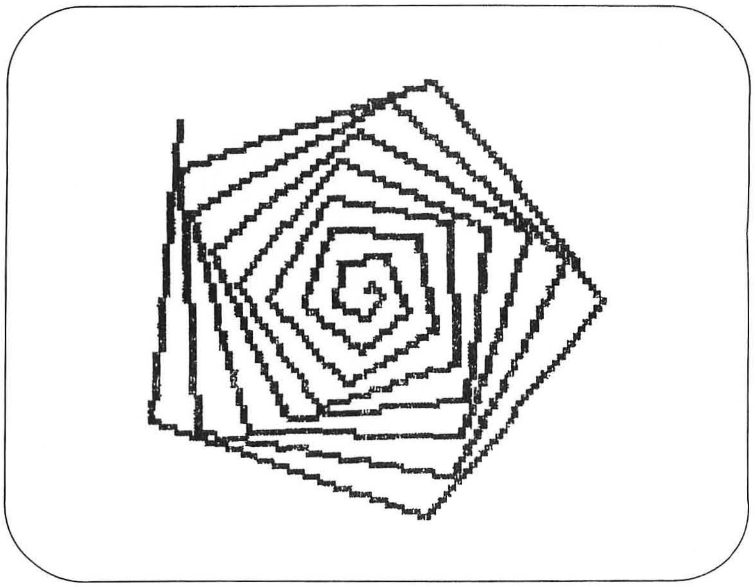
```
SQUIRAL 70 40
SQUIRAL 71 40
SQUIRAL 72 40
SQUIRAL 73 40
SQUIRAL 74 40
```







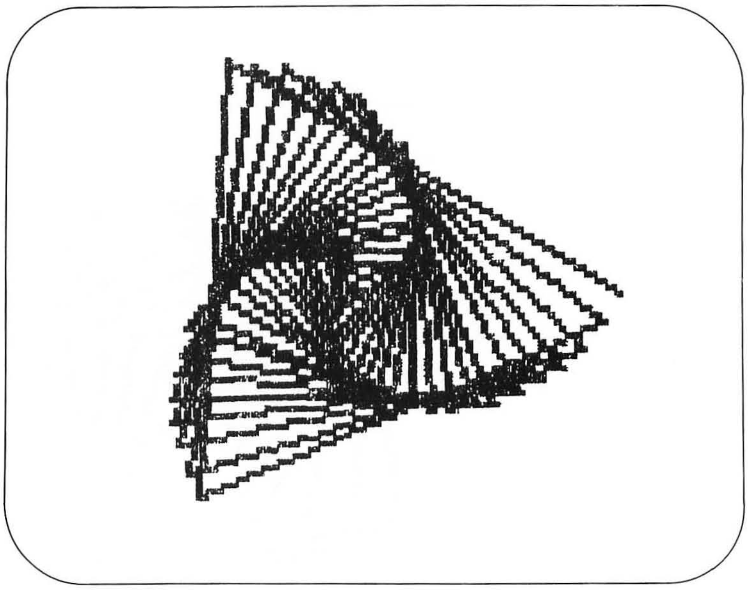


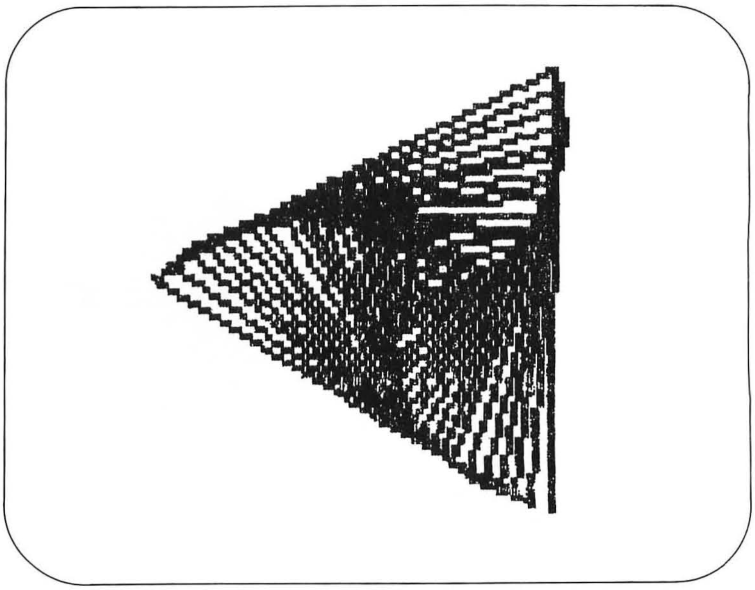


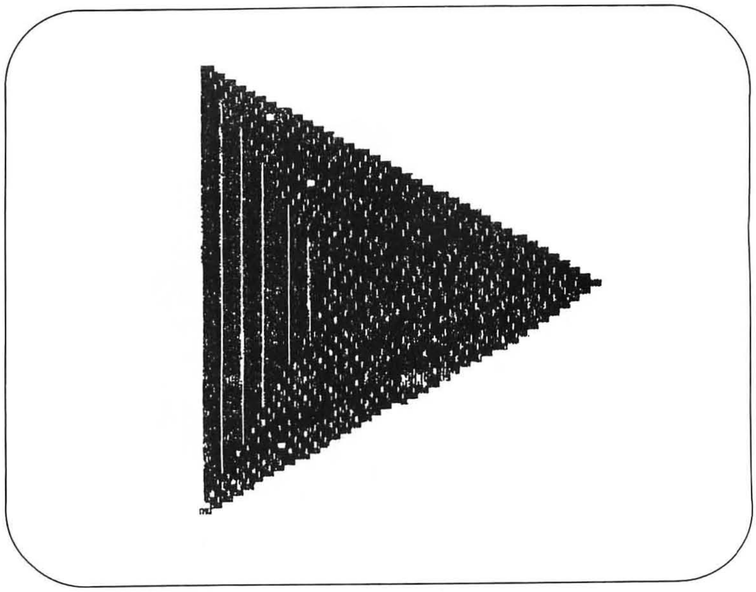
Once again, we can see the spiral arms in all patterns except SQUIRAL 72 40. The 72-degree angle is the angle associated with a regular pentagon. Notice that the curvature of the spiral arms is greater as you move farther away from 72 degrees in either direction.

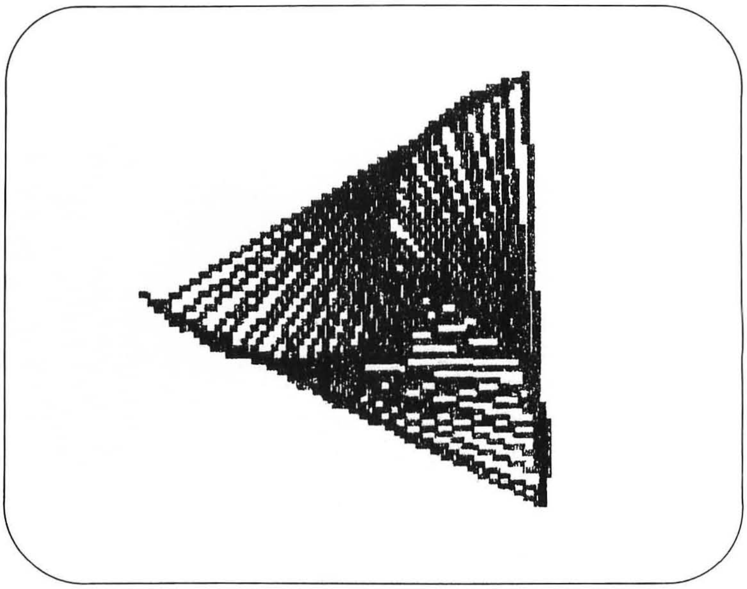
The next set of figures explores squirals in the vicinity of 120 degrees:

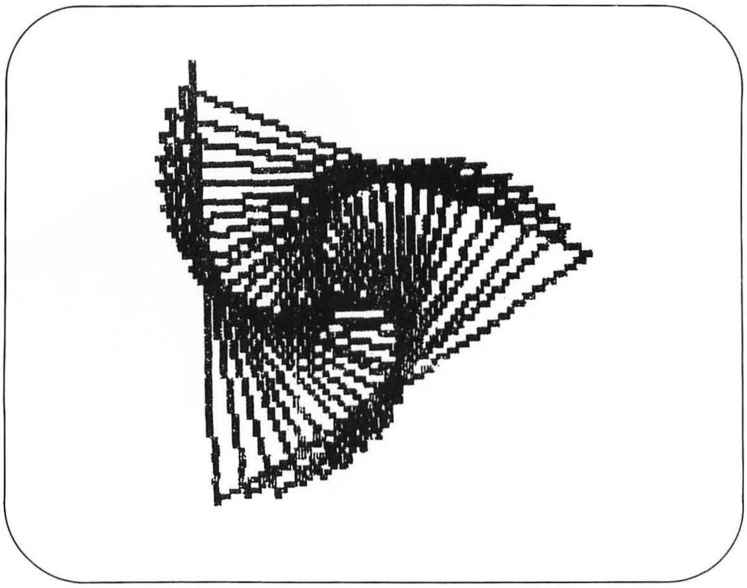
```
SQUIRAL 118 60
SQUIRAL 119 60
SQUIRAL 120 60
SQUIRAL 121 60
SQUIRAL 122 60
```





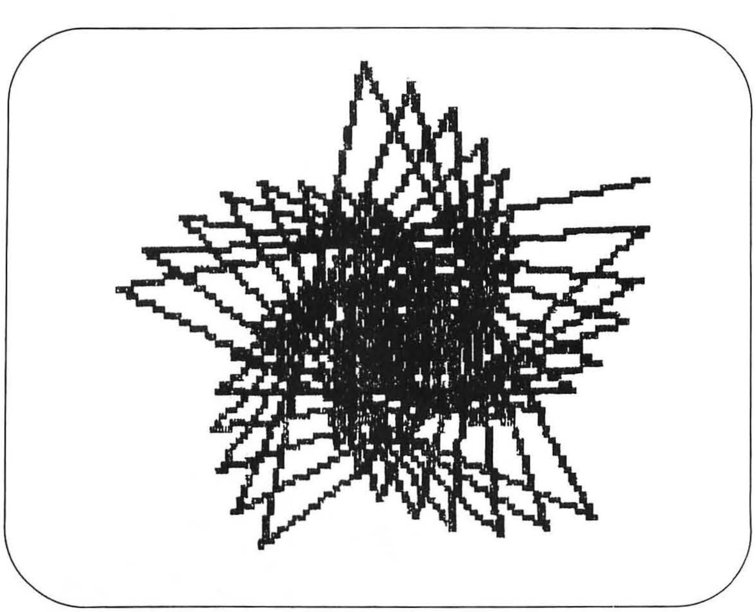


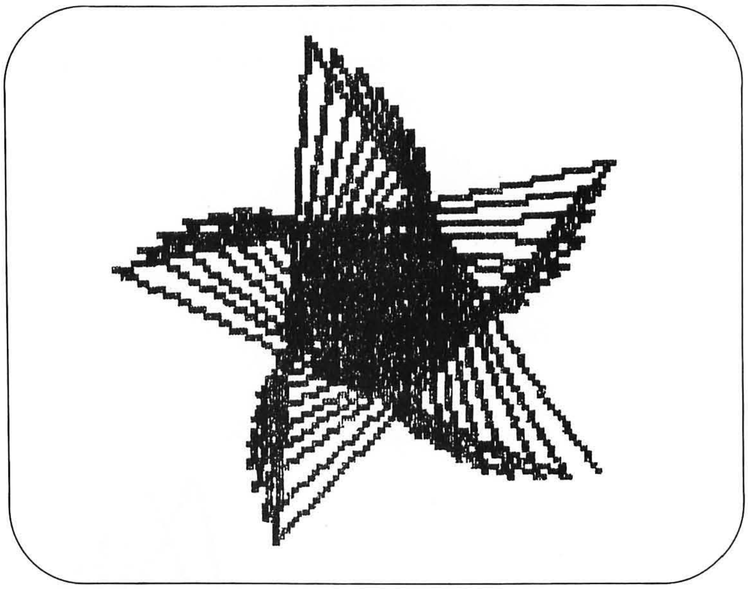


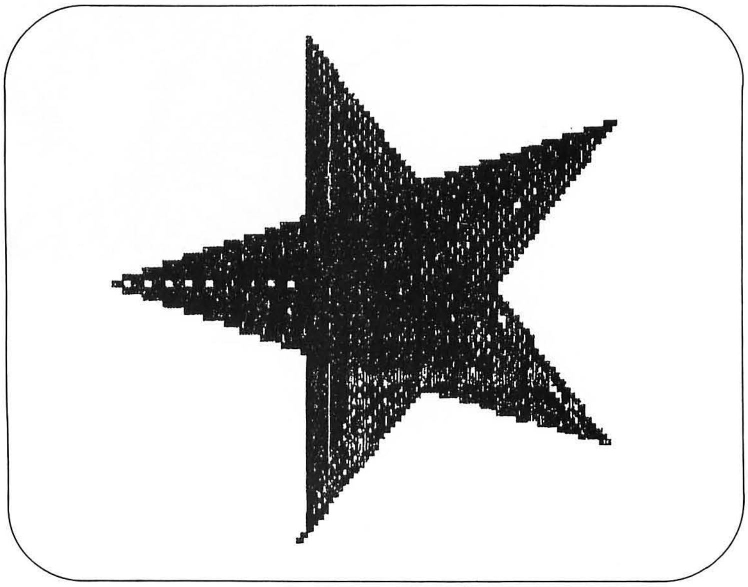


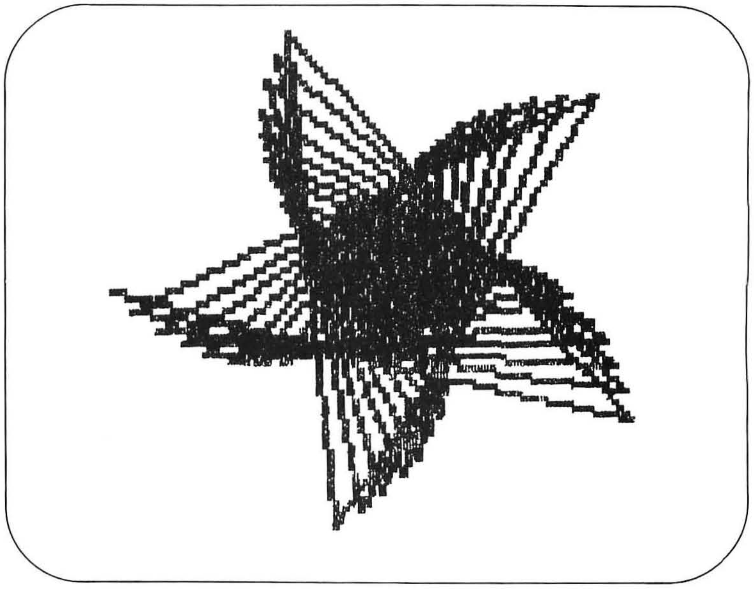
Squiral patterns need not be based on simple polygons. The following attractive figures, for example, occur in the vicinity of 144 degrees:

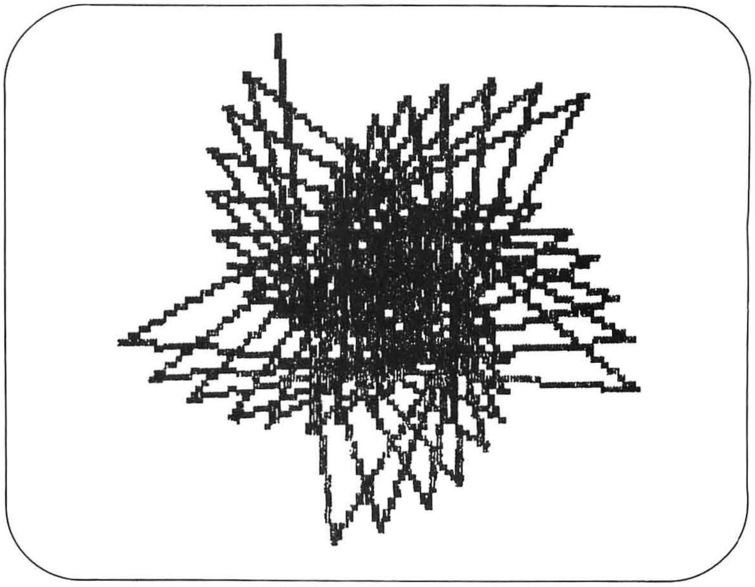
```
SQUIRAL 142 70
SQUIRAL 143 70
SQUIRAL 144 70
SQUIRAL 145 70
SQUIRAL 146 70
```











As you experiment with the SQUIRAL procedure, you will find many interesting patterns. Each of these patterns reflects the process of growth by which it was created.

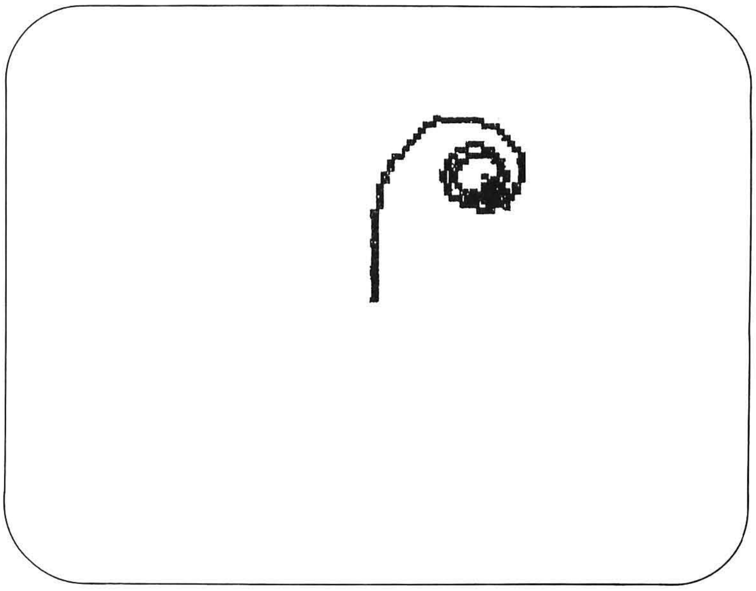
Closed Spirals

In the preceding section, we created squirals by keeping the turning angle fixed and increasing the size of the drawn lines. By reversing this sequence, we can create some interesting spirals. To make a spiral curve, we can draw a series of fixed-length lines and turn the turtle by increasing amounts at the end of each step.

Rather than starting with a spiral procedure, we will create a spiral interactively by using primitive Logo commands. As an example, enter

```
CS ST
MAKE "ANGLE 0
REPEAT 45 [FD 6 RT :ANGLE MAKE "ANGLE :ANGLE + 1]
```

This set of instructions starts the turtle off on a gentle arc to the right that begins to circle in on itself quickly.



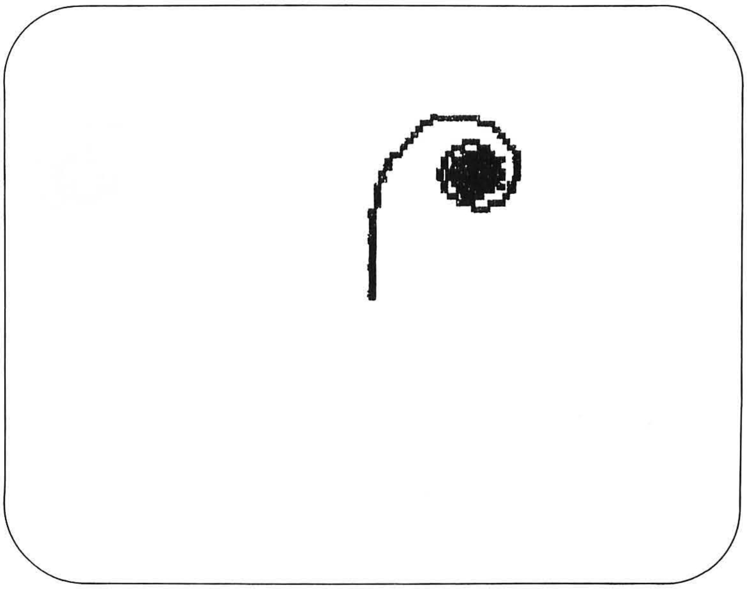
Next, enter

```
REPEAT 45 [FD 6 RT :ANGLE MAKE "ANGLE :ANGLE + 1]
```

This continues to tighten the spiral. If we take an additional 90 steps (making 180 in total) by entering

```
REPEAT 90 [FD 6 RT :ANGLE MAKE "ANGLE :ANGLE + 1]
```

we will bring the turtle to the center of the circular area that is forming on the screen.

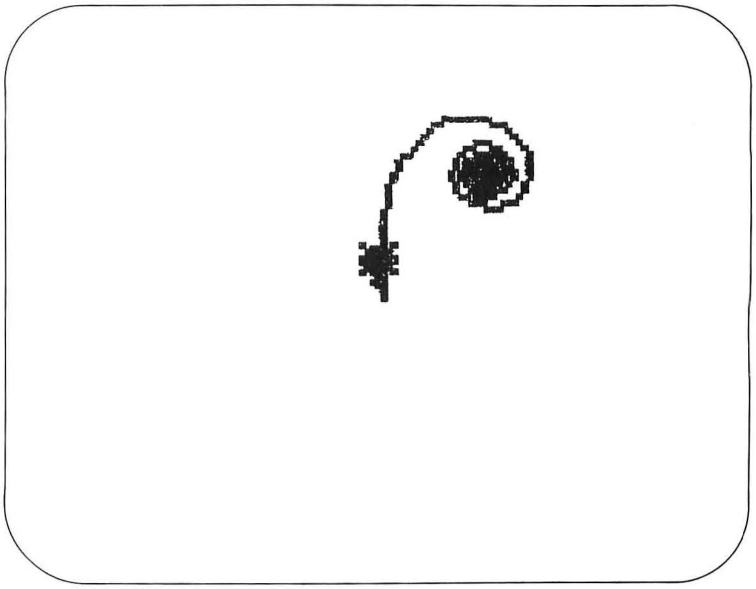


What will happen for angles greater than 180 degrees? When we started, the turtle turned first by 1 degree, then by 2 degrees, and so on. After the first four steps, the turtle had turned by only 10 degrees. Contrast this with the turning that took place at the four steps starting at 89 degrees. After these four steps, the turtle had turned over 360 degrees. By the time we reached 180-degree increments, the turtle was simply moving back and forth over its position.

To see what happens for increments greater than 180 degrees, enter

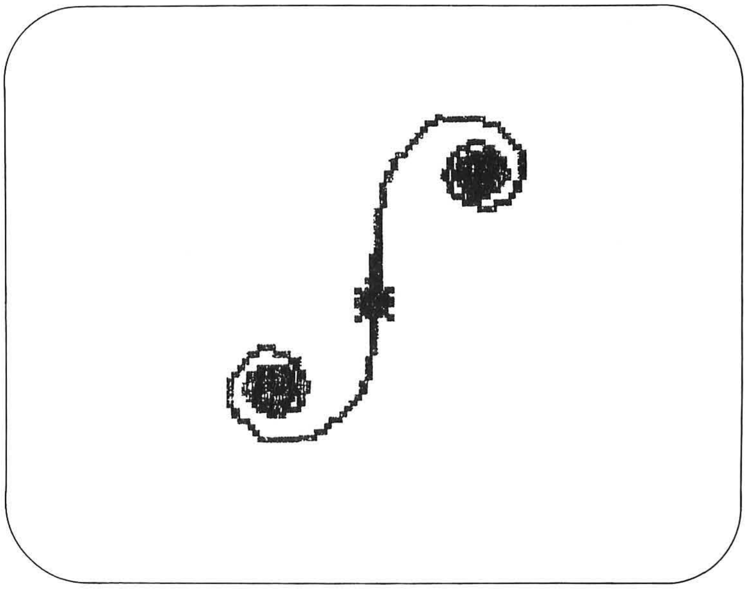
```
REPEAT 180 [FD 6 RT :ANGLE MAKE "ANGLE :ANGLE + 1]
```

At the end of 360 steps, the turtle is back at its origin, having retraced its steps. The turtle is pointing down, however, instead of up.



To complete the figure (and return the turtle to its home position), we need to take an additional 360 steps:

```
REPEAT 360 [FD 6 RT :ANGLE MAKE "ANGLE :ANGLE + 1]
```



This spiral figure was made as a result of taking 720 steps. The first 360 steps created the upper right arm of the spiral, and the second 360 steps created the lower left arm. If we were to continue repeating these same commands, we would retrace the original figure.

This type of spiral is but one member of a large family of such curves. A procedure that lets us create this and other examples of closed spirals is as follows:

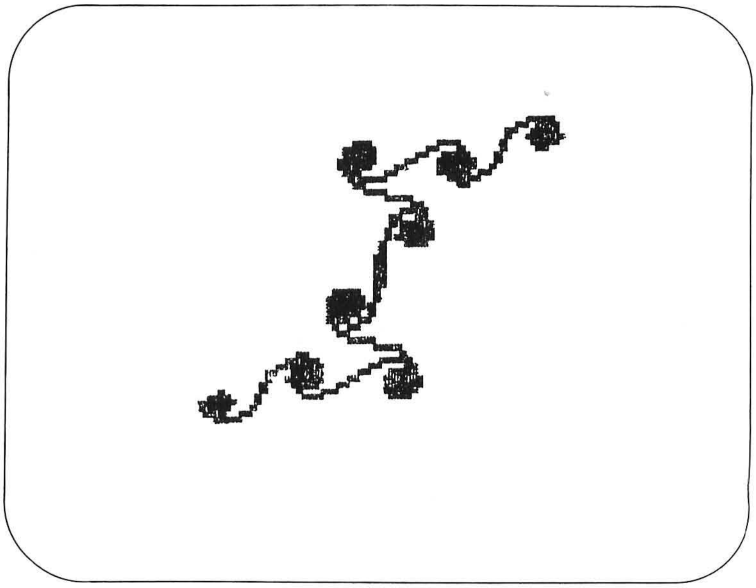
```
TO CLOSESPI :SIZE :ANGLE :INCREMENT
REPEAT 720 [FD :SIZE RT :ANGLE MAKE "ANGLE :ANGLE +
  :INCREMENT]
END
```

This CLOSESPI procedure lets us create figures with different angle increments and with different starting angles. Our previous spiral can be drawn by entering

```
CS
CLOSESPI 6 0 1
```

Instead of increasing the turning angle in 1-degree steps, suppose that we choose another value—for example, 7 degrees. On first thought, we might expect to get a smaller version of the existing spiral, since we have changed only the increment by which the angle is changed. To see what happens, enter

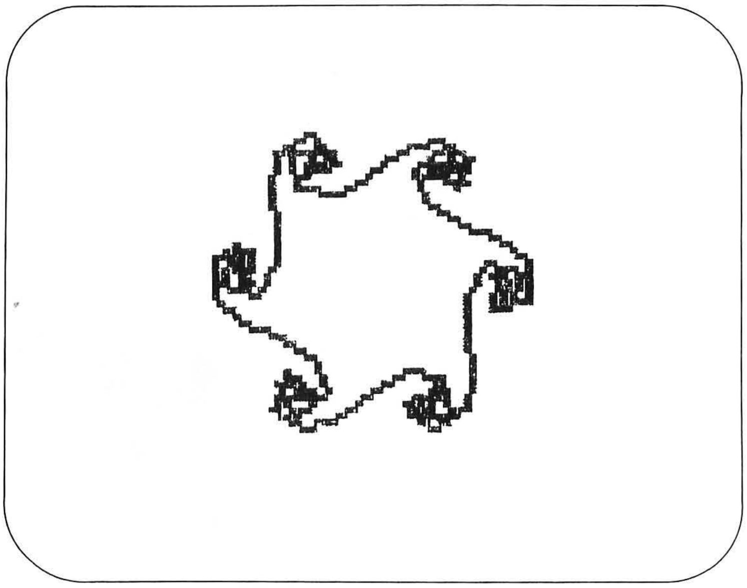
```
CS
CLOSESPI 5 0 7
```

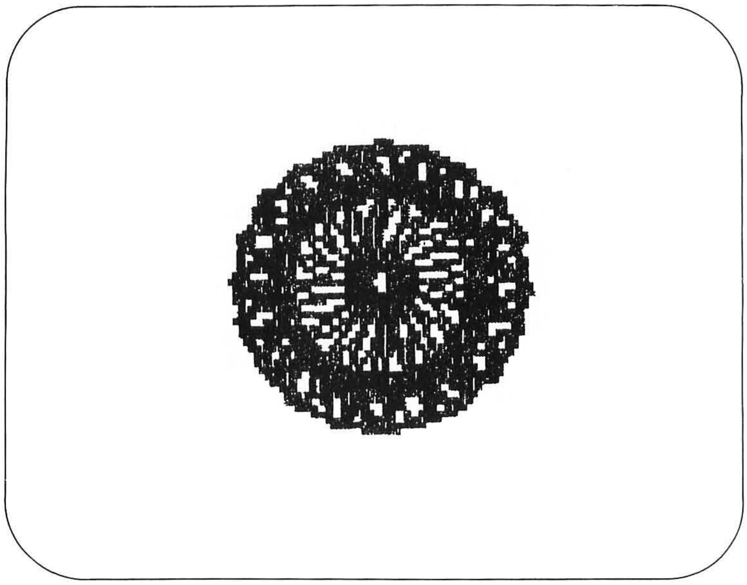


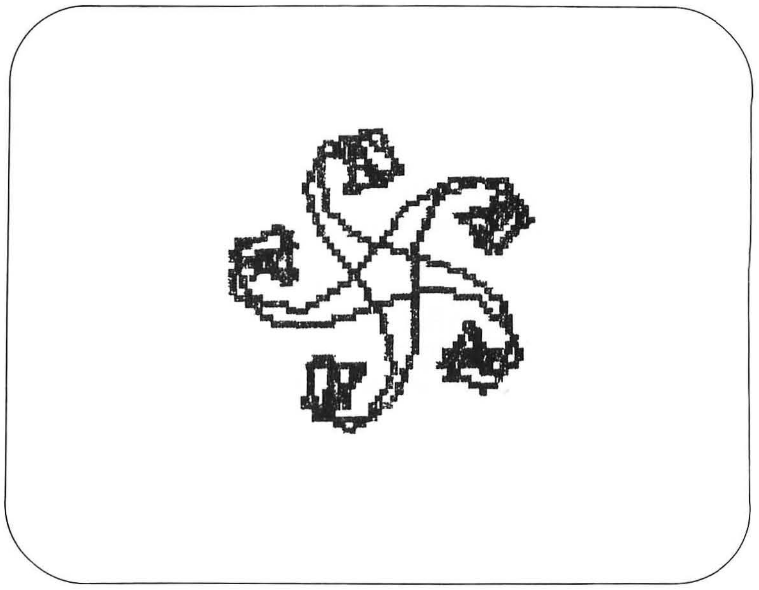
This surprising result occurs because the increment was chosen to keep us from reaching a turning angle of 180 degrees in the first arm of the spiral. As a result, the curve could not retrace itself until this condition was met.

More polygonal forms of the spiral can be made by starting with an offset angle. The following three figures were made using

```
CLOSESPI 15 40 30
CLOSESPI 20 1 20
CLOSESPI 20 2 20
```







(*Note:* You will get error messages when you draw some of these spirals because Atari Logo does not make turns in excess of 1,000 degrees. However, since you can subtract 360 degrees from a turning angle without changing the overall heading, a good project is for you to modify CLOSESPI to make sure that the turning angle never gets too large.)

You may wish to experiment with the CLOSESPI procedure some more. It can create many beautiful pictures.

Open Spirals

Spirals can be created from arcs that are drawn so that each segment is larger than its predecessor by some ratio.

Spirals that grow by fixed ratios are commonplace in nature. They appear in snail shells, whirlpools, and numerous other natural objects and phenomena.

Spirals of this type can be approximated by a procedure that draws a series of 90-degree arcs, with each arc larger than its predecessor by a fixed factor. There are two types of spirals—those that turn to the left and those that turn to the right. To generate a right-handed 90-degree arc, we can use the RARC procedure:

```
TO RARC :SIZE
REPEAT 5 [FD :SIZE RT 18]
END
```

We can also use this procedure for generating right-handed spirals by creating the RSPIRAL procedure:

```
TO RSPIRAL :FACTOR :STEPS
MAKE "SIZE 1
REPEAT :STEPS [RARC :SIZE MAKE "SIZE :SIZE * :FACTOR]
END
```

To generate left-handed spirals, we can use the following procedures:

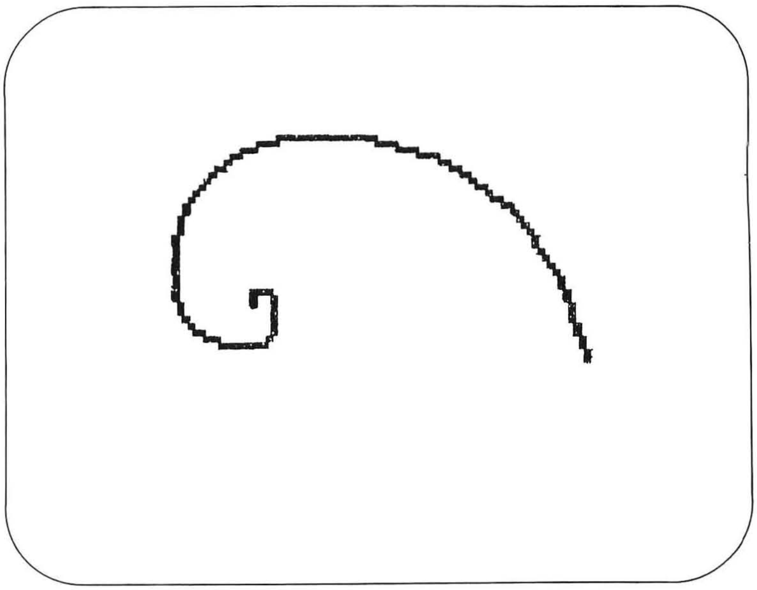
```
TO LARC :SIZE
REPEAT 5 [FD :SIZE LT 18]
END

TO LSPIRAL :FACTOR :STEPS
MAKE "SIZE 1
REPEAT :STEPS [LARC :SIZE MAKE "SIZE :SIZE *:FACTOR]
END
```

Next, we can experiment with spirals that use different expansion factors. If you enter

```
CS
RSPIRAL 1 10
```

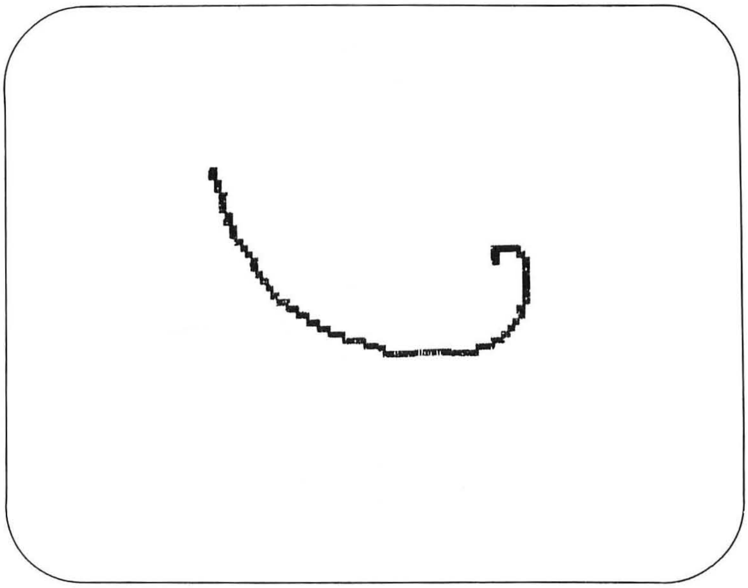
all you will see is a circle at the center of the screen. This is because you cannot increase the size of anything by multiplying it by 1. To see a spiral, we need a larger factor. Enter

```
CS
PU SETX -60 PD
RSPIRAL 2 6
```



By increasing the size of the expansion factor, we can create spirals that are more open. For example, enter

```
CS
RSPIRAL 3 4
```

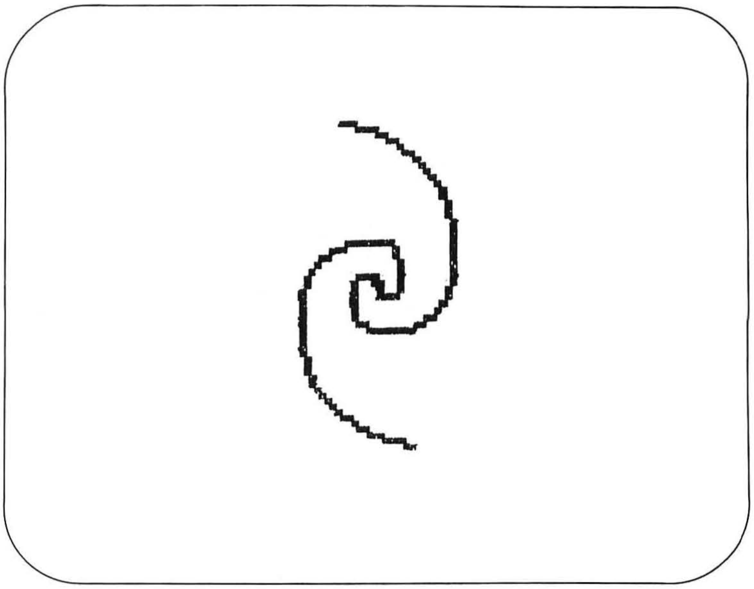


Interlaced spirals can be made by turning the turtle by different amounts before starting the spiral. For example, enter

```
CS
LSPIRAL 2 5
```

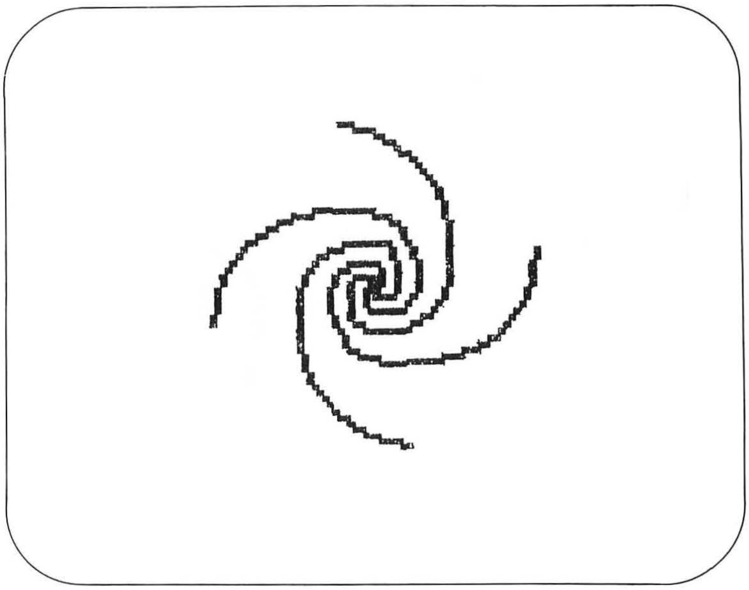
This draws a left-handed spiral from the center of the screen. Next, we will change the pen color to pink and draw the same spiral from the center, but we will turn the turtle by 180 degrees first. Enter

```
PU SETPOS [ 0 0] PD
SETPN 1
SETH 180
LSPIRAL 2 5
```



Notice that the pink spiral lies near the middle of the gap in the yellow spiral. To produce interwoven spirals with different locations, the same process can be repeated with other starting angles. The following example will add a dark green line on either side of the pink one:

```
SETPN 2
PU SETPOS [ 0 0] PD
SETH 90
LSPIRAL 2 5
PU SETPOS [0 0] PD
SETH -90
LSPIRAL 2 5
```



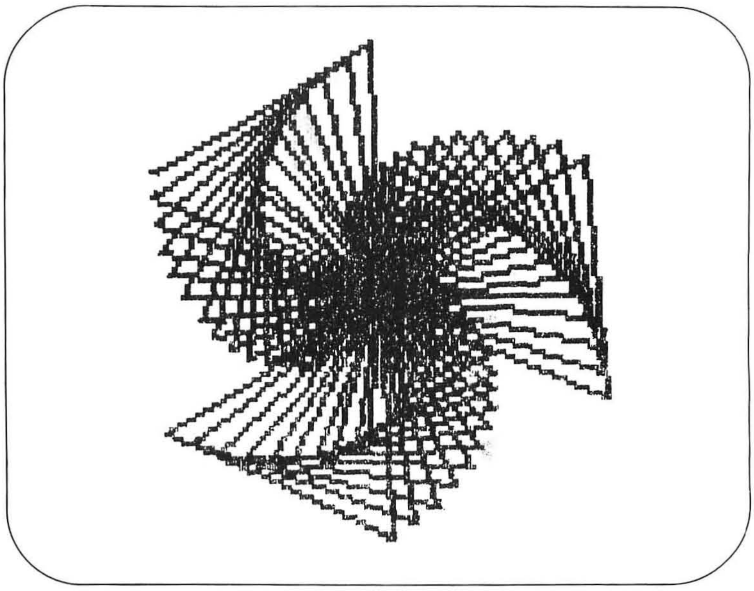
Projects That Involve Growth

Now that we have explored three types of spiral figures, you should explore some of your own. For example, try changing SQUIRAL to increase both the size and the angle at the same time. See how pleasing the figures are to you.

Next, you might wish to explore the properties of a procedure that moves the turtle back one step for each two steps forward (turning the turtle by some angle in between). One procedure for doing this is as follows:

```
TO TWOSTEP :ANGLE :STEPS
MAKE "SIDE 0
REPEAT :STEPS [FD :SIDE RT :ANGLE BK :SIDE / 2
  RT :ANGLE MAKE "SIDE :SIDE + 4]
END
```

The following figure was generated with TWOSTEP 61 60.



Try some other projects. For example, modify SQUIRAL (with judicious use of PENUP and PENDOWN) to have it draw only dots at the ends of each line. Replace the dots by small stars and see how you like the result.

Spiral procedures can generate many interesting pictures!

VIII. Additional Drawing Aids

As mentioned in the preceding chapter, you now have all the tools you need to create many wonderful graphic compositions. You may find, however, that it is tedious to use SETPOS and SETH all the time to pick the starting points for your procedures. Wouldn't it be nice if we could just move the turtle to some starting position and orientation without having to bother with a lot of typing? For that matter, wouldn't it be nice if some of our procedures could be drawn in response to a single keystroke?

This chapter introduces two new ways for you to communicate with your computer—using a joystick and having Logo "read" the keyboard in the middle of executing a procedure.

Before starting, clear your workspace of all procedures except such basic building blocks like POLY, STAR, and CIRCLE.

A Thing of Beauty Is a Joystick Forever

You may have noticed that your Atari computer console is equipped with two or four multipin jacks. These connectors are designed to accept joysticks and other controllers made just for the Atari computer. To use the joystick, you need to do two things: you must plug it into the computer, and you must learn which Logo words will let the computer monitor the joystick's position.

Connecting the joystick requires simply plugging it in. For our purposes, we will use a joystick plugged into the first controller jack. Since the connector will fit only one way, proper installation of the joystick should be completed in seconds. The status of the joystick can be determined with the Logo word JOY. To see how this word works, enter the following procedure:

```
TO JOYTEST
PRINT JOY 0
JOYTEST
END
```

JOY 0 is a Logo function that returns a number associated with the position of the joystick in the first controller jack. The JOYTEST procedure prints a column of numbers on the display screen corresponding to the position of the handle on the joystick.

JOYTEST has another interesting feature. After printing out the position of the joystick, it uses itself again. The task of having a procedure use itself is called *recursion*.

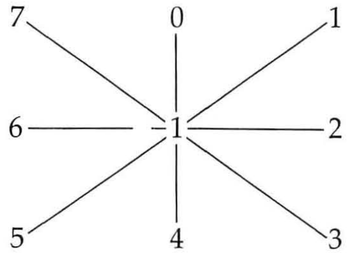
Now enter

```
TS CT
JOYTEST
```

You will see a column of numbers printed on the screen. If you are not pushing or pulling the joystick handle, the number should be −1.

Pick up the joystick and hold it in one hand, with the cable going away from you. If you grab the handle with the other hand and push or pull it, you will see the numbers in the column change.

As you move the joystick handle around, you will see a pattern to the numbers that appear, as follows:



Since each position of the joystick is associated with a unique number, we can use this number to move the turtle around on the screen.

As an example, let's create a procedure for the joystick that moves the turtle forward if the joystick is moved up and turns the turtle to the left or right as the joystick is pushed in either of these directions. We will have the procedure stop if the joystick handle is pulled toward you. The following procedure will do all these things:

```
TO MOVE
PU
START:
MAKE "STK JOY 0
IF :STK = 0 [FD 5]
IF :STK = 6 [LT 15]
IF :STK = 2 [RT 15]
IF :STK = 4 [PD STOP]
MOVE
END
```

We used the new word STOP here. When Logo encounters STOP, it halts the execution of the procedure and returns control to the user or to the procedure that used the procedure that just stopped. For example, if MOVE was used by another procedure (such as DRAW), then when MOVE stops, the DRAW procedure picks up where it left off.

Let's try MOVE to see how it works. Enter

```
CS
MOVE
```

As you push the joystick forward, the turtle starts to move up the screen. If you push the joystick to the right or left, the turtle will turn in the corresponding direction in 15-degree increments. Finally, pull the joystick back toward you. This will stop the procedure.

Single-Keystroke Procedures

Now that we know how to move the turtle around on the screen with the joystick, let's examine a way to create various figures by pressing just a single key.

Logo provides an easy way to read characters from the keyboard using the command RC (for *Read Character*). To see how this command works, enter the following procedure:

```
TO TYPEIT
MAKE "CHAR RC
TYPE :CHAR
TYPEIT
END
```

If you now enter

```
TS CT
TYPEIT
```

and start typing, you will see text appear on the screen. When TYPEIT encounters the RC command, it waits for you to press a key. As soon as you press a key, the character you pressed is placed in the variable CHAR. Next, this character is displayed on the screen with the TYPE command. The only difference between TYPE and PRINT is that PRINT starts on a new line each time it is used but TYPE does not.

To run procedures with single keystrokes, we first have to define our procedures to have single-character names. As an example, let's define the following procedures:

```
TO S
POLY 4 40
END

TO P
STAR 5 2 40
END
```

```
TO C
CIRCLE 5 10
END
```

Now we can combine all we have learned into a simple drawing program that moves the turtle under joystick control and lets us draw figures with single keystrokes. The procedure DRAW is one simple way to do this:

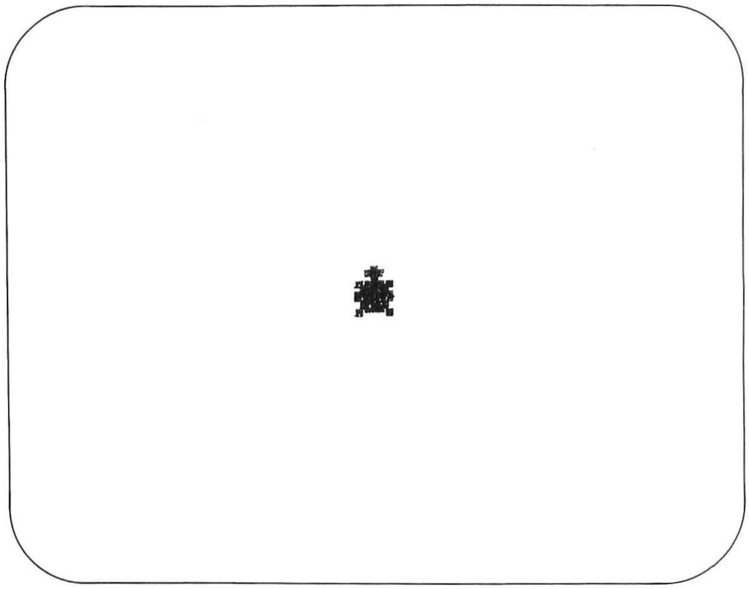
```
TO DRAW
MOVE
MAKE "CHAR SE RC []
REPEAT 1 :CHAR
DRAW
END
```

Because REPEAT needs a list as input, we can't just use the character returned from the RC command without making it into a list first. We do this with the SE (*SEntence*) command. Since SE makes a list from two words or lists, we use the character given by RC as the first word and the empty list, [], as the second. If you find this is getting too technical for your blood, don't feel that you must understand it perfectly—it works even if you don't fully understand it.

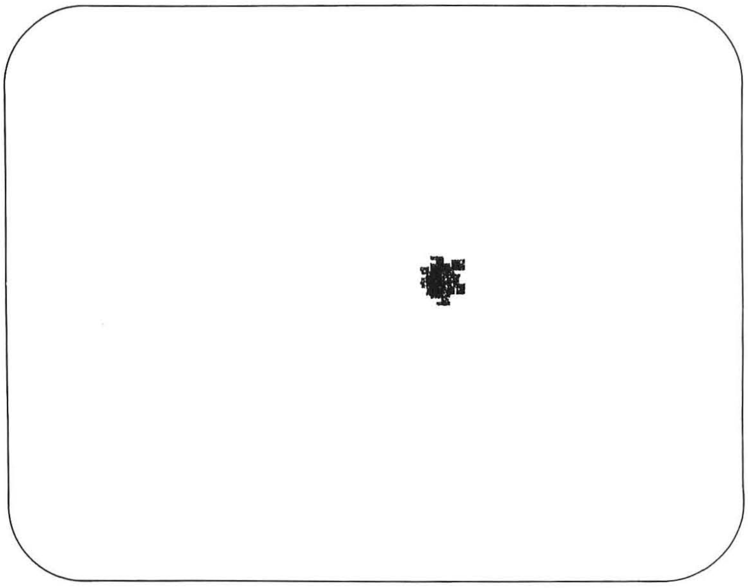
Without further ado, then, enter

```
CS
DRAW
```

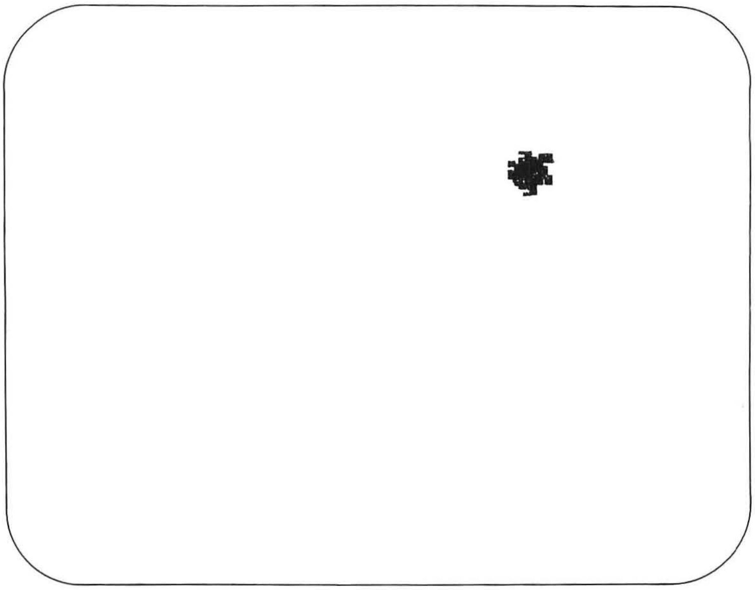
We will start with the turtle in the center of the screen.



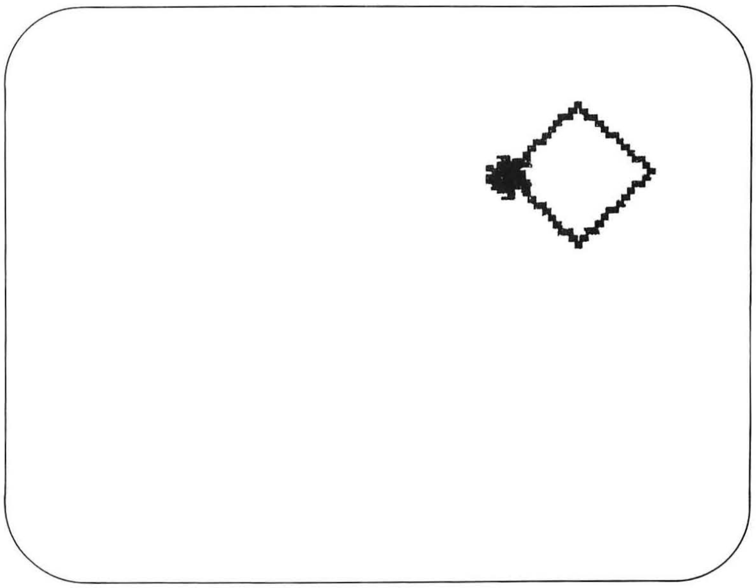
Now turn the turtle a bit to the right with the joystick.



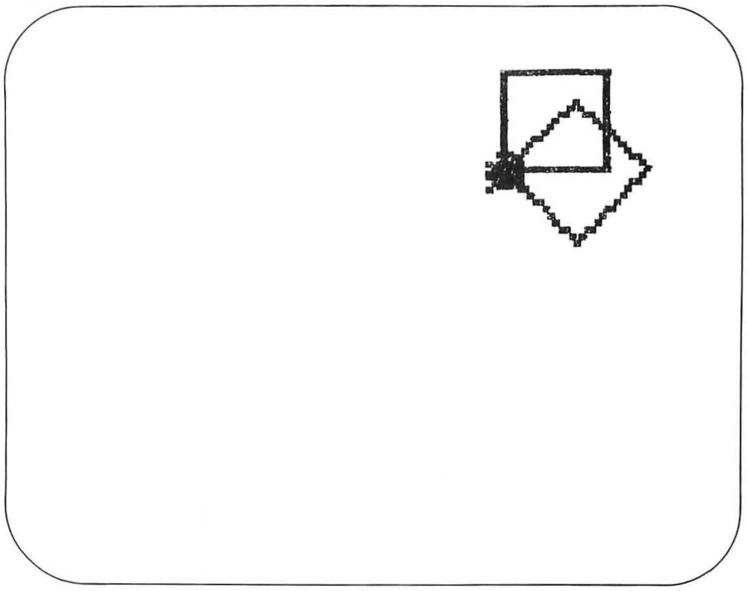
Push the joystick forward to a new location.



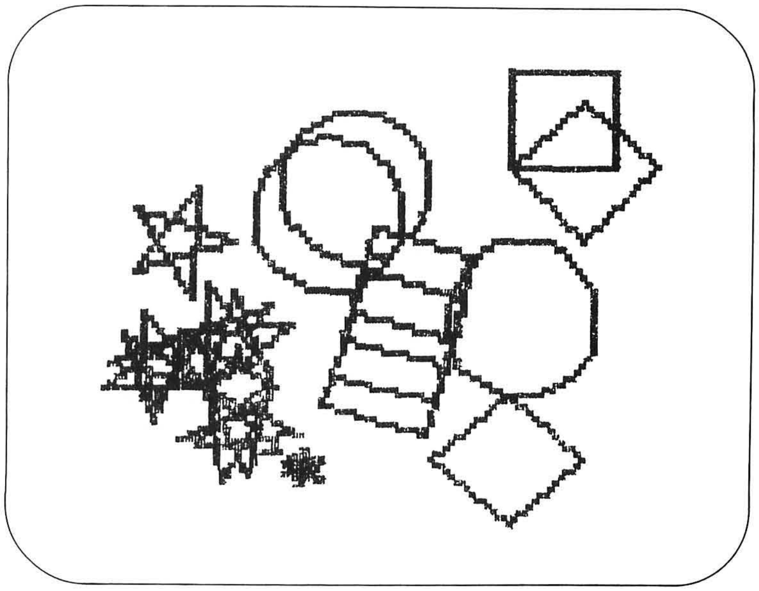
Next, pull the joystick back and press the S key. A tilted square will be drawn on the screen.



If you now push the joystick to the left, the turtle will turn to the left. When the turtle is pointing straight up again, pull the handle and press S again.



Using the joystick to steer and move the turtle, add some more figures to this picture.



As you can see, we have used Logo to create a simple graphics *program* that lets us "rubber-stamp" procedures at different screen locations. Using the ideas we have explored in this chapter, add some other features to this program to make it useful for your purposes. Some possibilities would be to create single-keystroke procedures to change pen color and background color and to draw different shapes.

Saving Pictures

The most efficient way to save pictures created with Logo is to save them as Logo procedures. It is sometimes hard to do this, however, because we have created our pictures interactively and have lost track of the commands that generated them. In such cases, we can still save the image on the screen by placing a copy of the computer's display memory on the diskette. The image you see on the display screen is "refreshed" many times per second by the computer system. For the computer to perform this task, it has to devote some of its memory space to keeping track of the colors of the various picture locations (called *pixels*) on the screen. This region of memory is called the *frame buffer.* In Atari computers, the Logo frame buffer occupies 3,840 memory locations that start at the memory "address" 16384. Also, Atari computers use memory locations 708, 709, 710, and 712 to store information on the pen and background colors. (Don't panic if this sounds like gibberish to you—the result is all you will need to understand.)

To save the screen image, we must tell the disk drive that we want to save some information. Next, we have to examine the information at each memory location and place it on the disk. The SAVEPICT procedure does this, with the help of two primitives: SETWRITE and .EXAMINE. SETWRITE tells Logo to prepare a device (such as the disk drive) to receive data. The function .EXAMINE reads the value of the data stored at a specified memory location. (Commands like .EXAMINE and others that start with a period are used infrequently—often for good reason.)

The SAVEPICT procedure that works this magic for us is defined as follows:

```
TO SAVEPICT :FILENAME
SETWRITE :FILENAME
MAKE "LOC 16384
TYPE CHAR .EXAMINE 708
TYPE CHAR .EXAMINE 709
TYPE CHAR .EXAMINE 710
TYPE CHAR .EXAMINE 712
REPEAT 3840 [TYPE CHAR .EXAMINE :LOC MAKE "LOC
  :LOC + 1]
SETWRITE []
END
```

This procedure saves not only the picture but also the pen color and background information.

Suppose that you had a garden scene on your display that you wanted to save. To save the picture, you might enter

[illegible]

```
SAVEPICT "D:GARDEN.PIC
```

In other words, SAVEPICT works like SAVE, except that you are saving a picture image rather than Logo procedures. The use of an extension like .PIC will help remind you that this is a picture file, not a Logo file.

To read your picture back from the diskette, you need the companion procedure LOADPICT:

```
TO LOADPICT :FILENAME
HT FS
SETREAD :FILENAME
MAKE "LOC 16384
.DEPOSIT 708 ASCII RC
.DEPOSIT 709 ASCII RC
.DEPOSIT 710 ASCII RC
.DEPOSIT 712 ASCII RC
REPEAT 3840 [.DEPOSIT :LOC ASCII RC MAKE "LOC :LOC + 1]
SETREAD []
END
```

LOADPICT is used in the same manner as LOAD. For example, to load the picture GARDEN.PIC from the diskette, you would enter

```
LOADPICT "D:GARDEN.PIC
```

When using SAVEPICT, you must be conscious of your available disk space. Each picture file occupies 31 sectors of the diskette. This means that you can store only 20 images on a single diskette. Even so, SAVEPICT and LOADPICT are invaluable tools for letting you save an otherwise lost image.

The Turtle Meets the Koala

Joystick-based drawing programs (such as the procedures used with DRAW) are fine for stamping images around on the screen. Sometimes, however, it is nice to be able to make "freehand" sketches—even when we are using a computer system—to display our images. One way to make freehand sketches is with the help of a graphics tablet. The KoalaPad tablet (from Koala Technologies, Inc.) is an inexpensive device that connects to the Atari controller jack and allows you to move the turtle around on the display screen in response to the motion of your finger or a stylus on the tablet surface. The tablet also contains two switches that Logo can examine.

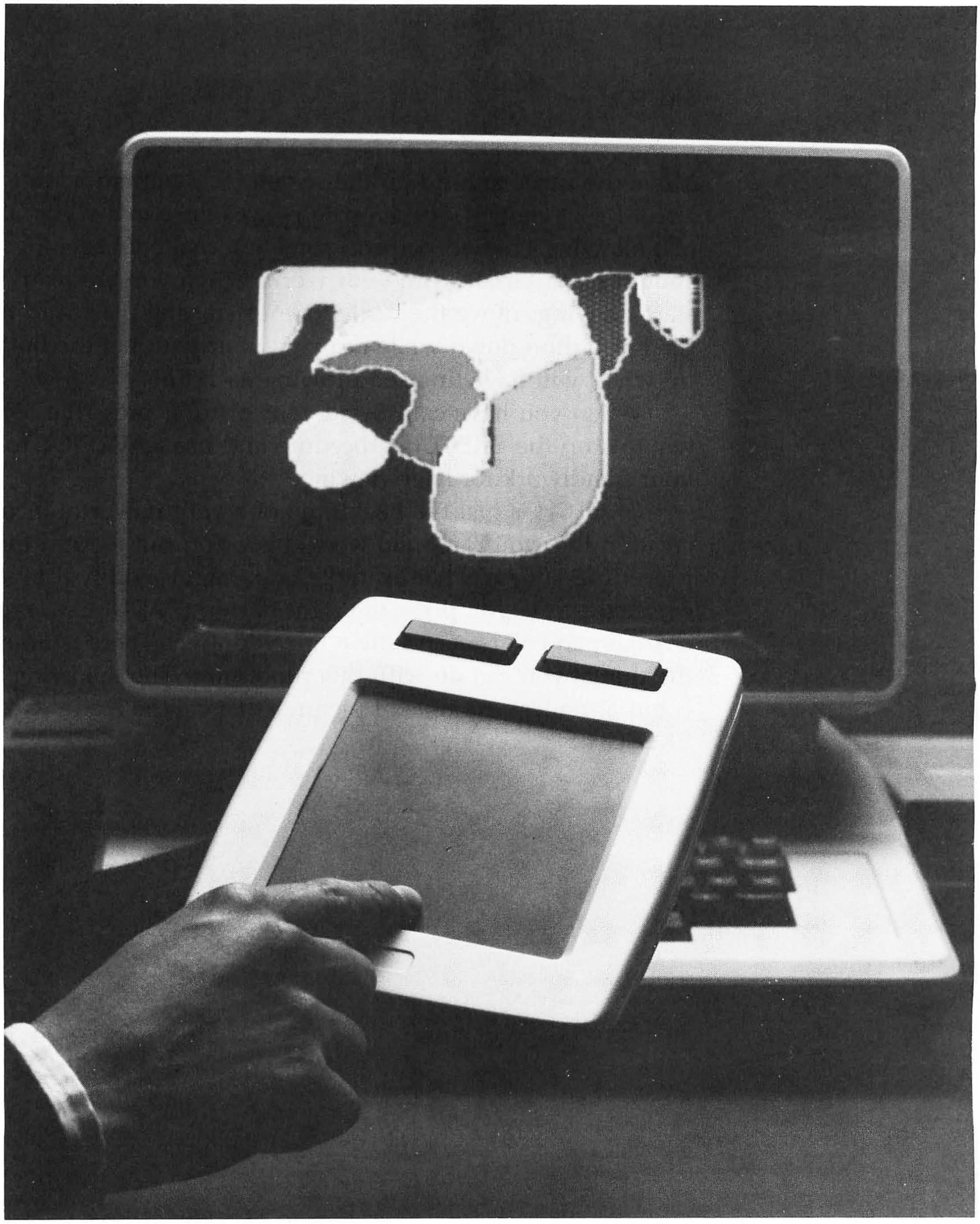
The Logo primitives that read the status of the tablet and its buttons are PADDLE and PADDLEB. These names were chosen because the KoalaPad is read by Logo in the same way one would read a set of paddle controllers. If the graphics tablet is plugged into the first controller jack, PADDLE 0 will indicate the *x*-coordinate of a stylus on the tablet surface, and PADDLE 1 will indicate the *y*-coordinate. PADDLEB 0 will show the state of the left tablet button, and PADDLEB 1 will show the state of the right button.

If you experiment with the tablet, you will find that the PADDLE 0 values vary from the mid-220's on the left side to near 0 on the right side. For PADDLE 1, these values correspond to points near the top and bottom of the tablet surface. To translate these values to useful screen positions, we need to perform some simple calculations. These calculations are performed as part of a simple sketching procedure called SKETCH:

```
TO SKETCH
IF PADDLEB 0 [PD] [PU]
MAKE "X 1.3 * (110 - (PADDLE 0))
MAKE "Y (PADDLE 1) - 110
SETPOS SE :X :Y
SKETCH
END
```

The IF command in this procedure puts the pen down if the left tablet button is pressed and lifts the pen otherwise.

The best way to test the SKETCH procedure is to use it. Enter

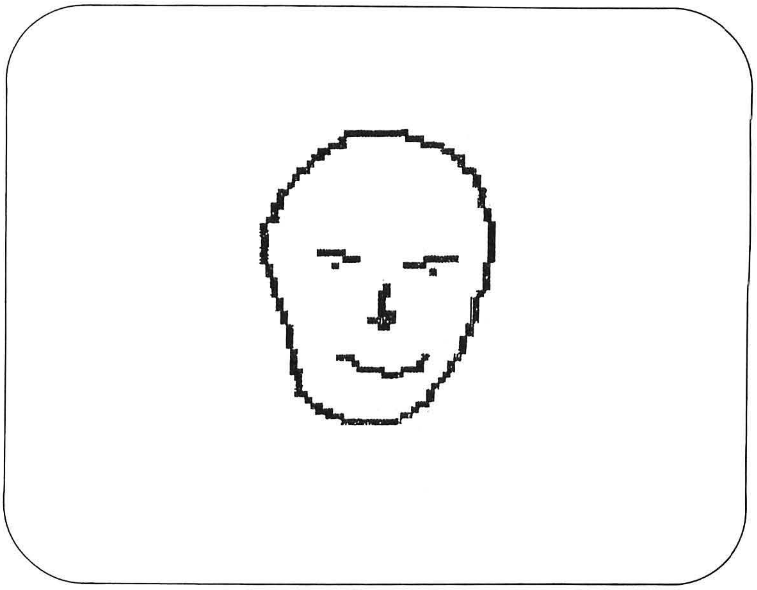


```
CS ST FS
SKETCH
```

Move the turtle around on the screen for a minute or so to get a feel for its motion. If the turtle pauses once in a while, it is just allowing Logo to perform some internal bookkeeping; this should not interfere with your work. When you are ready to start drawing, move the turtle to your starting place and hold the left button down while drawing your line. Let the button up when you have finished drawing each line.

When you have completed your picture, press the BREAK key to stop the SKETCH procedure and use SAVEPICT to save your handiwork for later viewing.

SKETCH is just the beginning of a very powerful picture-creation tool you can build with Logo. You can expand this procedure yourself. For example, you might modify it to start over each time you press the right button (PADDLEB 1) or to change pen colors each time a line is drawn. There is no end to the things you can do with this procedure. The following figure is but a modest example of pictures that can be made with SKETCH.



Shaded Cubes and Three-Dimensional Graphics

One of the unique features of Atari computer systems is their ability to let you choose a color as a combination of a hue and luminance (see Chapter III). Because you have independent control of these parameters, you can use this powerful feature to create properly shaded "three-dimensional" objects. The effective use of shading in your pictures can give vitality and depth to your computer graphics.

To illustrate this feature, we will create an image of a three-dimensional cube with Logo. Because we can only see three faces of a cube at a time (at most), we can draw a cube with the following procedure:

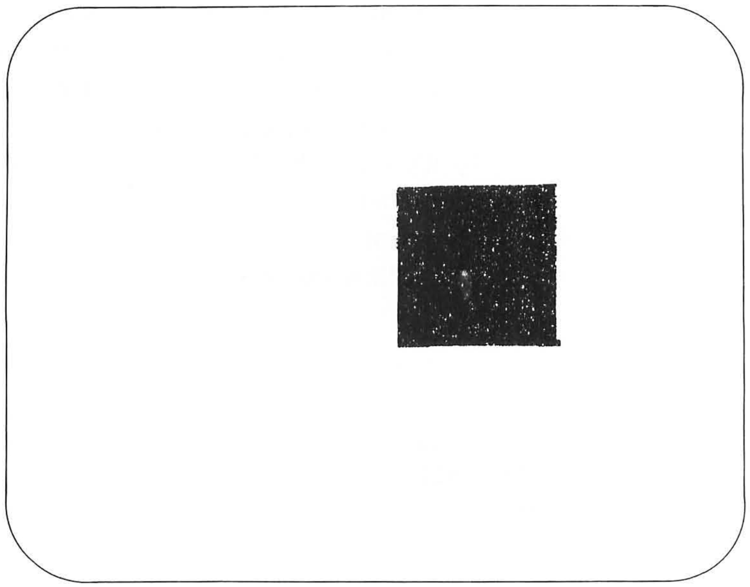
```
TO CUBE :SIZE
FRONT :SIZE
SIDE :SIZE
TOP :SIZE
END
```

Each of the faces should be drawn with a different pen to allow us to shade it properly. Also (and this is important), each face should be filled in, not just drawn as an outline. Although writing a Logo procedure to fill various generalized polygons might be a nice project to pursue at your leisure, one way to make filled objects is to draw them that way in the first place, as a sequence of adjacent drawn lines. For example, the following procedure (which we will use to draw the front face of the cube) will draw a solid square of any size:

```
TO FRONT :SIZE
SETPN 0
REPEAT :SIZE [FD :SIZE BK :SIZE RT 90 FD 1 LT 90]
END
```

To test this procedure, type

```
CS HT
FRONT 60
```



Next, we need to create the procedures for the side and top faces. For our projection of a cube, each of these faces will be a rhombus with a 45-degree angle. The SIDE procedure is as follows:

```
TO SIDE :SIZE
SETPN 1
REPEAT :SIZE * 0.707 [FD :SIZE BK :SIZE RT 45 FD 1 LT 45]
END
```

The factor of 0.707 is included to provide the proper depth for the perspective we have chosen. To draw the top of the cube, we must first move the turtle to the starting point and then draw another filled rhombus:

```
TO TOP :SIZE
FD :SIZE LT 90
SETPN 2
REPEAT :SIZE * 0.707 [FD :SIZE BK :SIZE LT 45 FD 1 RT 45]
END
```

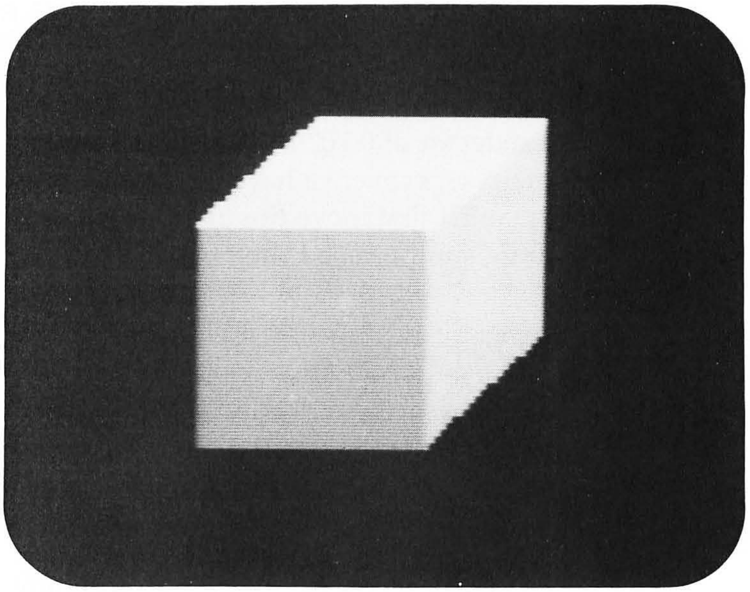
Next, we need to create a procedure that will assign the same hue to all three faces but will also provide proper shading. The following procedure sets the background to black (*very* striking) and sets the pens to the same hue but to different luminance settings:

```
TO COLORS :HUE
SETBG 0
SETPC 0 (8 * :HUE) + 3
SETPC 1 (8 * :HUE) + 4
SETPC 2 (8 * :HUE) + 5
END
```

This procedure will make our cube appear to be illuminated from above.

To see the results of our efforts thus far, enter

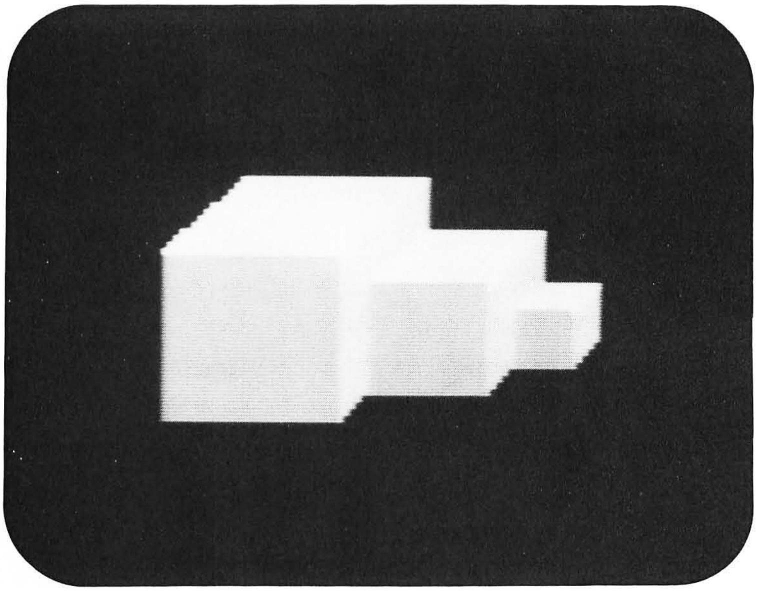
```
CS HT
COLORS 4
CUBE 80
```



The resulting image on your display screen should be a fairly realistic representation of a shaded cube. The shading, by itself, gives the illusion of dimensionality to this figure.

For another striking image, you might want to try the following arrangement of nested cubes:

```
TO NEST
PU SETH 0 SETPOS [-60 -20] PD CUBE 60
PU SETH 0 SETPOS [10 -10] PD CUBE 40
PU SETH 0 SETPOS [60 0] PD CUBE 20
END
```

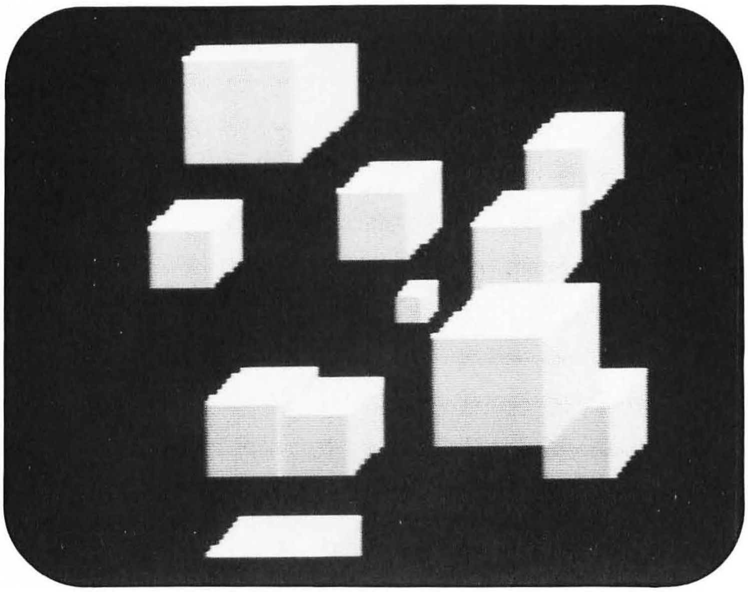


Once you have drawn your figures, you can change the colors by using the COLORS procedure with different hue values. The hue value can be any number from 0 to 15.

The next procedure places cubes of random size at random screen locations. The procedure pauses between drawings until you press a key—any key—on the keyboard.

```
TO RANDOMCUBE
PU SETH 0
SETPOS SE ((RANDOM 200) -100) ((RANDOM 150) -75) PD
CUBE (RANDOM 40) + 10
MAKE "KEY RC
RANDOMCUBE
END
```

The following figure was created with the RANDOMCUBE procedure.



When your picture has achieved the complexity you desire, you can save it by pressing BREAK and using the SAVEPICT procedure.

Some Projects

At this point in our exploration, we have not only covered turtle graphics, but we have learned how to create simple drawing programs and the illusion of three dimensions.

The next few chapters focus on the other powerful graphic capabilities of Atari Logo, including animation. This is a good time for you to examine your feelings about computer graphics once again. Do you like creating pictures with turtle graphics? Do you find that you are expressing ideas differently with this medium than you would with watercolors, for example?

You might want to spend a week or so exploring some picture ideas on your own before proceeding to the next chapter. Can you make a harsh picture using only straight lines? Can you make a warm picture with arcs and circles? Next, try reversing things: make a harsh picture using only arcs and circles and a warm picture using only straight lines. Don't forget that you can use color in your pictures; color is a powerful tool in itself.

IX. Multiple Turtles and Animation

One feature of television is that it allows us to see moving images. Although your Atari computer uses a television screen for its display, all the figures we have examined thus far are static. In this chapter, we will explore ways that Atari Logo lets you create moving images on your display.

Atari Logo is equipped with a powerful graphics tool that lets us create multiple turtles that can be placed anywhere on the screen. Furthermore, these shapes can be assigned a velocity or speed with which they will move across the screen. By combining several of these turtles into a larger shape, multicolored moving objects can be created.

Multiple turtles are the tools that allow you to create your own animated sequences. In a later chapter (Chapter XI), you will learn how to record these sequences on videotape, so you will soon be able to create your own animated films in the comfort of your own computer workplace. First, however, we must learn how to create multiple turtles and move them on the screen.

Moving Shapes

To see an example of something new we can do with a turtle, enter the following commands:

```
CS PU
SETSP 20
```

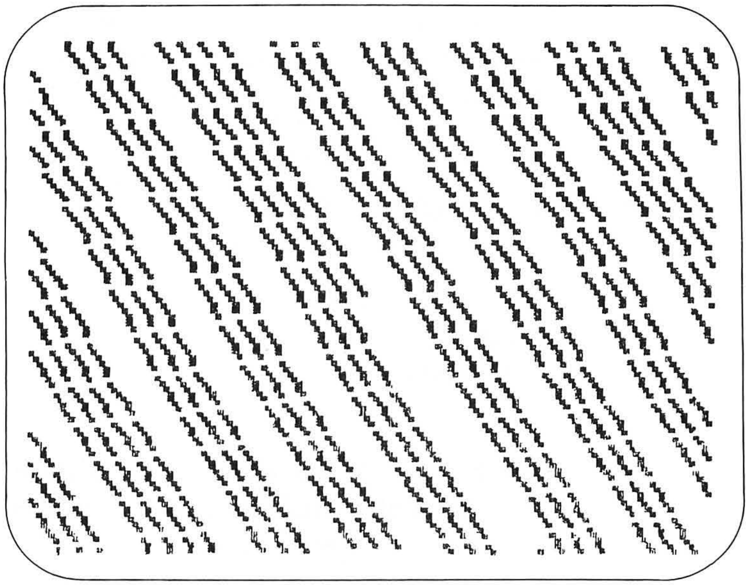
The SETSP (*SET SPeed*) command starts the turtle moving up the screen at a fixed velocity. So long as your screen is in the WRAP mode, the turtle will reappear at the bottom when it runs off the top of the screen. If you had given the WINDOW command first, the turtle would have kept going when it reached the top.

To increase the speed, enter

```
SETSP 100
```

Now the turtle is really moving! Try other values of SETSP, such as −80, 200, and so on, to find out what values SETSP can take. Next, have the turtle move at a modest pace, and enter other turtle graphic commands, such as PD, RT 90, and the like. You will find that your moving turtle can draw lines as it moves. Note, however, that these lines will be unbroken only if the turtle speed is under about 40. The following figure was made by giving the turtle the following instructions:

```
CS HT PD
SETH −30
SETSP 50
```



In addition to allowing us to impart a velocity to the turtle, Atari Logo lets us assign shapes of our own design and color to the turtle. The only difference you will find with your new shapes is that commands like LEFT and RIGHT don't change the orientation of the shape but do change the direction of the turtle's subsequent motion. If we tell a turtle carrying a new shape to turn to the right by 90 degrees, the image of the shape won't turn, although it will move in the new direction when we type a movement command, such as FD 50.

The SETC (*SET Color*) command can be used to set the color of the turtle itself. For example, the command

```
SETC 70
```

will set the color of the turtle to blue.

The SETSP command imparts motion to turtles. This motion will be forward for speed values from 1 (very slow) to 199 (very fast) and backward for speed values from −1 to −199.

As you have discovered, you can change the direction of a turtle while it is moving. For example, enter

```
SETSP 30
RT 90
```

As soon as you entered RT 90, the turtle changed direction and started traveling from the left of the screen to the right.

We can use SETSP to send our turtle on a square path. The following procedure should do this very nicely:

```
TO PATH
SETSP 40
WAIT 80
RT 90
PATH
END
```

To try this procedure, enter

```
CS
PATH
```

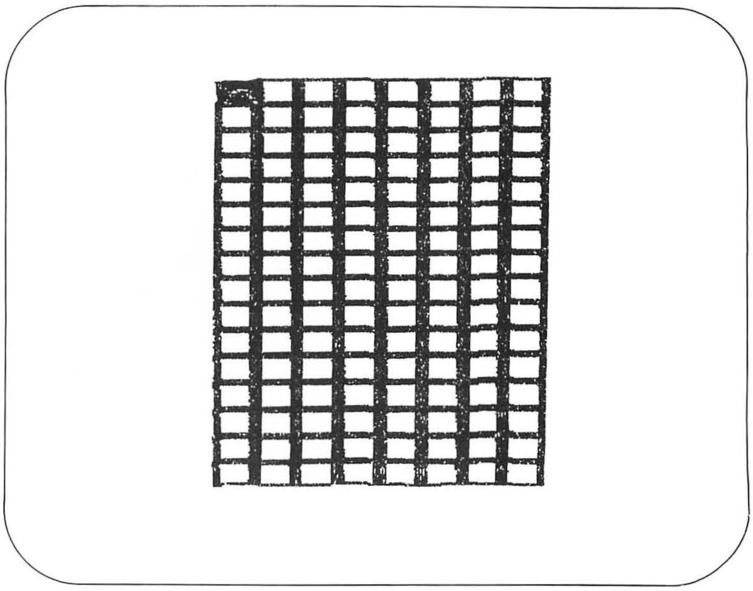
The turtle will now trace a square path on the right half of the display screen. The command WAIT causes Logo to pause for a while before executing the next instruction. Each unit in the WAIT command is 1/60 of a second. When you tire of this path, press the BREAK key and type CS to stop the procedure.

Making Your Own Shapes

Atari Logo lets you create 15 shapes of your own. These shapes are blank when the computer is first turned on. To define a shape of your own, you must use the EDSH (*EDIT SHape*) command.

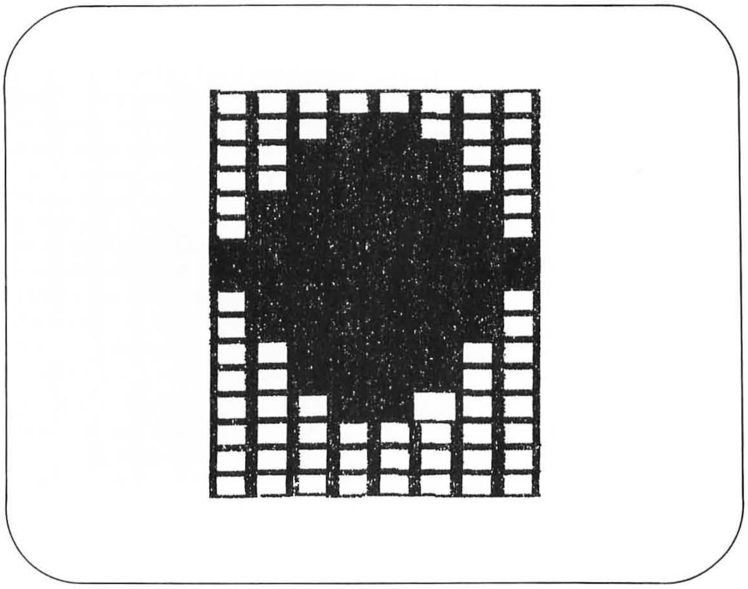
To illustrate the shape-defining process, let's make a shape that looks like a ball. Enter

```
EDSH 1
```



As you can see, a large grid appears on the screen, with a black box (the cursor) located in the upper left corner. If you count the boxes in the large grid, you will see that turtle shapes are made from a matrix formed from an 8 × 16 dot array.

To create a shape, you use the cursor control keys and the space bar. The cursor keys (used in conjunction with the CTRL key) move the black cursor to any position you want on the grid. When you press the space bar, you reverse the color of the dot at the cursor location. If you press the space bar once, the dot will be filled. If you press it again, the dot will be cleared. Using these keys, edit the image so that it has the following shape:



When you have edited the shape to your satisfaction, press the ESC key to return to the normal display screen.

Shapes can be saved automatically on the diskette, along with any procedures you may have defined. To save shapes (and to redefine them later), you must first assign the shape data to a variable. Since you will be using shapes a lot, it is convenient to have one procedure that will define variables for all your shapes and another that will reset all the shapes when you load them back from your diskette. The following two procedures—SAVESHAPES and LOADSHAPES—will do this for you:

```
TO SAVESHAPES
MAKE "SH1 GETSH 1
MAKE "SH2 GETSH 2
MAKE "SH3 GETSH 3
MAKE "SH4 GETSH 4
MAKE "SH5 GETSH 5
MAKE "SH6 GETSH 6
MAKE "SH7 GETSH 7
MAKE "SH8 GETSH 8
MAKE "SH9 GETSH 9
MAKE "SH10 GETSH 10
MAKE "SH11 GETSH 11
MAKE "SH12 GETSH 12
MAKE "SH13 GETSH 13
MAKE "SH14 GETSH 14
MAKE "SH15 GETSH 15
END

TO LOADSHAPES
PUTSH 1 :SH1
PUTSH 2 :SH2
PUTSH 3 :SH3
PUTSH 4 :SH4
PUTSH 5 :SH5
PUTSH 6 :SH6
PUTSH 7 :SH7
PUTSH 8 :SH8
PUTSH 9 :SH9
PUTSH 10 :SH10
PUTSH 11 :SH11
PUTSH 12 :SH12
PUTSH 13 :SH13
PUTSH 14 :SH14
PUTSH 15 :SH15
END
```

It is important to use these procedures properly to ensure that your shapes are saved and reloaded properly. Just before you are ready to save your procedures and shapes on your diskette, you should enter

```
SAVESHAPES
```

and then save your workspace on the diskette. When you are loading a fresh set of files into your workspace, load them in from the diskette and then type

```
LOADSHAPES
```

This will automatically initialize all the shapes to their proper values. Obviously, you must have a copy of SAVESHAPES and LOADSHAPES in the file as well.

Now that your new shape has been created, you should send it streaming across the screen. To do this, just give the following commands:

```
CS
SETSH 1
PU SETSP 50
```

The SETSH (*SET SHape*) command assigns the shape corresponding to a given shape number to the most recently addressed turtle (or turtles).

Many Turtles

Four turtles are available for your use, and you can have all four of them on the screen at the same time. Each turtle can carry any of 16 shapes—the standard turtle (shape 0) and the 15 shapes you can define—and can have its own color, speed, location, and orientation. Atari Logo allows you to send messages to turtles individually or in any turtle cluster you want. The ability to cluster turtles is powerful, since it lets you create multiple-turtle objects that can be moved as a unit with a single command.

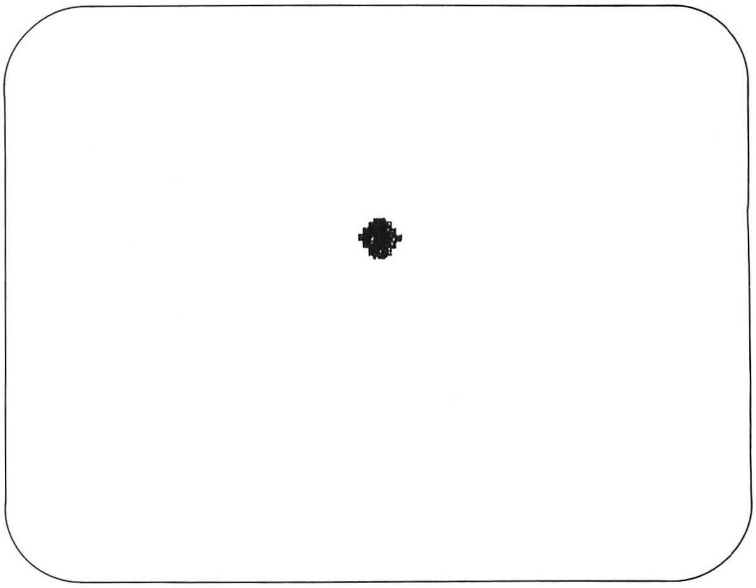
As an example, let's do an experiment with all four turtles. Enter

```
CS PD
TELL [0 1 2 3]
SETSH 0
```

This sets up all four turtles so that all subsequent messages will be obeyed by the group as a whole. Thus, for example, if we enter

```
SETSH 1
SETC 14
```

we will see a single ball in the center of the screen. In fact, all four turtles are there, carrying identical shapes with identical colors (set with the SETC command), and they are stacked one on top of the other.



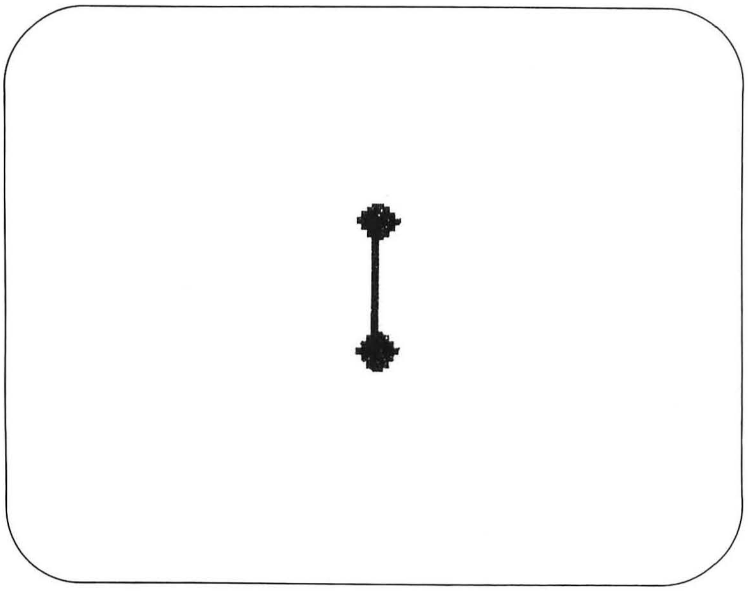
What is the stacking order? To find out, let's send some messages to turtles one at a time. Enter

```
TELL 0
SETC 70
```

The ball turned blue as soon as we pressed RETURN, showing that turtle 0 is on the top of the stack. If we now enter

```
FD 50
```

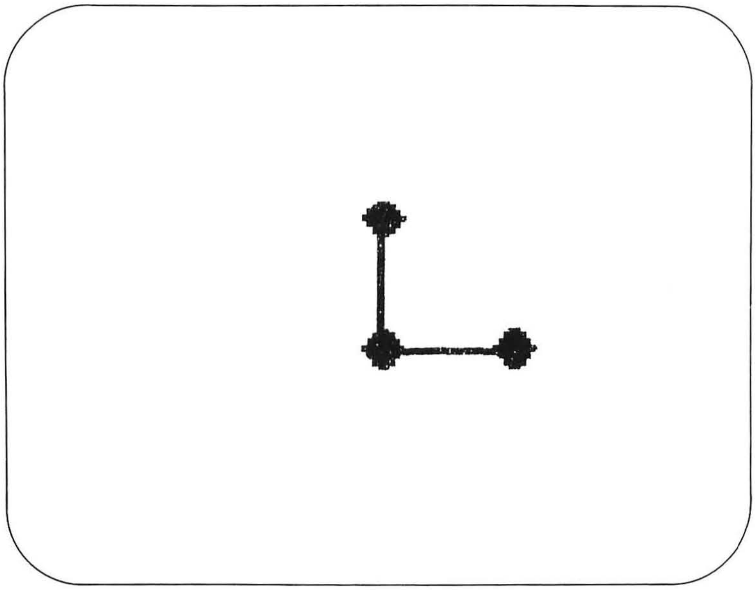
the movement of the blue turtle reveals the yellow turtle (number 1) underneath.



If we send the message

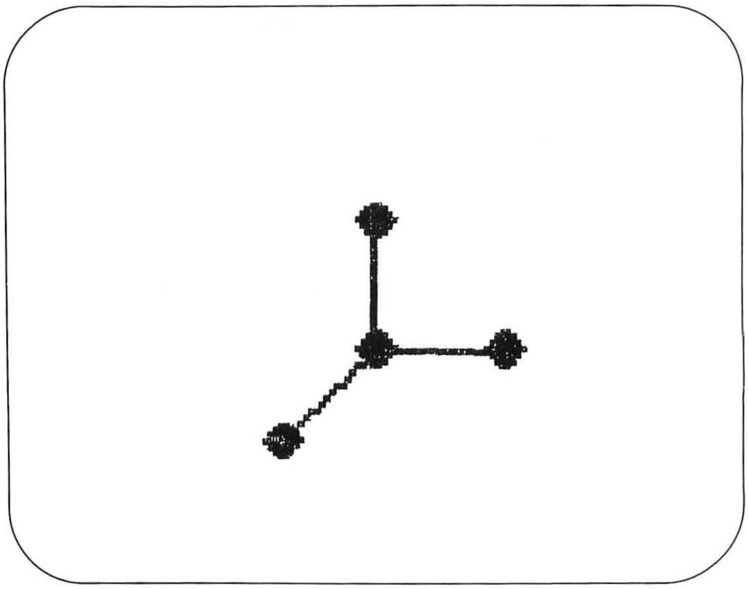
```
TELL 1
RT 90
FD 50
```

we will move turtle 1 to the right and reveal turtle 2.



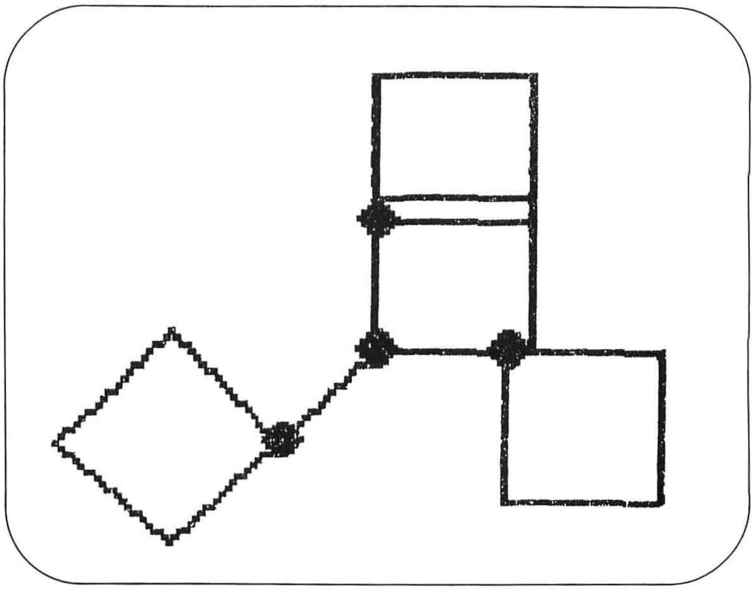
We can establish the identity of turtle 2 by entering

```
TELL 2
LT 135
FD 50
```



Now we can create all kinds of havoc on the screen by sending each turtle off in a different direction. For example, enter

```
TELL [0 1 2 3]
PATH
```



You will see a dazzling array of moving balls on the screen. To stop the motion, press the BREAK key and enter

```
SETSP 0
```

and all the turtles will stop.

To make all the turtles disappear, enter

```
HT
```

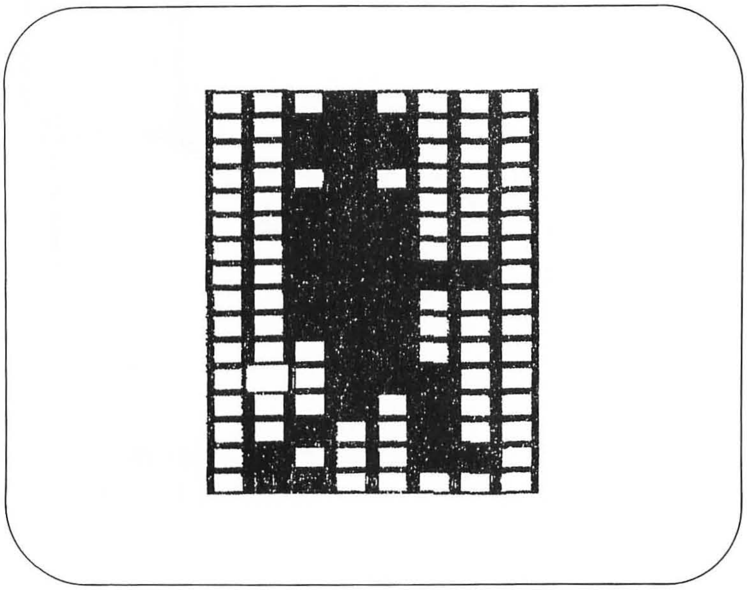
This hides all the turtles, because our last TELL command was directed to all of them.

One Turtle, Many Shapes

The ability to make a shape move across the screen is only one aspect of animation. To see why we need even more techniques, let's experiment with the shape of a human figure. Create this image as shape 6 by entering

```
EDSH 6
```

and edit it to look like this:



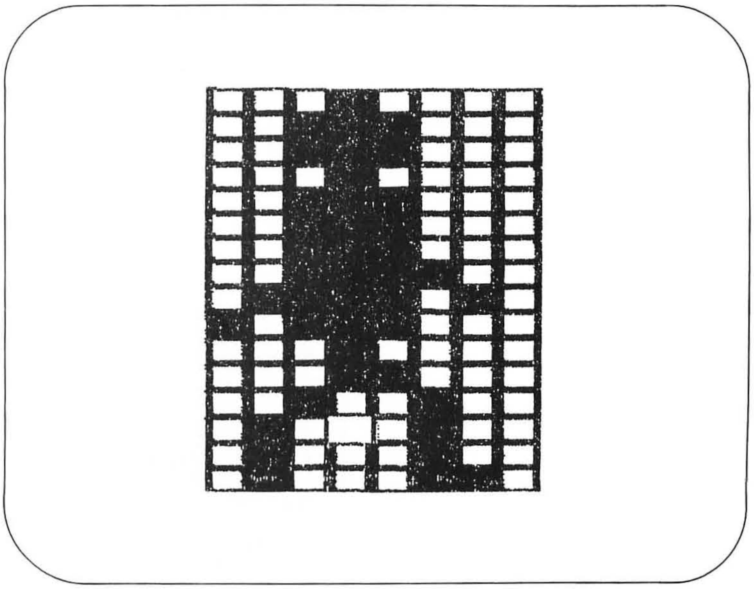
If we now set this figure on the screen and make it "walk," we will see that something is very wrong. Enter

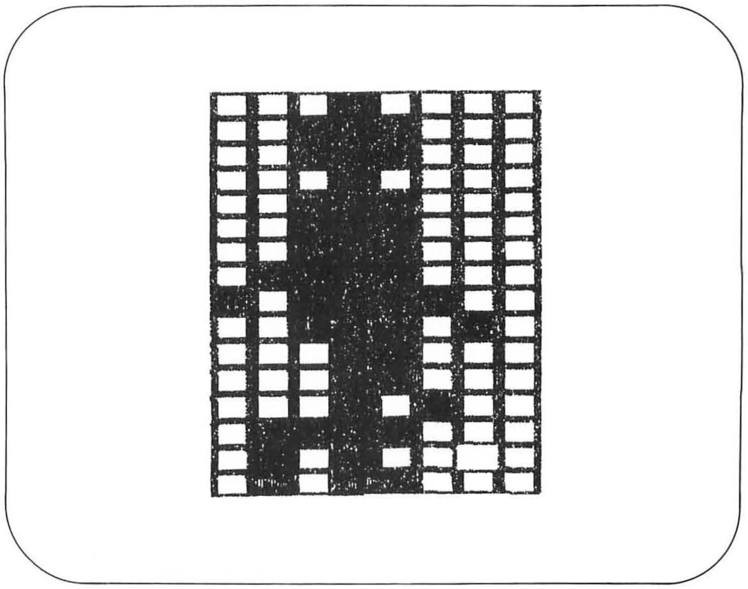
```
TELL 0
ST PU
SETSH 6
RT 90
SETSP 15
```

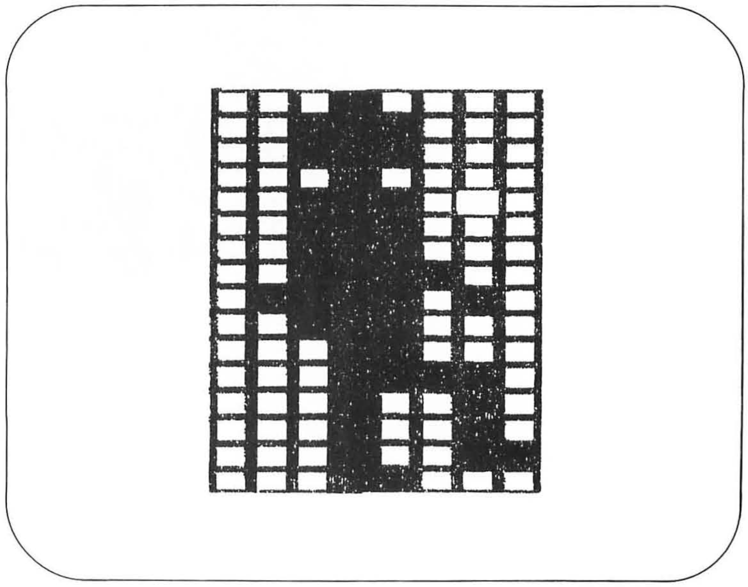
The figure moves from left to right across the screen, but it appears to be floating rather than walking. To fix this, we need to define some other shapes that show intermediate body positions while it is walking. The next few shapes should help us a great deal. Define them in order, as shapes 7, 8, and 9. To make the editing task a bit easier, you can start each figure with the pattern in shape 6 by entering

```
PUTSH 7 GETSH 6
PUTSH 8 GETSH 6
PUTSH 9 GETSH 6
```

(PUTSH stands for *PUT Shape* and GETSH stands for *GET SHape*.)







Finally, to get the illusion of walking, we not only need to have the turtle move, but we need to repeat the sequence of walking frames over and over again. The following procedures will let us experiment with this:

```
TO WALK :SPEED :GAIT
TELL 0
PU
SETH 90 SETSP :SPEED
ANIM :GAIT
END

TO ANIM :GAIT
SETSH 6 WAIT :GAIT
SETSH 7 WAIT :GAIT
SETSH 8 WAIT :GAIT
SETSH 9 WAIT :GAIT
ANIM :GAIT
END
```

If you now enter

```
CS
WALK 20 5
```

you should see a far more realistic image of a person walking across the screen than you did at first. Experiment with different speeds and gaits to find other combinations that look proper.

If you are interested in making your images look even more realistic, you should read books such as *The Animation Book,* by Kit Laybourne (Crown Publishers). Although it is oriented to film animators, this book also offers much of value to computer animators.

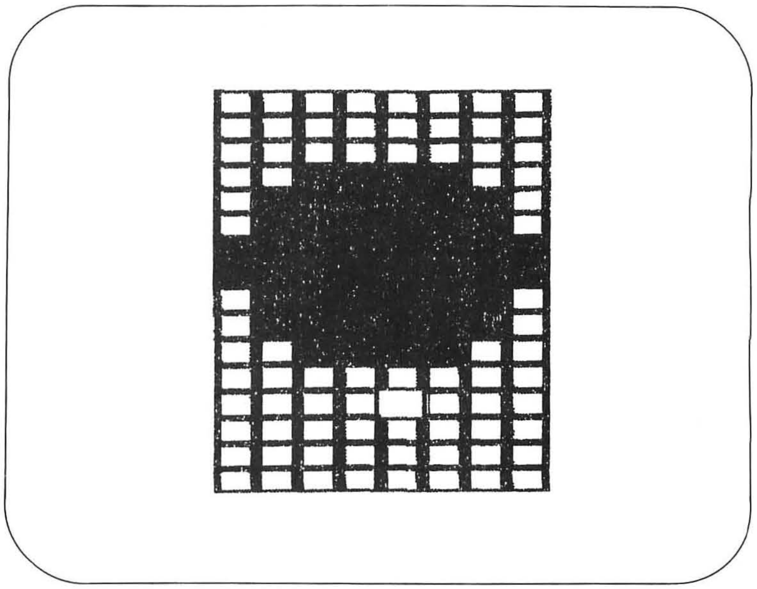
Projects That Use Multiple Turtles

If you have ever tried to create animated sequences using other techniques, you can appreciate what a time-saver multiple turtles and computer graphics can be. Our project of animating a walking figure, for example, would not be a normal starting point if we were experimenting with frame-by-frame animation techniques.

A more common starting point for experimenting with animation is to create a sequence that shows a bouncing ball. Using the techniques covered in this chapter, write a procedure that has a ball bounce near the bottom of the screen and go back up again. Next, define a shape for a ball with a slightly squashed bottom and modify the procedure so that, when the ball reaches the bottom, it looks compressed before going back up.

There are several ways to write a procedure to do this. The easiest (and most powerful) way to accomplish such tasks in Atari Logo is to make use of a feature called the "WHEN demon." A WHEN demon is a special Logo object that continuously monitors the computer, looking for any of 21 special events to occur. Whenever one of these events takes place, the "demon" associated with that event executes its own set of Logo instructions, no matter what other instructions or procedures are being used at the time. When these demon instructions or procedures are finished (and the WHEN condition is no longer satisfied), Logo goes back to whatever it was doing before the demon procedures were used. The best way to understand the power and utility of the WHEN demon is to see it in use.

To make a bouncing ball routine (with a squashed ball), we will first need to define the squashed ball shape. Define shape 2 (using EDSH) to have the following pattern:



This ball is squashed on both the top and the bottom. Next, enter the following procedure:

```
TO BOUNCE
CS
WHEN 0 [SETSH 2 WAIT 5 SETSP -SPEED WAIT 10 SETSH 1]
PU HT SETPOS [-100 100] PD
SETH 90 FD 200
PU SETPOS [-100 -60] PD
FD 200
PU SETH 0 SETPOS [0 0]
SETSH 1 ST
SETSP -30
END
```

Before using this procedure, we will explain how it works. The first line that requires explanation is

```
WHEN 0 [SETSH 2 WAIT 5 SETSP -SPEED WAIT 10 SETSH 1]
```

where the WHEN command is followed by a number and a list of instructions. The number refers to one of the conditions shown in the WHEN demon condition table.

WHEN Demon Condition Table

CONDITION NUMBER	DETECTS WHEN
0	Turtle 0 touches line drawn with pen 0
1	Turtle 0 touches line drawn with pen 1
2	Turtle 0 touches line drawn with pen 2
3	Button on joystick is pressed
4	Turtle 1 touches line drawn with pen 0
5	Turtle 1 touches line drawn with pen 1
6	Turtle 1 touches line drawn with pen 2
7	Each second has elapsed
8	Turtle 2 touches line drawn with pen 0
9	Turtle 2 touches line drawn with pen 1
10	Turtle 2 touches line drawn with pen 2
11	(Not used)
12	Turtle 3 touches line drawn with pen 0
13	Turtle 3 touches line drawn with pen 1
14	Turtle 3 touches line drawn with pen 2
15	Joystick position is changed
16	Turtle 3 touches turtle 0
17	Turtle 3 touches turtle 1
18	Turtle 3 touches turtle 2
19	Turtle 0 touches turtle 1
20	Turtle 0 touches turtle 2
21	Turtle 1 touches turtle 2

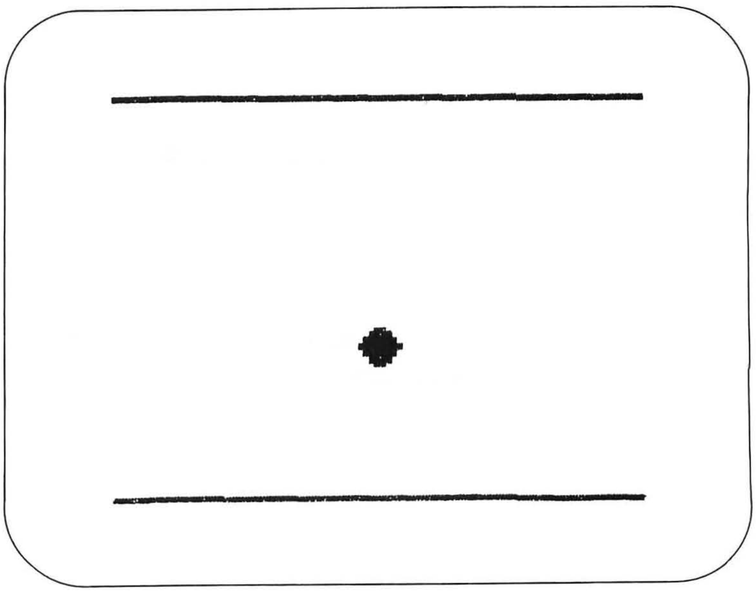
Whenever this condition is satisfied, the list of instructions is executed. As you can see from the table, condition 0 will occur whenever turtle 0 collides with a line drawn by pen 0. Thus, whenever our default turtle touches a line drawn with the default pen, the WHEN demon will execute the list of commands shown, no matter what other commands or procedures Logo may be executing at the time!

The commands we have chosen replace the round ball with the squashed one for a short time and then reverse the direction of the turtle motion by changing its speed (given by the Logo function SPEED) to its negative value.

WHEN demons must be created when the computer is in either split-screen or full-screen mode. Once a demon is created, it remains active until you return it to its inactive state or clear the screen with the CS command. To return a demon to an inactive state, you just enter, for example:

```
WHEN 0 []
```

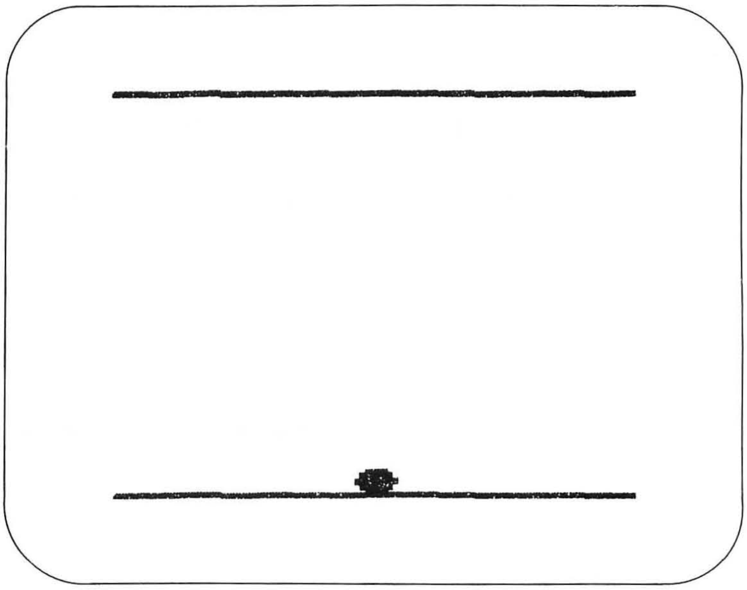
The next six lines of the BOUNCE procedure draw horizontal borderlines at the top and bottom of the screen with pen 0 and place turtle 0 (with the round ball shape) in the center of the screen.



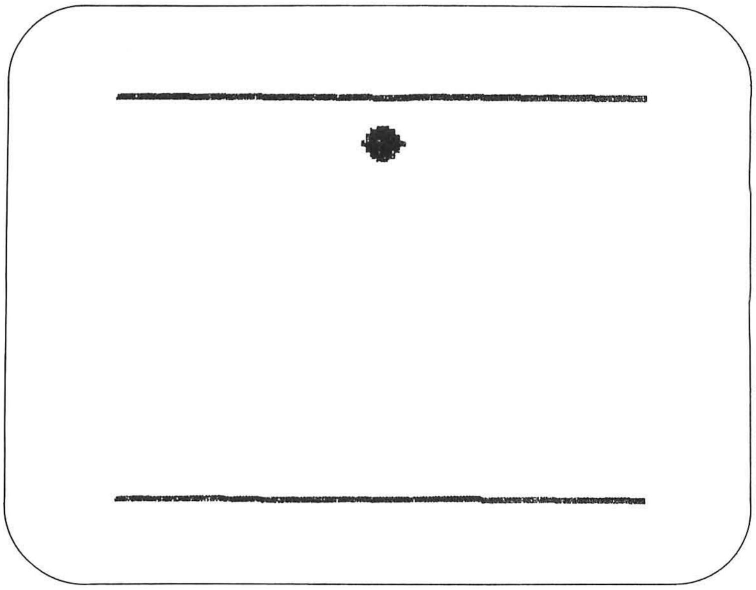
The command

```
SETSP -30
```

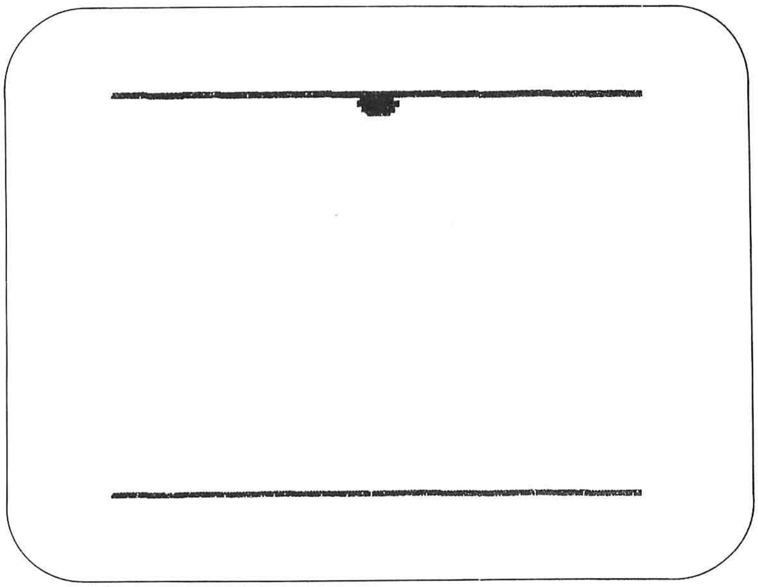
starts the ball moving toward the bottom line. When the ball reaches the bottom, it takes on the squashed ball appearance and starts back up the screen.



The squashed ball quickly restores itself to its round shape as it moves up.



When the ball hits the top line it gets squashed again and starts back down to repeat the process forever—until you stop the procedure or turn off the power to the computer.



To see how automatic the process is, press the BREAK key to try to stop the procedure; notice that it will keep on going by itself until you type something like CS.

WHEN demons allow you to do all sorts of interesting things when your turtles collide with lines or with each other. They can form the heart of many spectacular animation projects.

Since the normal blue background color makes a nice sky, try using multiple turtles to make some fluffy white clouds. If you start out with four overlapping cloud-shaped turtles and have them move apart from each other, you can create changing weather patterns. If you slowly spread the clouds out and then change them from white to gray, you can create the beginnings of a thunderstorm.

Using one turtle for a tree trunk and another to hold the shapes for the leaves, create an animated sequence to depict the changing of the seasons. In the spring, the bare trunk should sprout leaves that become full by summer. In the fall, the leaves should change from green to rust and then fall off. The tree should be covered with snow during winter, and the snow should melt as spring approaches.

As you gain facility with multiple animated turtles, you might want to intermix these shapes with turtle-based background images. By combining graphic techniques, you will discover that you have all the tools you need to create animated stories on your television screen. Your patience will pay handsome dividends—especially when you realize how much easier it is to create animated sequences using computer graphics than it is using traditional animation techniques.

X. The Creation of Animated Sequences

The preceding chapter covered the last of the graphic techniques available to us with Atari Logo. Rather than bringing us to the end of our studies, however, it has finally allowed us to begin! You see, the real task is not just to learn the techniques of computer graphics—that is easy enough to do. The real task is to apply these techniques in the creation of your own artwork.

The creation of static works of art has been covered in the preceding chapters. The creation of animated artwork is sufficiently complex, however, that we will devote this entire chapter to it.

At best, this chapter will serve as a set of hints that might be useful to you. Some of these hints will be more helpful than others. But, as with any creative endeavor, however, you must find your own path—your own method of design and expression. If you have worked with animation before, much of what you already know will give you a head start. If you haven't worked with film or flip books, your freshness to the subject of moving pictures may provide you with insights that might otherwise be missed.

Animation can take many forms, ranging from Saturday morning cartoons to such abstract masterpieces as John Whitney's *Arabesque.*

Whether your sequences tell a simple story or are designed to evoke a purely emotional response, all animated sequences have a design with a beginning, a middle, and an end. If you have a clear idea where your artwork is heading, the result will be successful, no matter how simple its execution might appear. If your sequence rambles or is aimless, your efforts to communicate will fail, no matter how sophisticated your images may be. No amount of experience or technique can compensate for the lack of flow or continuity in a sequence—and that flow and continuity is completely in your hands.

Because of the importance of knowing where you are going with a piece, we will spend some time exploring a major tool of the animator—the storyboard.

Storyboards

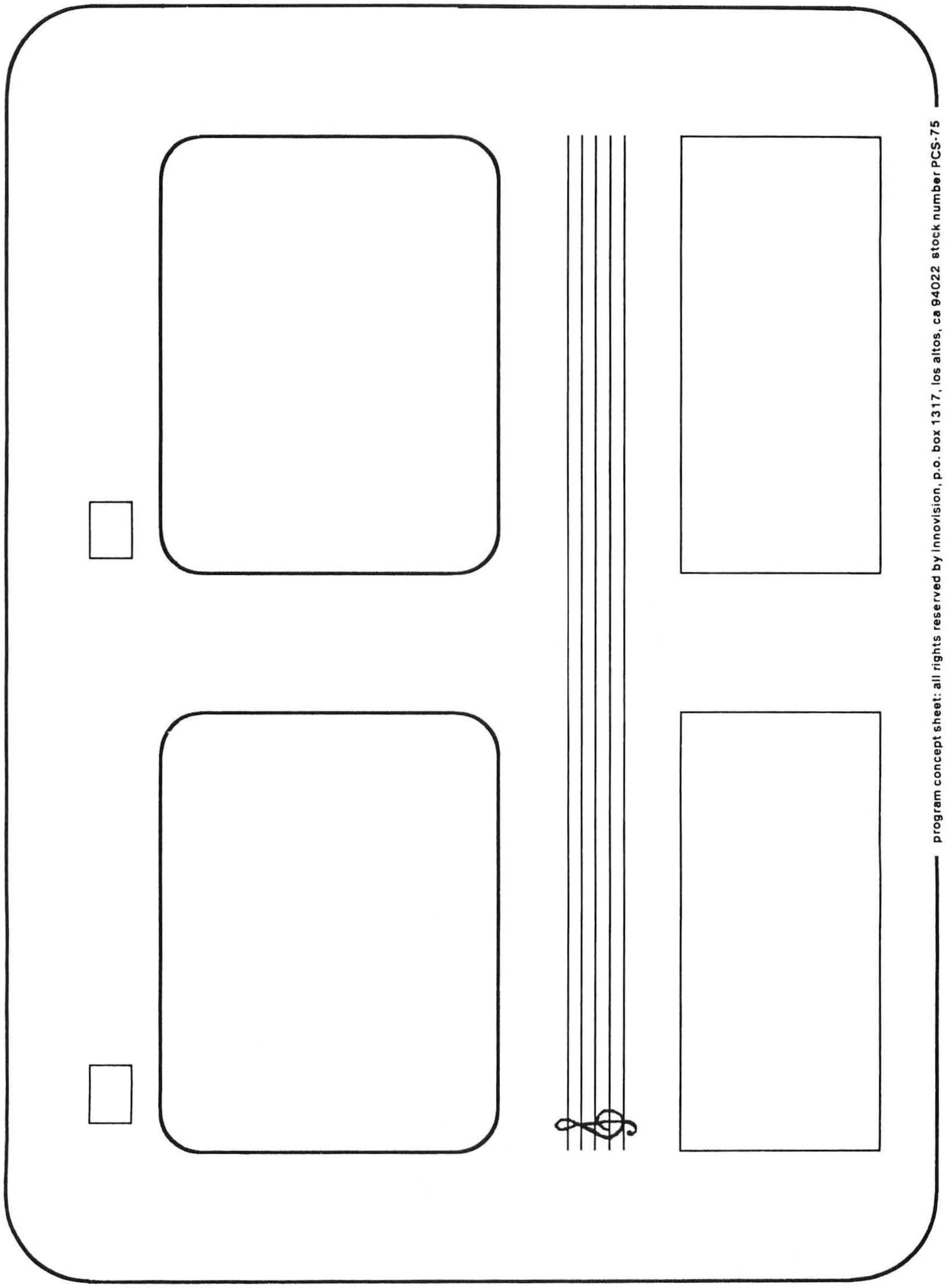
Of all the tools available to the animator, the simplest and most valuable is the storyboard. A storyboard is a conceptualizing tool that lets you examine an animation project scene by scene. To make a storyboard, you create a series of small sketches of key frames and tack them up on a wall, where the project can be looked at as a whole. Each element of a storyboard is a sketch of one frame. These sketches may be quite simple, showing only the general background and figure placement. The goal is not to make a perfect model of the final frame, but to do a quick rendering that pins the key concept for later reference.

How many frames you should sketch depends on the project. Generally, you will find that you will want to see the starting and ending positions of each motion sequence. The storyboard can help you make smooth transitions between scenes and will serve as the outline for your piece.

Although just about any bits of paper can be used for your storyboard, you might want to copy the format shown on page 193. This format has room for a starting and an ending frame in the curved boxes, a musical staff for notating sound effects or other musical additions, and two rectangular boxes at the bottom in which you can write descriptions of the action you want to create or in which you can make other notes to yourself.

Depending on the length of your story, you may have a few key frames or you may have over a hundred. No matter how many you have, it is important to complete the storyboard before beginning the task of creating the final animation sequence. Remember that the storyboard is a practical, not an aesthetic, tool, so don't labor over your sketches.

When you have tacked your images on the wall, look at the result carefully. Do the sequences lead smoothly into each



other? Is there a sequence that requires a special technique you should master first? The overall flow of your story should be apparent from your storyboard. If potential viewers can't figure out what you are doing from the storyboard, they probably won't be able to understand the finished project either.

Although the storyboard can help you present your ideas to others, it is primarily a personal tool to help you to clarify your own ideas. Good storyboarding is the foundation of good animation and is thus deserving of some practice on your part.

Starting Projects

A common tendency in choosing an animation project is to pick one that is too complex. No matter how skilled you are as an artist or computer programmer, you won't be tackling a project like Lisberger's *TRON* as your first animated artwork.

Probably the best starting point is for you to do some "studies." These studies should be short animated sequences (usually 10 seconds or less) that will help you create and refine your graphics techniques. The bouncing ball experiment in the last chapter is a good example of a study.

What kinds of projects make good studies? Consider the walking figure sequence we created in the last chapter. We made the figure look more realistic in its motion by going from one shape to a set of four shapes repeated in sequence. Even so, our figure's gait was somewhat stilted. If you analyze the motion carefully, you will find that the head and torso of the figure don't move up and down at all. You might want to spend some time studying walking. Can you make the figure walk on its toes? Can it walk in a slouch? Can it skip like a child? You might have the character go from a standstill to a brisk walk in gradual steps, rather than going immediately to its final speed. Learn how to make the figure walk partway across the screen and stop before going on.

There is no limit to the number of animation sequences that can serve as excellent studies—the bounce of a ball, the splash of a raindrop, the movement of a cloud, the setting of the sun. Each of these can be a powerful learning experience and a source of great pleasure to you.

As you create your studies, keep good records of your procedures and turtle shapes for future use. For the studies, you needn't concern yourself with background illustrations or painstaking detail. You are learning how to analyze and model motion, and that is a large enough task at this point to consume all your effort.

Creating Short Scenes

When you have gained some experience with specific animated sequences of interest to you, it is time to animate some complete scenes. These scenes should also be short—perhaps as long as 30 seconds but no longer than a minute. The goal here is to tell a simple story. You should design a background scene using turtle graphics. This scene will remain stationary, and all the turtles will appear to move in front of it. An alternative approach, commonly used to convey the feeling of motion over long distances, is to have a "moving" object of interest, such as a truck, remain stationary and to have the immediate background (for example, houses and trees along the roadway) move in an opposite direction to the trucks's perceived motion. This creates an excellent illusion of motion and keeps the object of focus at the center of the screen. You may find that this technique is a little trickier to implement than scenes with stationary backgrounds and thus may want to defer it for a later experiment.

As with the studies, strive for simplicity. You will be learning how to integrate your turtles with a background. The background shouldn't be too busy, or attention will be diverted from the animated portion of the scene. Only your own taste and experience can guide you effectively here.

As a guideline, the scene should be sufficiently simple that it all fits in the computer's memory at the same time. The scene should also have a simple theme. An abstract sequence might show the metamorphosis of one geometrical shape into another; or you might want to depict a scene from a haiku. Your story should be clear enough to be understood by anyone who will see it. Try to depict simple events—driving through the woods or walking on a beach with sea gulls flying overhead.

You might want to depict an event of history—such as the apple falling on Newton's head—but don't try something as ambitious as the Battle of Hastings!

In designing your sequence, make heavy use of storyboards. Once the scene is clear in your mind, you are ready to put it into the computer. Generate the background procedures first. Try to develop a few generic shapes that can be used over and over again. You can then build the background much as you would construct something with bricks. Remember that the background should complement and support the action that takes place in front of it. Keep the background simple. If it is too busy, the animation will fail. Choose colors carefully; keep the background in neutral tones, and make sure it isn't distracting.

When the background is finished, save your procedures on a diskette. In fact, you should be saving your files every 30 minutes or so, just in case the power goes out or you accidentally push SYSTEM RESET.

To develop the animated portion of the sequence, you might want to make a first attempt at creating all the turtle shapes you will need. Don't worry if they aren't perfect; you will be able to edit them later. Make notes on your storyboard, assigning a number to each shape as it is created. This will help you later when you go to put everything together.

Next, start writing the animation procedures that will place the turtles on the screen and set them in motion through the action of the scene. Again, this can be done roughly at first. If you know where you want the turtles to go but aren't sure of the speeds, make the speeds into variables that can be specified when the procedure is run.

Finally, use the background procedure to draw its image, and then run the turtle procedures to see how they work. Make note of shapes and motions that seem awkward. Is the motion too sweeping or too stilted? Are the turtles the right shape, size, and color? If one object is shown in several positions (such as a walking figure), does each pattern lead smoothly to the next?

Refine your sequence until it is perfect, and then make a copy of it on your diskette. You should use a separate diskette that will become your portfolio of finished works.

In viewing your completed work, you might feel that it would be enhanced by music or narration. In the next chapter, we will show how to add such accompaniment to your recorded performances.

Music can be a very powerful addition to your animated works. As you listen to music, think about how the mood, color, and tempo fit with your scenes. You will be surprised at how easily music blends with computer animation. I tend to prefer simple rather than heavy music to accompany abstract animation sequences. In particular, I find some works by Debussy, Satie, Ravel, and Varese quite appropriate. You should choose your own favorites, however—jazz, classical, rock—whatever you want. Remember that you are expressing your own ideas, not someone else's. To be honest, your artwork should reflect *your* tastes.

Creating Longer Works

When you have made several studies and have completed a few scenes, you may be ready to tackle a complete story. Again, think small. These projects take a lot of time, and by keeping them tractable you will enjoy your learning experience rather than feeling burdened by it. An animation project with a running time of 5 minutes can be a tremendous effort, but you should be ready for it! As with all projects, plan exactly what you want to do and create a storyboard. Story topics might range from folk tales to an abstract choreography to a piece of music. Remember that all your programming tools are available to you—turtle graphics and the turtles themselves. If you are creating a work that is to fit with a specific piece of music, consider using the joystick to start procedures at exactly the right moment, rather than trying to time everything to run perfectly from scratch.

When you have picked the theme for your piece, create the storyboard and see how the result feels to you. Do you have too many scene changes or not enough changes? Can one background be used in several scenes? How many turtle shapes will you need? How will your layout make a smooth transition from scene to scene? (The technical details of this step will be covered in the next chapter.) As you answer these questions, separate your storyboard into blocks representing specific scenes. Each of these scenes should be complete in itself and probably should be stored as a separate file on the diskette.

Above all, remember that even though your animation project is separated into chunks, these chunks must all flow together as one coherent whole to make the project work. This integration task is very important. You may find yourself spending a great deal of time "tweaking" the end of one scene to make it flow smoothly into the next.

If your project is sufficiently complex, you may find that it will have to be interrupted at the end of each scene to change procedures. The only way to do this smoothly is to record each scene on a video recorder, one at a time, and then play the completed work out as a whole. The next chapter will show you how to do this.

XI. Artwork Recording Techniques

Although the creation of graphic images and animated sequences may provide a great deal of personal satisfaction, you probably would also like to share the results of your efforts with others. Clearly, your computer system is too cumbersome to carry around to show to all your friends, so a more transportable form for your artwork must be found. This chapter describes two ways of capturing your artwork for later viewing—the use of a camera for recording still pictures and the use of a video cassette recorder for recording animated sequences.

Recording Static Images

There are several ways to make color photographs of your computer-generated artwork. The easiest technique is to photograph the image directly from the television screen with a camera. Although many types of cameras can be used effectively, I prefer to use a 35 mm single lens reflex camera, so that I can see the image exactly as it will appear on the film.

To photograph the screen, you will have to use an exposure of 1/30 second or longer. The reason for this is that the image you see on your television screen is formed from two interlaced scanned patterns, each of which takes 1/60 second to form. It takes two scans to build the whole image, so a shutter speed of 1/30 second is the absolute minimum for capturing the entire image. Depending on the brightness of your television screen and the speed of the film you are using, you may find that even longer exposure times are required. Because of the length of the exposure times involved, you must have your camera mounted on a stable tripod to prevent blurring of the images.

The tripod height should be adjusted so the camera lens is perfectly centered on the screen. By centering the lens, you eliminate an undesirable effect called *keystoning*. Keystoning occurs when an image is photographed or projected at an angle. The result is that one edge of the image appears larger than the opposite edge. A square photographed this way will take on the trapezoidal shape of a keystone—hence the name. So remember, if you want your pictures to be as accurate as possible, you will have to adjust your tripod height accurately.

When you take your photographs, you will be capturing everything that appears on the television screen. This includes the image you want as well as fingerprints on the screen and any reflections from other light sources. Before photographing any images, clean your television screen with a damp cloth and a mild soap. If the TV hasn't been cleaned in a long time, you may be surprised at the accumulation of dirt. Because of the high voltages used in televisions, the display screen usually acquires a static charge that acts like a dust magnet.

When you are ready to start photographing your images, the only source of light in the room should come from the display screen itself. I find that I get my best results at night in a completely darkened room. If adjacent rooms are illuminated, you can seal light leaks around doors with masking tape. Also, if your television has an illuminated channel selector, block its light with tape.

The choice of film type is largely up to you. For color prints, films such as the ASA 100 Kodacolor VR give fine results. For slides, you may wish to use a high-speed Ektachrome (ASA 400) or a slower-speed Kodachrome. The main difference you will notice between these films is that Kodachrome tends to produce brighter and warmer colors than Ektachrome. Experiment with film types to see which you like best.

The distance between the camera and the TV should be chosen so that the computer image perfectly fills the viewfinder. If the image is too small, move the camera forward; if it is too large, move the camera back. As you adjust the focus on the camera, the image may get smaller or larger. When the focused image is the right size, you might want to mark the tripod location on the floor with masking tape to simplify setup the next time you are ready to take photographs.

The next step in your picture-taking experience is to adjust the lens opening for the correct exposure. You should never rely on the light meter built into your camera when photographing a screen image. You will want to control both the shutter speed and lens aperture controls manually. Your best bet is to use a separate light meter and take your readings from the front of the TV screen. You should take readings from several areas of the screen. When you have made your readings, follow the instructions on your light meter for adjusting your camera. If you can't find an appropriate lens opening for a shutter speed of 1/30 second, set the shutter to a slower speed (for example, 1/10 second) and adjust the lens opening to the correct value. You will find that aperture settings in the range of f/4.0 or higher will provide you with some tolerance in focusing. If you have your lens set at an opening below f/2.0, be sure that your camera is well focused before taking any pictures.

The first time you photograph a series of images, you should repeat the same shot several times with different lens openings. Keep a log of your settings, so that when the film is developed, you will know the optimal settings for your camera.

Although the foregoing technique is perfect if you already have a camera, it is rather cumbersome—especially if you are going to take hundreds of pictures. There are special cameras designed just for taking pictures of TV images. These professional cameras are fairly expensive (typically, more than $2,000), but they produce results far superior to images photographed directly from the TV screen.

Some of the images in this book were photographed with an Image Resources Videoprint 5000 system. Other camera systems, such as those made by Lang Systems, Inc., operate in similar fashion. Rather than displaying the image on a color television tube, these systems electronically separate the display signal into its red, green, and blue components. Each signal is then displayed on a black-and-white monitor, one at a time. A color filter wheel is placed between the display screen and the camera lens. When the red image is displayed on the black-and-white screen, the image is recorded on the film after passing through the red filter. This process is repeated for the green and blue filters. By exposing each component separately, the full-color image is reconstructed on the film. Since color filters can match the primary colors far more accurately than can the phosphors used in color televisions, the result is a photograph with far greater brilliance and depth than can be obtained otherwise. You would want to be quite sure of your serious interest in computer graphics before investing in such a system, but the results easily justify the cost for the serious graphic artist.

Once your images are on film, you can continue to be creative. Let's suppose, for example, that you have made an 8″ × 10″ black-and-white print of a turtle graphic image of a squiral. If you have access to a thermal copier, your local art supply store might be able to supply you with a heat-sensitive medium that would allow you to transfer the image to a silk screen for about a dollar. Once this screen is made, you can print the image, using traditional screening techniques, in any color, and you can transfer the image to paper, wood, or fabric. You can even make your own computer graphics T-shirts.

As with the design of the images themselves, you should let your imagination run free when you think about things to do with the photographed copies of your computer-generated artwork.

Recording Moving Images

Besides the television set itself, one of the most popular entertainment products seems to be the video cassette recorder (VCR). Normally, a VCR is used for playing back prerecorded movies or for recording programs that you would like to view at another time. By connecting your computer system to a VCR, however, not only can you capture your images on tape for others to see, but you can string together several short animated sequences to create a complete animated film.

There are many advantages to using a VCR to record your video masterpieces. First, you can see the results immediately and can re-record sequences that don't look right. Second, because of the low cost of videotape (compared to film), you can record hours of computer graphics inexpensively. Also, unlike using a camera to photograph the screen, you can use the VCR in broad daylight. If you don't yet own a VCR, you might want to borrow one from a friend to see how you like using it.

Two major videotape formats are commonly used in the home—Beta and VHS. Because most video recorders have the same overall features, it doesn't matter which format you use. Be aware, however, that these tapes are not interchangeable; you can't play a VHS tape on a Beta machine, and vice versa.

If you are shopping for a VCR of your own (some models are fairly inexpensive), here are some features to look for. To record multiple sequences of animation, you will need a recorder with a PAUSE feature. The PAUSE button immediately stops the tape to allow you to load a new sequence into the computer. When you play the tape back, your sequences will appear with no time gaps between them.

Another feature to look for is direct video input and output jacks. These let you record from your computer directly, without using a radio frequency (RF) modulator. Most people feel that direct video input produces a higher-quality image. To use this feature, you will also have to buy a video monitor cable for your Atari computer, which is available from your dealer. (If you are using an Atari 400 computer, however, you will not be able to use direct video input.) Aside from improved video quality, the use of direct video input lets you use some special-effects equipment to enhance your images.

There are other features that your VCR might have. The ability to freeze frames and make images move in slow motion can be very valuable in letting you analyze your work in detail. VCRs are available in many price ranges, and you should select one that has the performance characteristics you want but stays within your budget.

When you have acquired your VCR (and some blank tape), you are ready to connect it to your computer. The easiest way to do this is to use the TV signal cable and connect it to the VCR antenna terminals, as shown in the manual that came with your video recorder. Your television set should also be connected to the VCR, so that you can see the images as they are being recorded.

Now that everything is hooked up, you are ready to start recording your images. Rather than starting with an animated sequence, we will start with a "slide show" of your static artwork. By now, you probably have quite a few pictures saved on your diskette. To record these images on the videotape recorder, you might first want to make a procedure that displays a title page.

You can be quite imaginative in the creation of title pages—having the title move across the screen character by character or changing colors of characters after they are displayed. The easiest way to make a title screen is to use a procedure similar to the following (but with your own title and name):

```
TO TITLE
TS CT
PRINT []
PRINT []
PRINT []
PRINT [REFLECTIONS ON A SILVER TREE]
PRINT []
PRINT [A COLLECTION OF COMPUTER]
PRINT [GRAPHIC IMAGES BY]
PRINT []
PRINT [AMY DOAKES]
PRINT []
PRINT [1983]
END
```

Starting with this title procedure, design even better ones for your own use. You might also want to create a procedure called THEEND to provide more credits at the end of the show.

Let's suppose that you have several pictures in your computer memory at the same time. For simplicity, we will assume that these are called PICT1, PICT2, and so forth. If you want to record each picture as it is being drawn, you should first make a procedure called NEXT that lets you advance to the next picture by pressing any key:

```
TO NEXT
MAKE "KEY RC
END
```

Load the tape in the VCR, make sure it is rewound, and then enter

```
TO SHOW
TITLE NEXT
PICT1 NEXT
PICT2 NEXT
.
.
.
THEEND NEXT
END
```

If you now enter

```
SHOW
```

the screen will clear and you will see your title image on the TV screen. At this point, press RECORD and PLAY on your VCR (or whichever combination of buttons is needed to start the recording process for your machine). After enough time has elapsed to be sure everyone can read the title page, press any key on the computer to advance to the next picture. Keep each picture on the screen long enough for everyone to look at it before advancing to the next image. Keep repeating this process until you have come to the end of your show.

If you have more pictures to add to the tape, press PAUSE on the VCR and load the new pictures into the computer. When you are ready to start again, press the PAUSE button again and you can pick up where you left off.

After completing your first recording, rewind the tape and play it back. Pay attention to details. Are the pictures on the screen for the right amount of time? Are they on too long? Is there one picture that isn't shown long enough? Are the pictures in the correct sequence? Try recording the same set of pictures in a different sequence to see if the result is better or worse. Remember that the advantage of the VCR is instant playback; use this feature to help you perfect your technique.

After you have successfully recorded some static images, you should record some of your animated sequences. Start by taking some of your studies and recording them with a title screen at the beginning and a closing text screen. When you are comfortable with your recording expertise, you'll be ready to make a tape of a longer animation project using several scenes—a movie!

Before you start to record your movie, make a note of which scenes you are using and which files they are on. Start with the title image and record scene 1. As soon as the action finishes, press PAUSE on the VCR and load your next scene. (Remember that you may have to erase the old scene to make room for the new one.) When the screen is loaded, start recording and release the PAUSE button. Repeat this process until all the scenes are recorded, and then record the closing text frame.

When you are convinced that your masterpiece is visually complete, you may be able to add an audio track to your tape if your recorder has an overdub control. This feature allows you to add music or voice to your recorded tapes through a separate audio input jack on your VCR. Because of differences among models of video equipment, you should check your VCR owner's manual to see just how to use this feature.

At last you are truly on your own! You have mastered computer graphics and animation and have learned how to record your artwork on film and videotape for others to see. As you gain more expertise, you will find that you can use other video accessories to enhance your artwork. Special video equipment, such as titling machines, is now becoming affordable to home video enthusiasts. Such machines let you mix video from a camera with your computer graphic images. Other video accessories let you enhance your images, change colors in subtle ways, and perform other feats of video magic.

Although you have come to the end of this book, you have only begun to explore the world of computer graphics and animation with Atari Logo. As you continue to gain experience and expertise with computer graphics, you will find that this medium opens new worlds for you. If you are new to computers, I hope you have found this medium to be interesting. If you are new to graphic art, I hope you have learned to tap those creative resources buried inside of you.

Above all, have fun!

Index

> $12.95 FPT USA

David D. Thornburg

Computer Art and Animation

A User's Guide to

Atari Logo

Turn Your Computer into an Electronic Easel!

Computer graphics has too long been thought of as the domain of experts—complicated and very expensive. But now, owners of the Atari home computers and the powerful but easy-to-understand programming language Logo can learn to use their computers to create attractive, professional-looking graphics.

Whether you're an artist, a student, or simply someone who is interested in home programming, here is a book to get you started with Atari Logo. **COMPUTER ART AND ANIMATION: A USER'S GUIDE TO ATARI LOGO** will:

- Give you an easy-to-understand introduction to Atari Logo that even a first-time computer user can follow
- Show you how to use Atari's extended turtle graphics with multiple turtles and a set of 128 colors
- Explain how to draw simple geometric shapes that become the basis of complex drawings and designs, with sample projects for you to try
- Guide you in the use of joysticks and the KoalaPad tablet for increased creativity in your artwork
- Get you started on your own static and animated creations

From designing games to building an electronic artist's portfolio, **COMPUTER ART AND ANIMATION: A USER'S GUIDE TO ATARI LOGO** will help you to turn your Atari into an electronic easel!

David D. Thornburg is a well-known educator and enthusiastic advocate of a h
computer revolution. He is the author of several books, including **PICTURE THIS**
THIS TOO!, DISCOVERING APPLE LOGO, and other books in the **COMPU**
AND ANIMATION series for the most popular home computers.

Cover design by Marshall Henrichs

ISBN 0-2

ADDISON-WESLEY PUBLISHING COMPANY