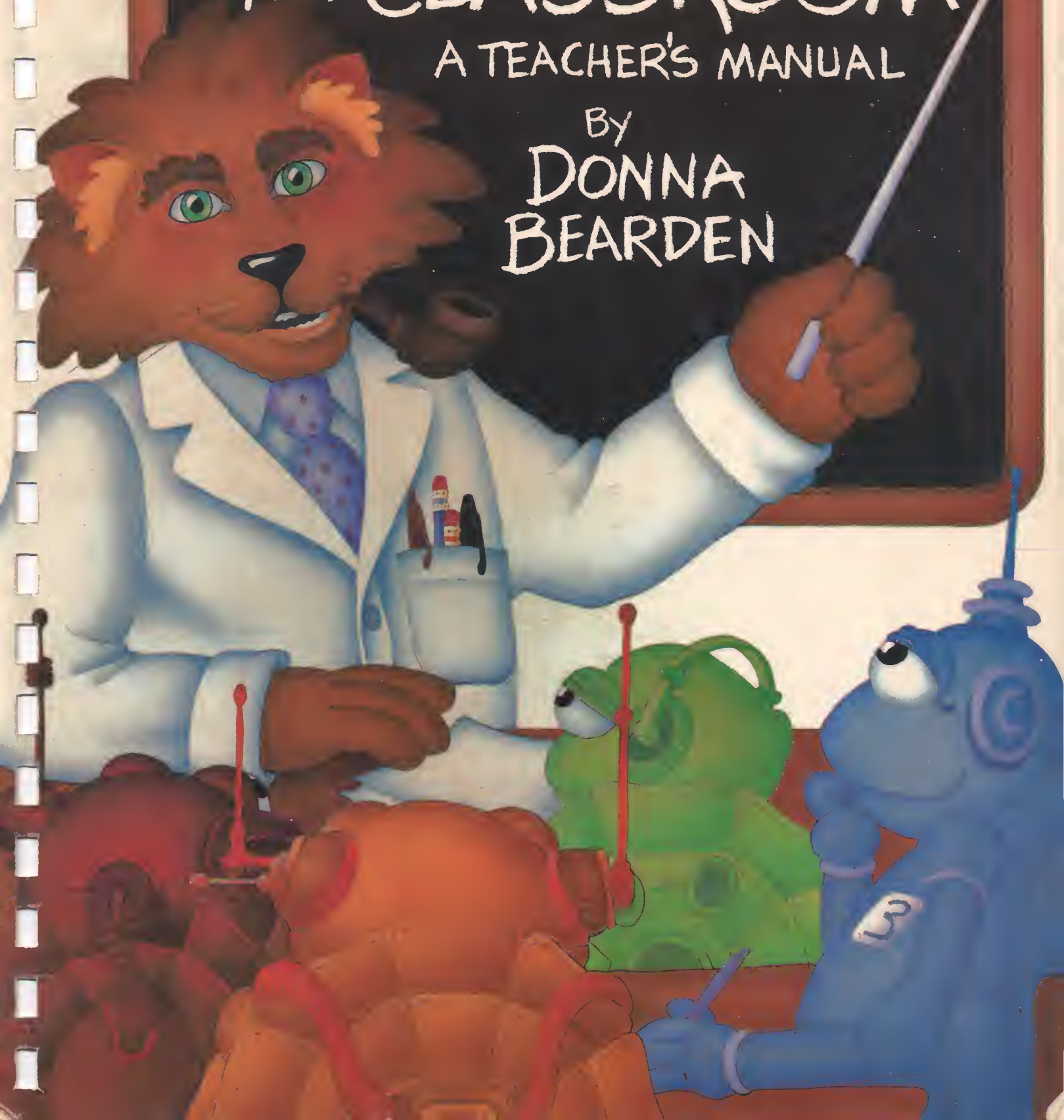


# ATARI LOGO in the CLASSROOM

A TEACHER'S MANUAL

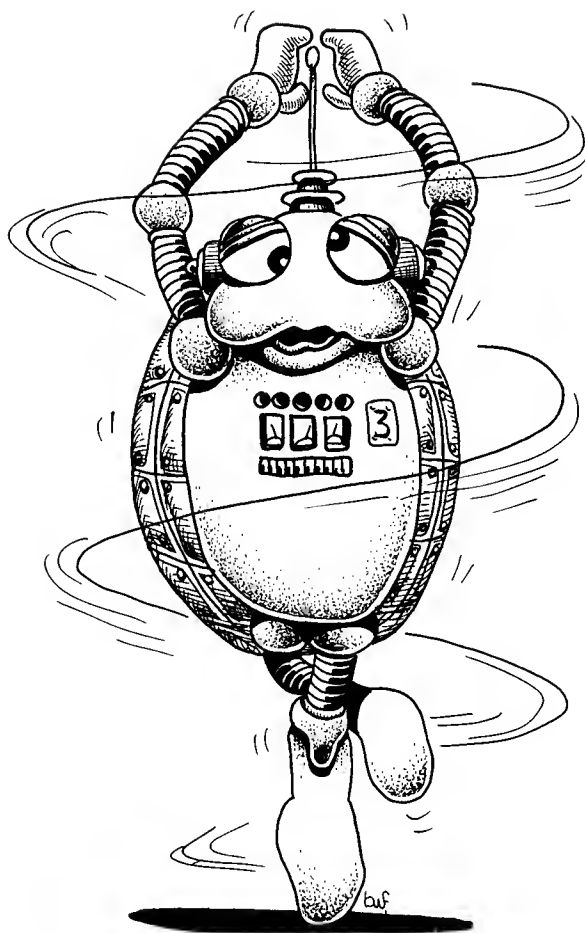
By  
DONNA  
BEARDEN





# **ATARI LOGO in the Classroom**





# **ATARI LOGO in the Classroom**

**by Donna Bearden**

Reston Publishing Company, Inc.  
Reston, Virginia  
*A Prentice-Hall Company*

**ISBN 0-8359-0121-1**

**Text © 1984 by Donna Bearden  
Illustrations © 1984 by Brad W. Foster  
Published 1984 by Reston Publishing Company, Inc.  
A Prentice-Hall Company  
Reston, Virginia 22090**

**All rights reserved. No part of this book may be  
reproduced in any way, or by any means, without  
permission in writing from the authors.**

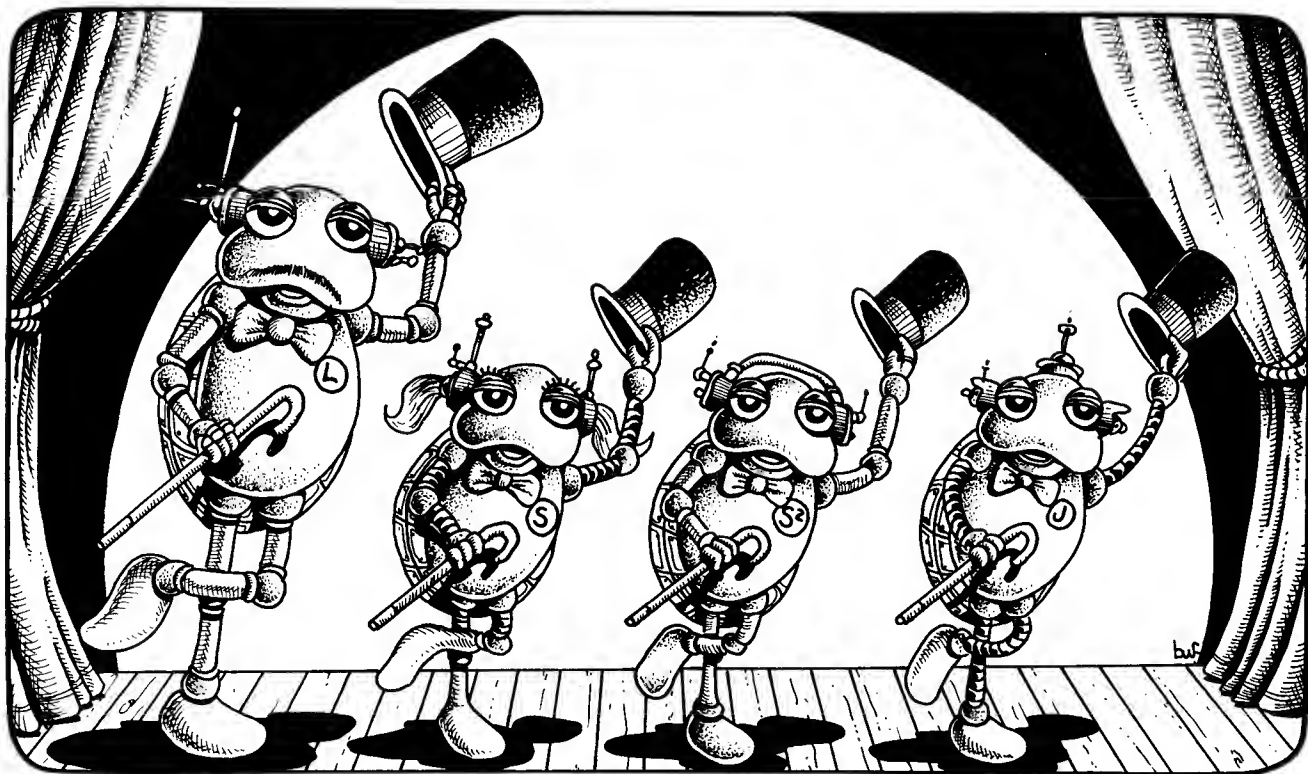
**10 9 8 7 6 5 4 3 2 1**

**Printed in the United States of America**

**This book is published by Reston  
Publishing Company which is not  
affiliated with Atari, Inc. and Atari  
is not responsible for any inaccuracies.  
ATARI is a registered trademark of  
Atari, Inc.**

**Most Reston Computer Group Books are available at special  
quantity discounts for bulk purchases for sales promotions,  
premiums, or fund raising. Special books or book excerpts  
can also be created to fit specific needs.**

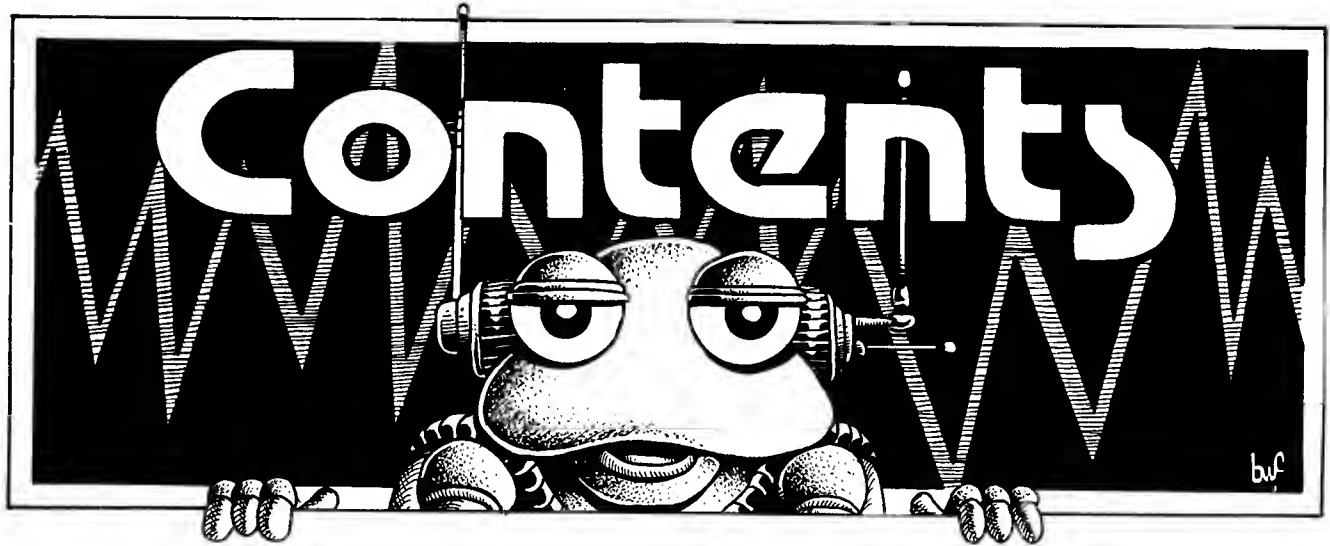
**For details write Sarah Boschen, Office of Computer  
Group Special Sales, Reston Publishing Company, 11480  
Sunset Hills Road, Reston, Virginia 22090.**



...THIS ONE IS FOR MY OWN MULTIPLE TURTLES!

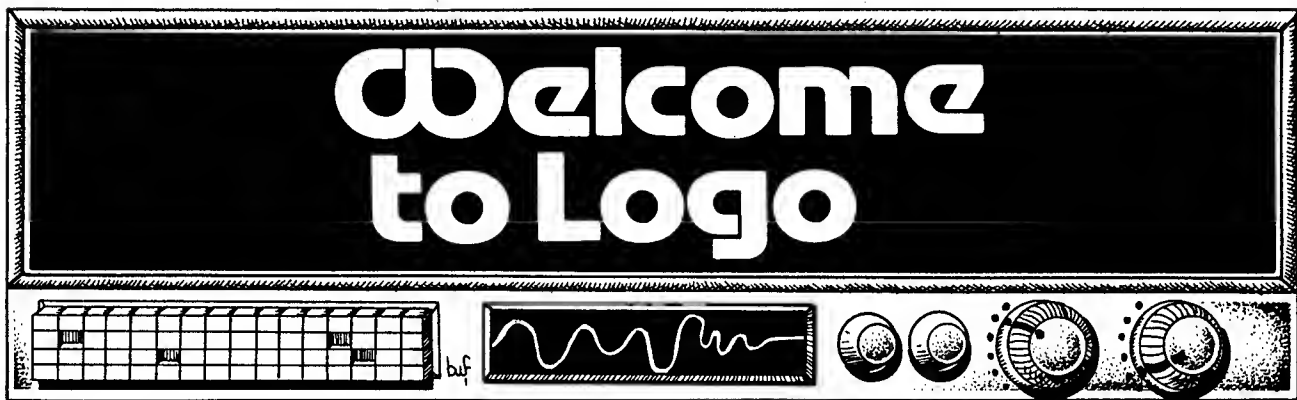






<b>Welcome to LOGO.....</b>	<b>IX</b>
<b>Chapter 1.....</b>	<b>1</b>
Meet the Turtles	
<b>Chapter 2.....</b>	<b>11</b>
Teach the Turtles	
<b>Chapter 3.....</b>	<b>25</b>
More Turtle Tools	
<b>Chapter 4.....</b>	<b>41</b>
The Total Turtle Trip	
<b>Chapter 5.....</b>	<b>59</b>
What in the World is Recursion?	
<b>Chapter 6.....</b>	<b>73</b>
Collision Detection	
<b>Chapter 7.....</b>	<b>87</b>
Turtle Adventures	
<b>Appendix.....</b>	<b>103</b>
Editing Features.....	<b>104</b>
Color Chart.....	<b>105</b>
Table of Collisions and Events.....	<b>106</b>
Musical Tone Frequencies.....	<b>107</b>
Procedures for Jack and Jill.....	<b>108</b>
LOGO Resources.....	<b>112</b>
<b>Index .....</b>	<b>113</b>
<b>Biographies.....</b>	<b>115</b>





Whether you are a teacher, a parent, or a student, consider yourself a student, for LOGO is a language of learning. It is not only a means to learn about computer programming, it is a tool to learn about thought processes, problem solving, and mathematics. It is also a catalyst for creativity.

But it is no more than a tool, no more than a catalyst. We, the students, are the most important ingredients. How we use this powerful tool is up to us. We are the explorers, the thinkers, the problem solvers, the players. And as we are all students, we learn from each other. The teacher and student often switch places.

If we are all students and are to learn from each other through exploration, why publish a teacher's manual? Perhaps a better title might have been "A Guide's Guide," for this, too, was meant as a catalyst.

Thus, this guide is just a guide. While it is a book of ideas for the classroom, it should not be misunderstood to be a curriculum. LOGO is its own curriculum, and students and teachers determine areas to explore. For every idea presented, you who are teachers will recognize how other activities you're already doing mesh with LOGO. You may discover new ways to promote problem solving skills and will modify them to fit your own situations.

Since students' interests and levels are varied, it is helpful to provide for a combination of specific and self-directed projects. Even specific projects, however, should have room for exploration. Worksheets are provided only as ideas for projects. The order in which new concepts and commands are presented will vary from class to class. Be flexible. As students ask questions and experiment, they will develop a need for new information. A musically inclined student may want to explore quite early with the music capabilities of LOGO and will be using some interesting mathematical correlations, whether he recognizes them or not. Some will be ready for recursion much sooner than others. And, horror of horrors, some will get way ahead of you if you let them. Let them! We are all teachers, we are all students.

As you work through this book, keep in mind that there are no right or wrong answers. There are many paths leading to a solution. Some are shorter and more elegant than others. As guides, it is not so much our responsibility to point out these paths, but rather to ask questions of the explorers so they will begin to recognize their own thought processes, come to their own decisions, and be able to see the consequences of those decisions.

Students should keep a record of their projects. This can be a written record in a notebook or a disk containing the student's procedures. If you have access to a printer, have the students print out their graphics and procedures to be mounted and displayed in the room. Even without a printer, pictorial representations can be hand drawn and procedures hand printed to be displayed. This is another way students can learn from and borrow pieces of each other's programs.



One last word. You've all heard about bugs. Bugs are nasty creatures who get into computer programs and cause havoc. The perfect computer programmer is one who rarely has any bugs, right? **WRONG!** The one who rarely has any bugs is the one who rarely touches the computer. Bugs are half the fun in LOGO. It is the bug that points out the bug in our thinking. It is the bug who issues the challenge. Without the bug, there would not be that tremendous sense of accomplishment, that "Aha! Gotcha!" feeling. Don't be too anxious to help remove a bug. Killing the bug may kill the accomplishment. And sometimes bugs lead to entire new paths of discovery. So welcome the bugs!

ATARI® LOGO<sup>1</sup> comes with a reference manual, an introductory programming manual<sup>2</sup>, and a Quick Reference Guide that includes short definitions of ATARI LOGO primitive (built-in commands) and editing features. Information on getting started, loading programs, saving to disk and tape, and other topics is contained in these manuals and is not duplicated here. There are many other good LOGO resources available and a bibliography is included in the appendix. LOGO is as much a philosophy of learning as it is a computer language. For an overview of the philosophy, read **Mindstorms: Children, Computers and Powerful Ideas**<sup>3</sup> by Seymour Papert, the father of Logo.

Above all, have fun. The best learning takes place when we are enjoying what we are doing. LOGO is a fun experience and a powerful learning tool.

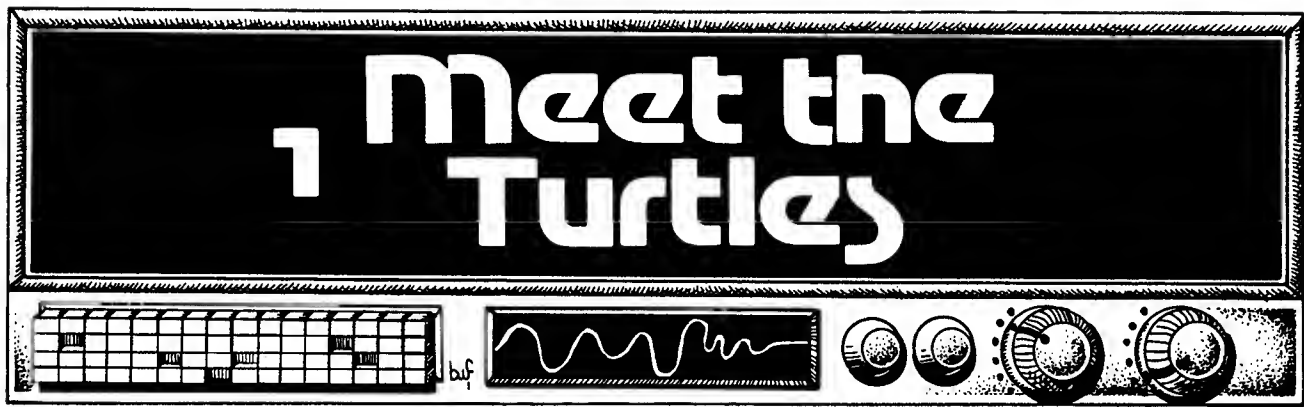
<sup>1</sup> ATARI is a registered trademark of Atari, Inc.

<sup>2</sup> **Introduction to Programming Through Turtle Graphics.** Canada: Logo Computer Systems, Inc., 1983.

<sup>3</sup> Seymour Papert, **Mindstorms: Children, Computers and Powerful Ideas.** New York: Basic Books, 1980.

# **ATARI LOGO in the Classroom**





Unlike other LOGO versions on the market, ATARI LOGO has four turtles (and they even look like turtles!) To see one of the turtles, type:

ST (for Show Turtle).

He's at HOME in the center of the screen. Anytime you want him to return to the center, you can type the command:

HOME

All of the turtles know several commands that will send them on various paths around the screen. They can move fast or slow. They can draw as they move or not draw. They can erase a line at a time or the whole screen at once. They can even disappear from view and still follow commands. And, best of all, they can be taught to remember what you want them to do and the order in which you want them to do it. All in all, they're quite obedient little characters.

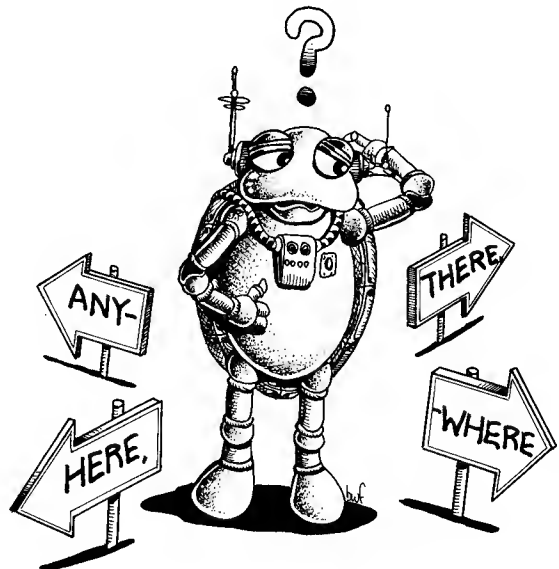
They also don't mind making mistakes. In fact, half the fun of LOGO is making mistakes! Sometimes something quite unexpected will happen on the screen that sends you or a student down a whole new path of discovery. So, don't worry about making mistakes! Make lots of mistakes! Have fun!

Since LOGO is an explorative language, remember to allow time for students to play with each new piece of information, to explore what happens with various inputs. Usually, through their questions, you will know when they are ready for another command or a short-cut. Sometimes you can challenge them with your questions, in essence creating a need for another piece of knowledge.

The first four commands are:

FORWARD or FD  
BACK or BK  
RIGHT or RT  
LEFT or LT

Follow each command with a space and then a number to tell the Turtle how far to go or how much to turn. Then press RETURN.



Have the students experiment with these four commands with lots of inputs, small numbers and large ones. What happens when a very large number is given? (Someone is bound to try FD 10000!) Something really interesting happens when the turtle is turned at an angle and then told to go forward a long way.

Sometimes you may see the message:

### I DON'T KNOW HOW TO...

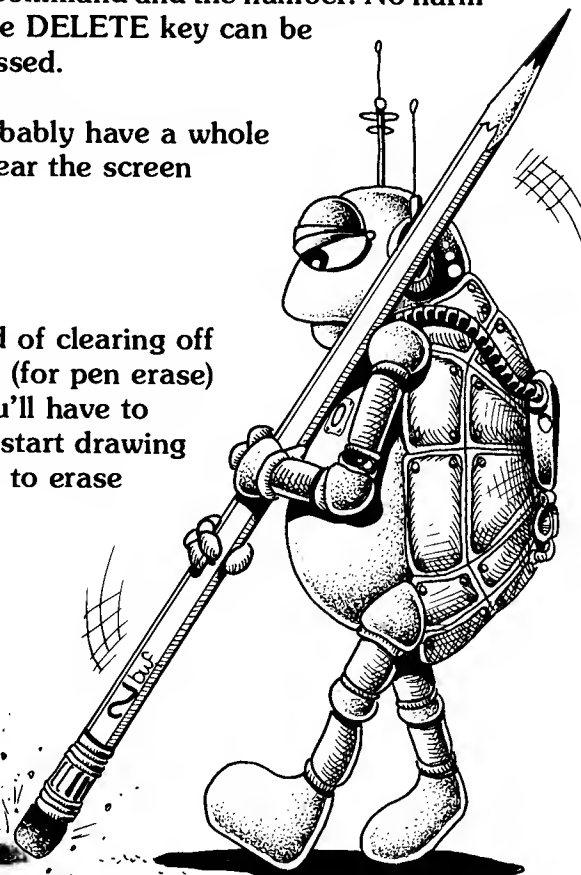
There was probably a typo or no space between the command and the number. No harm done. Simply retype the command and try again. The DELETE key can be used to erase a typo before the RETURN key is pressed.

If you explore very much, pretty soon you will probably have a whole screenful of MESS! At least we hope so! You can clear the screen and start over with:

### CS

Sometimes you only want to erase one line instead of clearing off the whole screen. In that case give the command PE (for pen erase) and then go back over the line you want erased. You'll have to give the command PD (pen down) to have the turtle start drawing again. For example, here's a sequence of commands to erase a line:

```
FD 50  
PE  
BK 50  
PD
```



If you want to move the turtle without drawing, tell him to pick his pen up:

### PU

Remember to give the pen down (PD) command when you want him to draw again.

### Playing Turtle

Young children especially benefit from experiencing turtle turns for themselves. One of the things that sometimes have difficulty with is distinguishing between their own left and right and the turtle's left and right. If the turtle is facing the bottom of the screen and is told to turn RIGHT 90, it will turn to its own right, which is the left side of the screen.

Have one student be the turtle and the others give it commands. This can be made even more fun by putting a blindfold on the Turtle and arranging chairs, tables, or desks in a maze. The turtle must obey exactly in order to move successfully through the maze.

As students become more familiar with angles, they may begin to respond to commands such as RIGHT 30. Start with RIGHT and LEFT turns that are 90 degree turns, however.



## Exploring on the Screen

Before introducing the other three turtles, allow ample time for your students to explore with the commands they have learned so far. The more they can explore, the more adept they will become at estimating angles and distances. As much as possible, allow them to define their own projects. There will be some who need to have specific tasks set for them. Here are a few ideas to get you started:

Figure out the dimensions of the screen and then draw the largest frame you can.

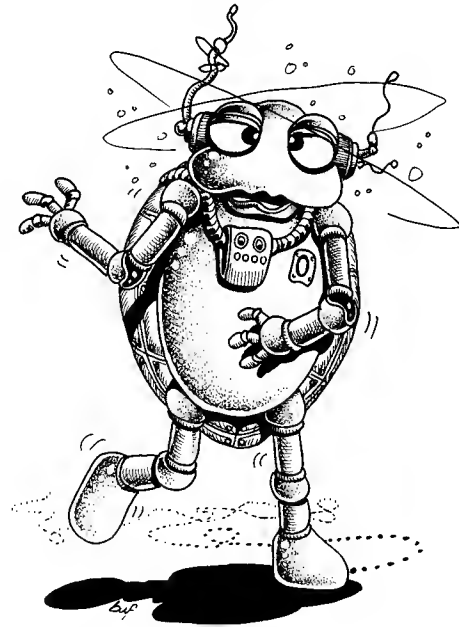
Make the turtle move from the lower lefthand corner to the upper right.

Write your first initial.

Draw the path of a dizzy ant.

Find the highest number of steps you can make the turtle go in one command.

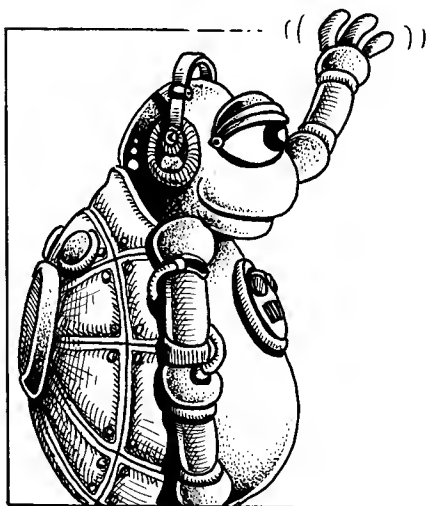
Divide the numbers in your phone number into four commands and make a design by repeating them over and over.



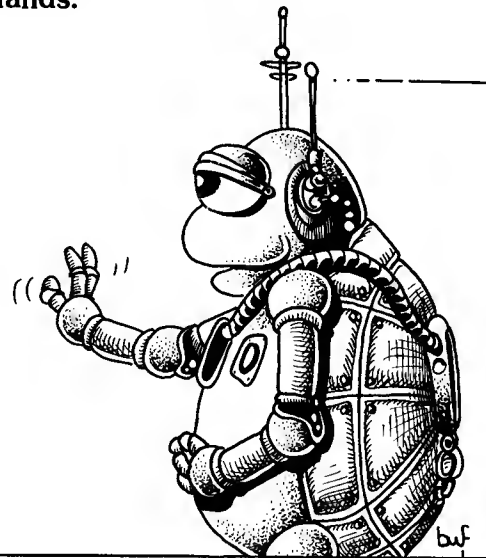
Once your students have become familiar with the basic turtle commands, introduce the other three turtles. The four turtles are numbered 0 - 3. The one you've been "talking to" is Turtle 0. The others are hidden until you TELL them to appear. With your screen clear, give the command FD 40. Turtle 0 will move forward 40 steps. Now give the command:

TELL 1

What do you know! A twin! Give the command FD 20 and Turtle 1 will move forward 20 steps. (As soon as you give the command TELL followed by a number, you are only commanding that particular turtle.) Now give the commands:



TELL 2  
BK 20  
TELL 3



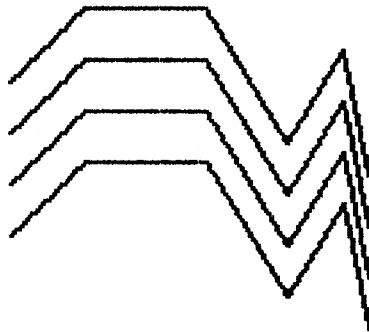
**NOTE:** If you call up a turtle with the TELL command and the turtle does not appear, it has probably been hidden with the command HT (Hide Turtle). Give the command ST (Show Turtle) and the turtle should appear.

You should now have the turtle quadruplets on the screen. You can talk to them one at a time, two at a time, three at a time, or all at the same time. Let's talk to all of them and repeat some of our earlier explorations with four turtles on the screen. First give the command:

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

Be sure to use the square brackets [ ] and not the parentheses ( ).

Now give several RIGHT, LEFT, FORWARD, and BACK commands. Wow! Synchronized swimmers!



When you give the command CS or HOME, all of the turtles will return home. You'll still be commanding all of them, but they'll be stacked up on top of each other and will leave only one trail.

This time let's start them in different directions and see what happens. Leave Turtle 0 pointing straight up. Turn the other three to the other three sides of the screen:

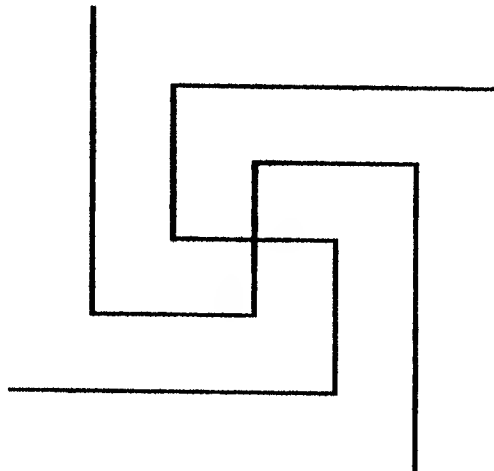
```
TELL 1 RT 90  
TELL 2 LT 90  
TELL 3 RT 180
```

Now give some commands to all of them. Remember to start with the command:

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

Try this:

```
FD 20  
RT 90  
FD 40  
RT 90  
FD 80
```

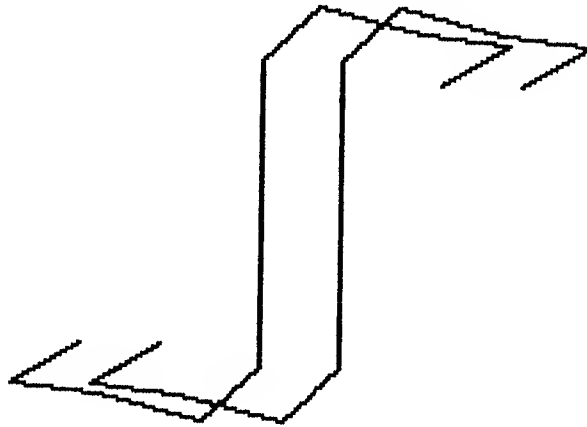


What other designs can you make with four turtles? Suppose you start two of them facing up and two facing down? Something like this:

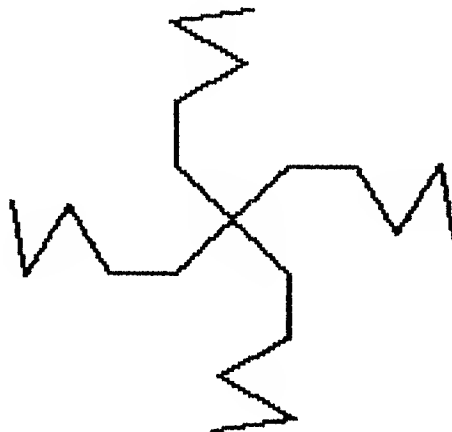
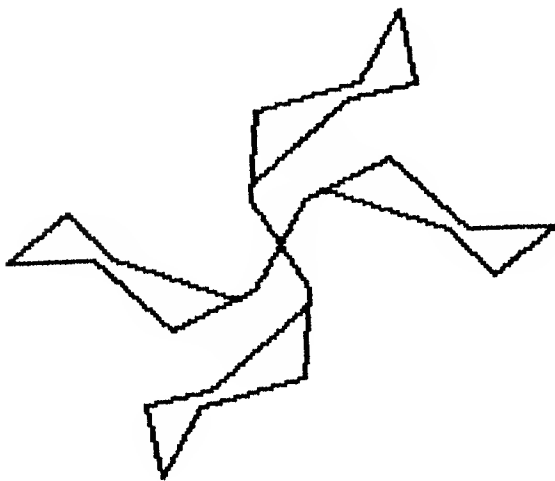
```
TELL 1 RT 180
TELL 2 RT 90 PU FD 20 LT 90 PD
TELL 3 RT 90 PU FD 20 RT 90 PD
TELL [0 1 2 3]
```

(Notice that you can put more than one command on one line.)

Now try a few commands and see what happens.

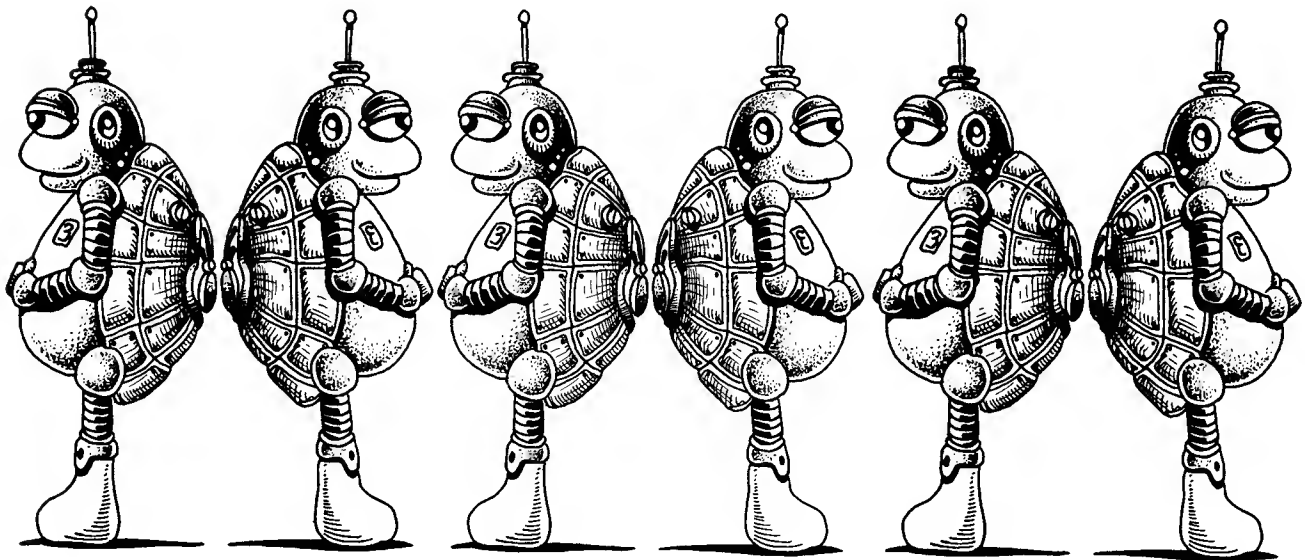


Suppose you...well, suppose you think up a new starting position and try some explorations of your own!



Do you see how this exercise can demonstrate left and right? With the Turtles starting in different directions, have the students draw what they think will happen if they give a **RIGHT 90** and **FORWARD 50** command. Were they right? Try several starting positions and several different commands. Use graph paper to explore various designs that could result from commanding four turtles starting in different directions.

Talk about symmetrical and asymmetrical designs. Use pattern blocks or cut out paper polygons to create symmetrical designs.



### Mirror Images

Have two students stand back to back. One is Turtle 0 and the other is Turtle 1. Give the two students commands to carry out and have the other students observe what happens. If the command **RIGHT 90** is given, the students will see that the two turtles turn in opposite directions, just as they did earlier on the screen.

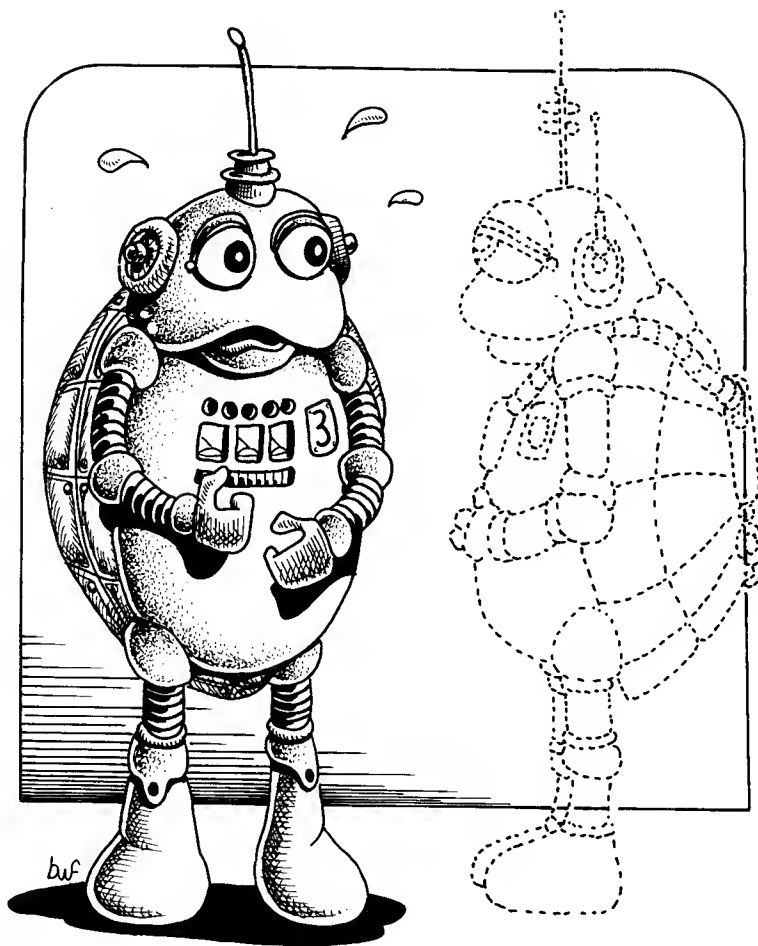
After giving several commands to the student-turtles, discuss the concept of mirror images. Have the two students stand about four feet apart facing each other. One is a regular kid and the other is an image in the mirror. If the kid raises his left hand and scratches his right ear, the mirror image would raise his right hand and scratch his left ear. If one leans to the left, the other leans to the right, but since they are facing each other, they both are leaning in the same direction (toward the blackboard, toward the window, etc.). If one moves forward, the other moves forward. If the first one moves back, the second one moves back.

Have the students describe what they are doing. They should be able to realize that forward and back movements were the same for the kid and the image. Right and left movements, however, were just the opposite.

Once the students have experienced mirror images through their own movements, try it with the turtles on the screen.

Since we'll only be working with two of the turtles, we'll hide the other two. To hide a turtle, give the hide turtle command:

**HT**



To hide Turtles 2 and 3, we could say:

**TELL [2 3] HT**

Have one student command Turtle 0 and another command Turtle 1. The first few commands might be something like this:

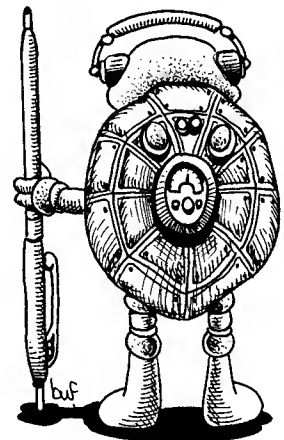
**TELL 0 FD 50 RT 90**  
**TELL 1 FD 50 LT 90**  
**TELL 0 FD 45 RT 45**  
**TELL 1 FD 45 LT 45**

Have the students take turns being the leader. What kinds of symmetrical designs can they come up with using the “mirror turtles?” Is there a way you can expand this activity to include all four turtles?

# Meet the Turtle

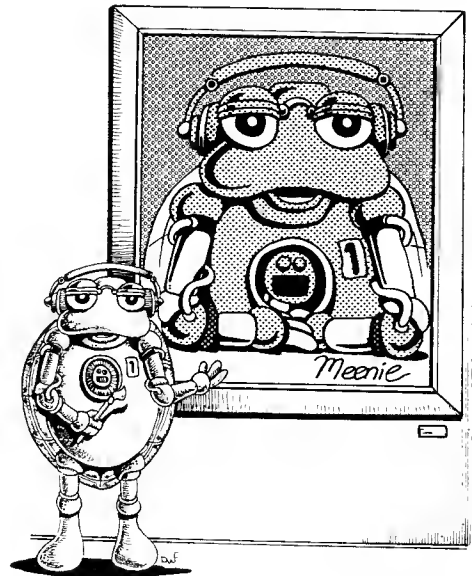
How big is the screen? How far can the turtle go FORWARD before he goes off the top? How far can he go BACK before he disappears behind the TEXT lines? How far is it from HOME to the right hand side of the screen? What about to the upper right hand corner?

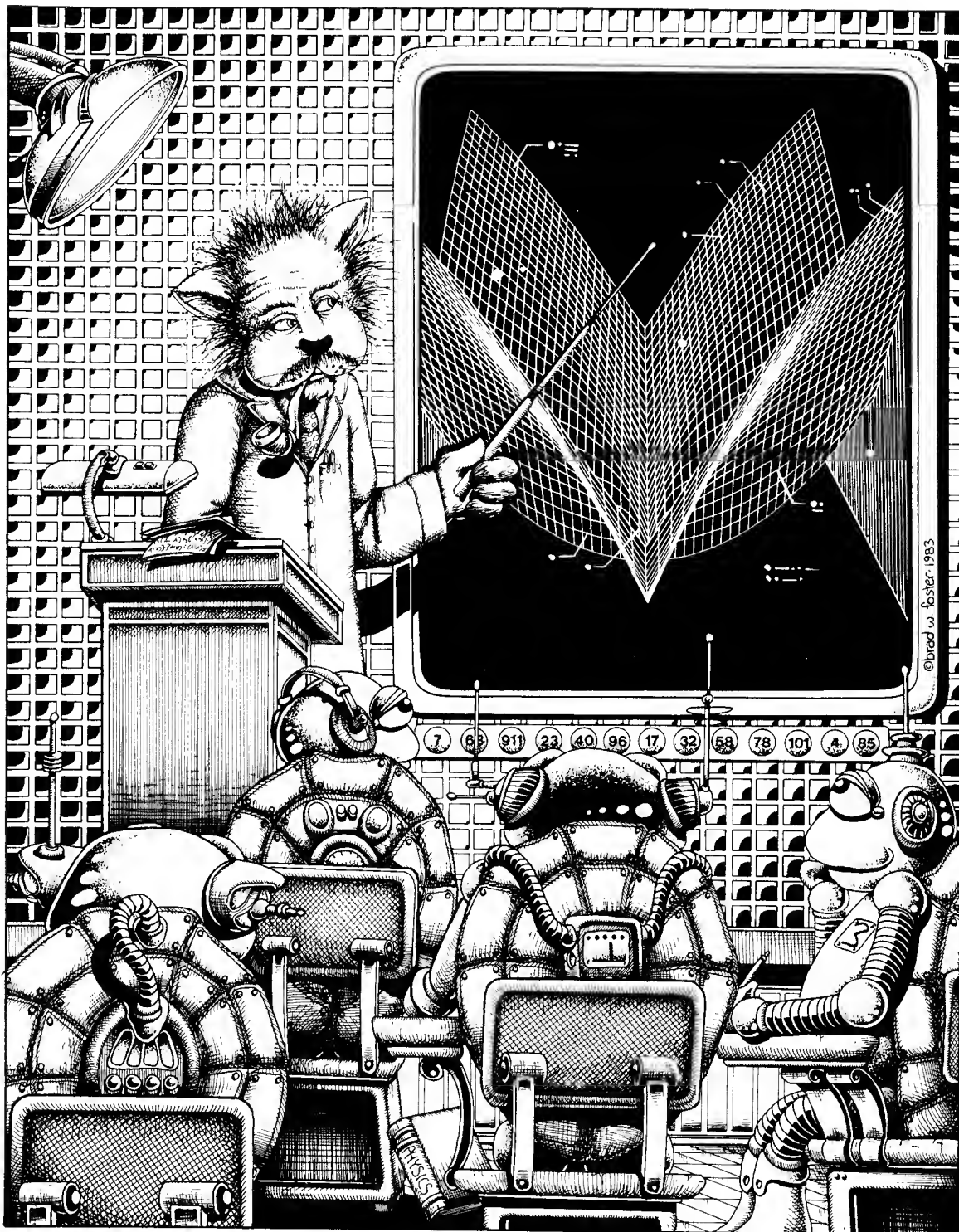
Can you command the turtle to draw your initials? Write down all the commands you use.



# Doodles and Designs

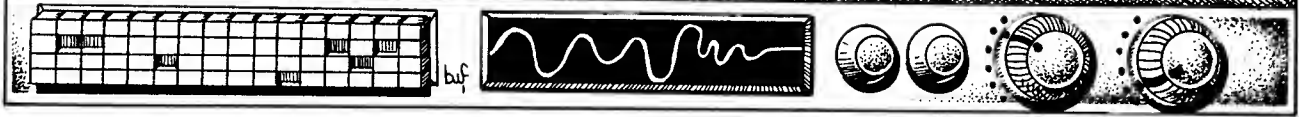
A page for keeping notes of doodles and designs and how they were made.







# 2 Teach the Turtles



By now your students have probably retyped the same sequences of commands enough times to wonder whether or not there's a way to teach the turtles to remember them. For example, if they tried very many experiments with the four turtles starting in different directions, they had to retype the commands to position the turtles each time they cleared the screen. Suppose we could say GO or START or ATTENTION and the turtles would automatically get in the right position. Well, we can, but first we'll have to teach the Turtles what the new command means.

If we *define* a new command or procedure for the turtles, every time we give the command, they will carry it out. Smart turtles, right? No, actually they're just robots doing exactly what we tell them to do — we're doing all the thinking!

In order to define a procedure, type the word TO and the name of the procedure. You can name it anything you want except for a name of a command that is already defined. For example, if you type TO RIGHT, you'll get the message RIGHT IS A PRIMITIVE. (A *primitive* is a command that is built into the LOGO system.) Let's write a procedure to start the turtles in different directions. We'll call it START. When you type TO START, the ? on the left-hand side of the screen changes to a >. Type in the commands to position the turtles, and then type END as the last line of the procedure.

```
TO START
TELL 1 RT 90
TELL 2 LT 90
TELL 3 LT 180
TELL [0 1 2 3]
END
```

Now you can START all kinds of designs quite easily with the turtles facing in four different directions. Simply clear your screen, type START, and press RETURN.

Remember the spiral we started drawing? We could put that in a procedure. Since we want the four turtles to start in four directions, we'll use START as the first command in our new procedure:

```
TO SPIRAL
START
FD 20
RT 90
FD 40
RT 90
FD 80
END
```

Now any time we want to see the turtles make a spiral, we can just type the word SPIRAL and press RETURN.

## Editing Procedures

No sooner do you define something than you figure out a way you want to change it. Maybe you want to add a few more lines to the spiral, or have the turtles turn RT 80 instead of 90.

First let's add two more commands to the SPIRAL procedure and change the 90 to 80. Type EDIT "SPIRAL and press RETURN. (Notice the quotation marks before SPIRAL.) The screen changes and you'll see your procedure in the upper left-hand corner. Across the bottom of the screen will be the words ATARI LOGO EDITOR. By holding the CTRL (control) key down and pressing the arrow keys, move the cursor to the first 90. There are two ways you can delete the 9 and change it to an 8. If you place the cursor right on the 9, press the CTRL key and the DELETE BACK S key at the same time. You can also place the cursor on the 0 and just press the DELETE BACK S key. Now simply type in the new number. Repeat the process with the second 90 in the procedure.

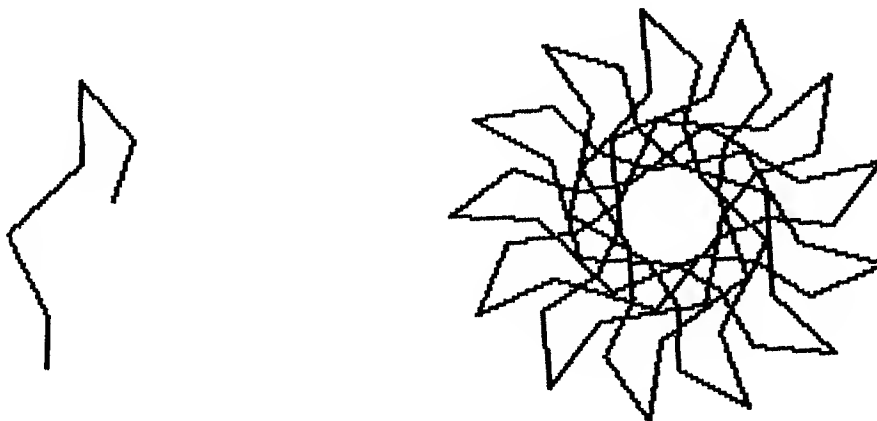
To add a new line of commands, move the cursor to the position where you would start the new line, in this case on the E in END. Press CTRL INSERT and a space will open up for you to add the new commands. Type the new line of commands.

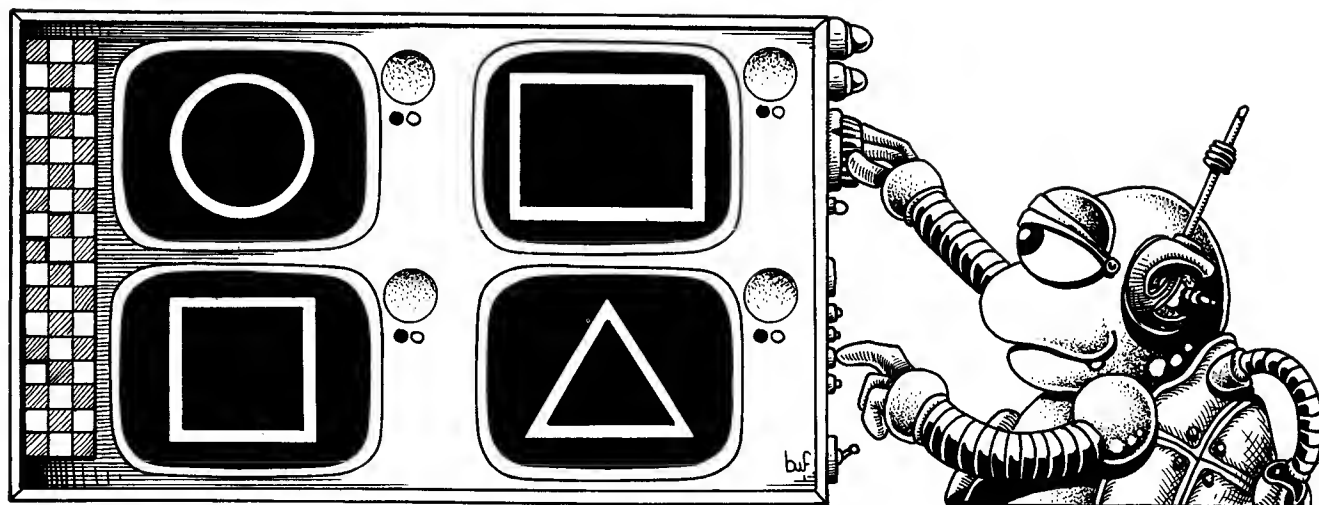
When you have edited the procedure and are ready to leave the editor, press ESC.

As you begin to use the editor more and more, there are a few other editing shortcuts you can learn. They are listed under "Special Keys" in your ATARI LOGO Quick Reference Guide and are included in the appendix. For now, these are probably sufficient.

When students have learned to define procedures, allow time for them to doodle and define. As much as possible, allow them to select their own projects. Even crazy shapes can be repeated several times and will result in interesting designs.

In fact, crazy shapes are helpful in practicing editing procedures. Type EDIT "CRAZY and press RETURN. Now type several commands, alternating movement commands with turning commands. When you have about a dozen commands, type END and press ESC. Try your CRAZY procedure. If it's not too interesting, type CRAZY again and press RETURN. Repeat it several times and it might develop into something. If it doesn't, EDIT it slightly and try again. Play with it.





### Simple Geometric Shapes

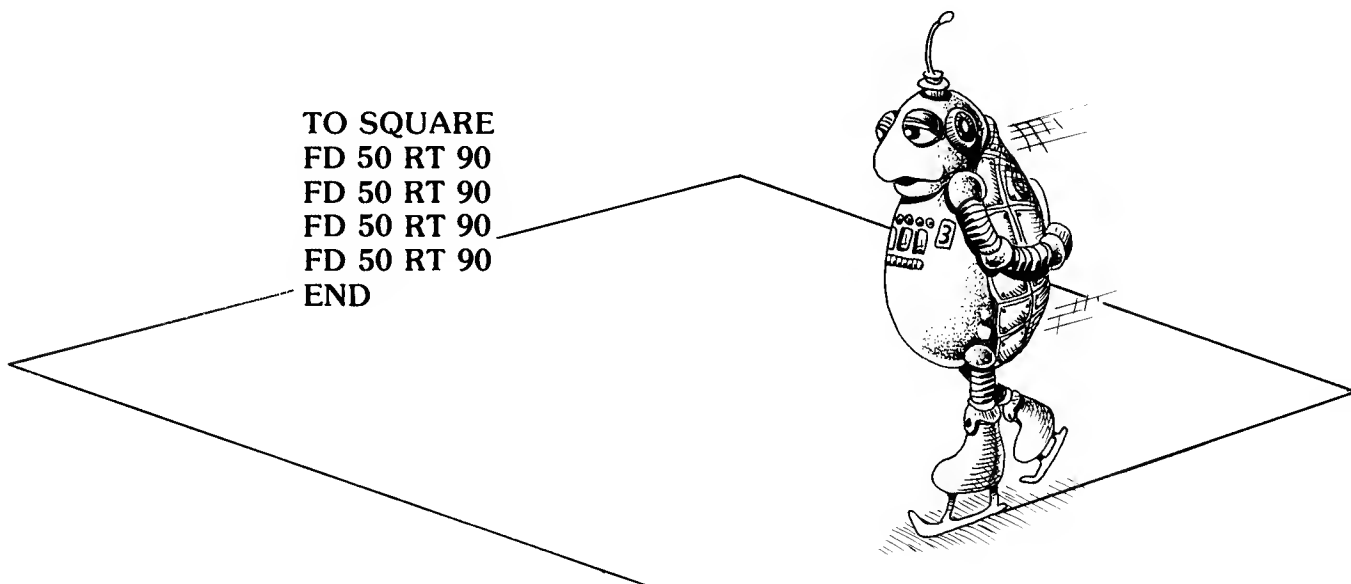
If you can teach the Turtles to draw squares and triangles and circles and rectangles and other geometric shapes, you can draw just about anything you want.

Have your students look around the classroom or take a walk around the block to observe the basic shapes of things. All objects can be broken down into simple geometric shapes. (Beginning drawing books are good resources.) Let's try to teach some shapes to the turtles and then start putting them together to form more elaborate designs and pictures.

Since the square corner or right angle is the easiest to recognize on the screen, most LOGO classes begin by defining a square. It is usually helpful to have students play turtle and "walk a square" first, describing what they are doing. Then have them try to command one of the turtles to draw a square on the screen.

Challenge the students to write a procedure to draw a square using what they've already learned about defining procedures. The turtle should end up in the same position and facing the same direction as he started. The procedure might look something like this:

```
TO SQUARE
FD 50 RT 90
FD 50 RT 90
FD 50 RT 90
FD 50 RT 90
END
```



Do you see that you use the same series of commands four times to draw a square? Whenever you repeat a series of commands several times, there's a shortcut you can use. This one's called a REPEAT command. We could define SQUARE like this:

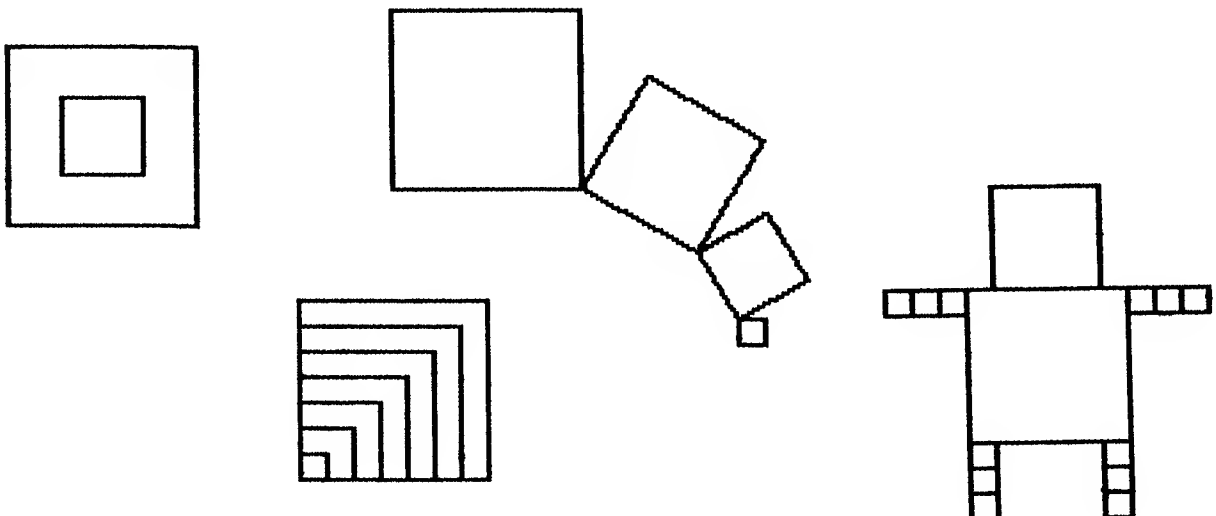
```
TO SQUARE  
REPEAT 4 [FD 50 RT 90]  
END
```

Be sure to use the square brackets [ ] and not the parentheses ( ).



A word of caution. Do not rush to introduce the REPEAT command. It is very important that students understand each step they are going through. Through exploration, they should discover the 90 degree turn and figure out that it takes four right or left turns to make a square. They should also figure out what number to change to make different sizes of squares. It may seem tedious to us to type the same commands four times, but it is through repetition that children learn. Be sensitive to their readiness for the shortcut.

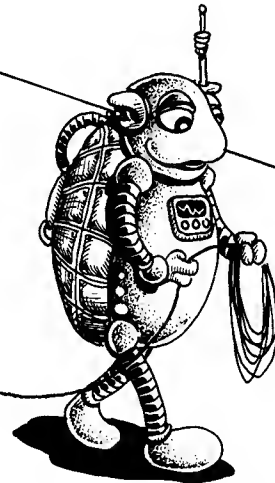
With four turtles to call on, there are many ways to make a square. We've seen an example of commanding one turtle to make a square. How would you command four turtles to each draw one side of a square. How about two turtles working on the same square? Can you make one turtle draw a square and another draw a square around it? What other designs can you make with squares?



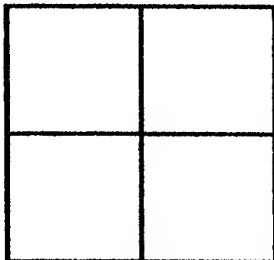
## Rectangles

What is the difference between a square and a rectangle? Can you figure out how to draw a rectangle? Have your students define a rectangle. Once they've learned how to use the REPEAT command, ask them to define a rectangle using REPEAT. Of course there are many different sizes of rectangles. Here's one possibility:

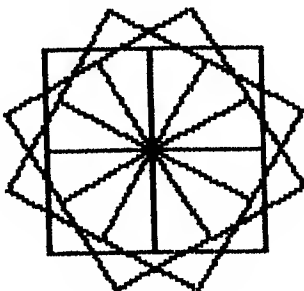
```
TO RECTANGLE  
REPEAT 2 [FD 40 RT 90 FD 80 RT 90]  
END
```



Before we go on to any other shapes, wouldn't it be fun to see what happens if we call up all four turtles, put them in their start position where they're all facing different directions and tell them all to draw SQUARE or RECTANGLE? Can you predict what will happen?



Spend some time exploring squares and rectangles with one, two, three, and four turtles. What kinds of designs can be made with squares and rectangles and additional lines? What happens when you spin a square? Have the turtle draw a square, turn RT 20 or 30, and draw another square. Repeat the square and the turn until the Turtle has returned to its original position.



## Triangles

Although there are many, many different kinds of triangles, we're going to concentrate on the equilateral. Since all of the sides are equal and all of the angles are equal, it is the easiest to create on the screen. Just as with the square, have the students explore with various numbers to discover the turn for the triangle.

Remind the students to have the turtle end up in the same position and facing the same direction as he started. Also, all three turns must be exactly the same. So, if they try RT 100 for the first turn, the other turns must also be RT 100. If they have learned the REPEAT command, have them explore different numbers with it. If they have had enough time to experiment with the square and have made various sizes of squares, they will know that the FORWARD number determines the size of the square. The REPEAT number is the number of sides. Thus, they already know two-thirds of the information they need to make a triangle. It will be REPEAT 3 [FD whatever number they want RT ?]

If they discover the correct number, the turtle should be able to go forward again and retrace the first side of the triangle. As much as possible allow the students to explore on their own. If need be, they can be guided by pointing out what happens if the number they choose is too small (the third line of the triangle doesn't meet the first) or too large (the third line of the triangle crosses over the first).



Too small

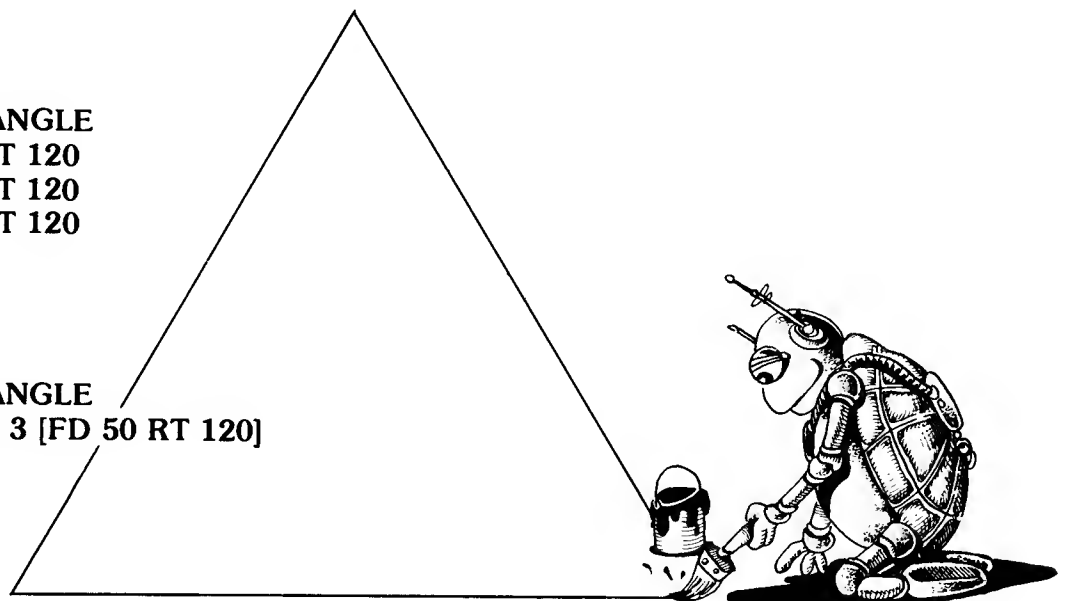
Too large

Define a triangle:

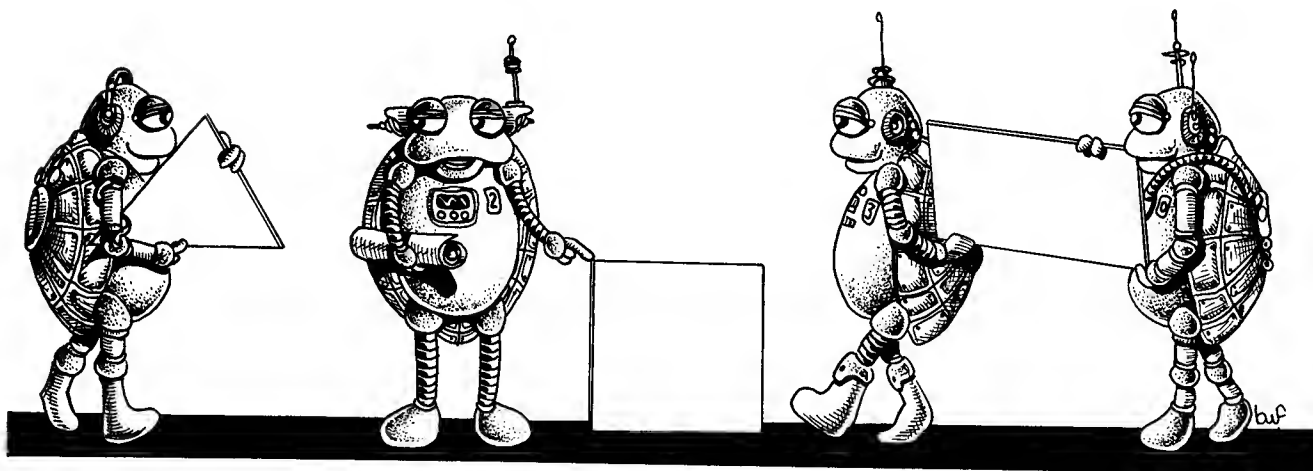
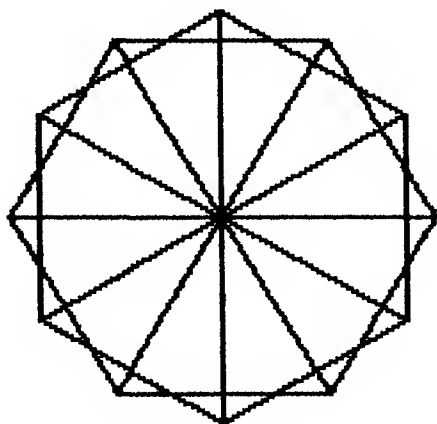
```
TO TRIANGLE  
FD 50 RT 120  
FD 50 RT 120  
FD 50 RT 120  
END
```

or

```
TO TRIANGLE  
REPEAT 3 [FD 50 RT 120]  
END
```

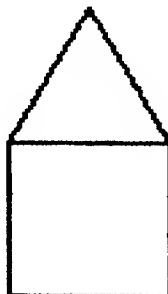


Wouldn't it be interesting to call up the four turtles, give them the command **START** and tell them to draw **TRIANGLE**! What if, after they each draw one triangle, you turn them **RT 30** and **TRIANGLE** again? What happens if you do it once more? If you were drawing this design with one turtle, how would you do it? Write a procedure to draw the **WHEEL** with one turtle and another to draw it with four turtles.

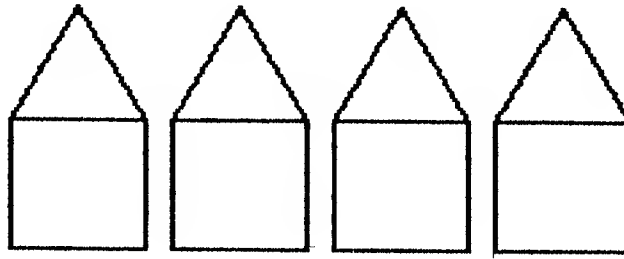


Combining Shapes

Try combining squares and triangles into simple designs. Just as most LOGO classes start with defining a square, it also seems to be a LOGO initiation rite to put a triangle on top of a square to make a house. (It's a good one, so how about a **HOUSE**?)



Once you have house defined, see if you can position the four turtles to draw four houses in a row. How would you create the same neighborhood with only one turtle as the contractor? Can you add doors to your houses? Chimneys? Windows?



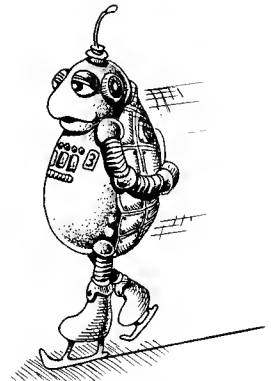
Have your students come up with other simple pictures using squares and triangles. Have them draw them with one turtle or more. Define the pictures in procedures.

Once the students have started generating their own designs, recreate them on paper and hang them up in the classroom. Often one student's design will inspire an idea for someone else. Part of the fun of LOGO is sharing ideas and working out problems cooperatively.



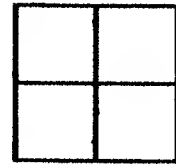
# Squares

Define four different procedures to draw a square. You may use one, two, three, or four turtles.



# Squares and Rectangles

Can you use all four Turtles to draw a window? Use the least number of commands you can.



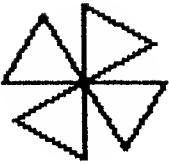
How would you use one turtle to draw a rectangle and another to draw a rectangle around it?



Design something with squares and rectangles and teach the turtle to draw it. Write down what you do.

# Triangles

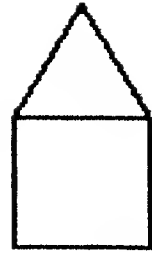
Write procedures to draw the following designs made from triangles.



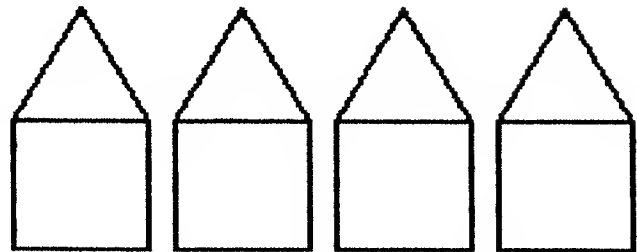
Design something of your own with triangles. Write down what you do.

# Triangles and Squares

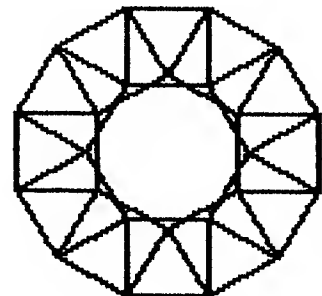
Draw a house with one turtle. Write down what you do.



Draw a neighborhood with four turtles. Write down what you do.

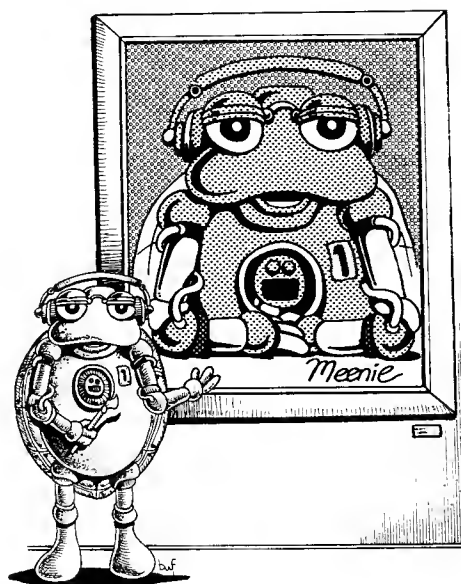


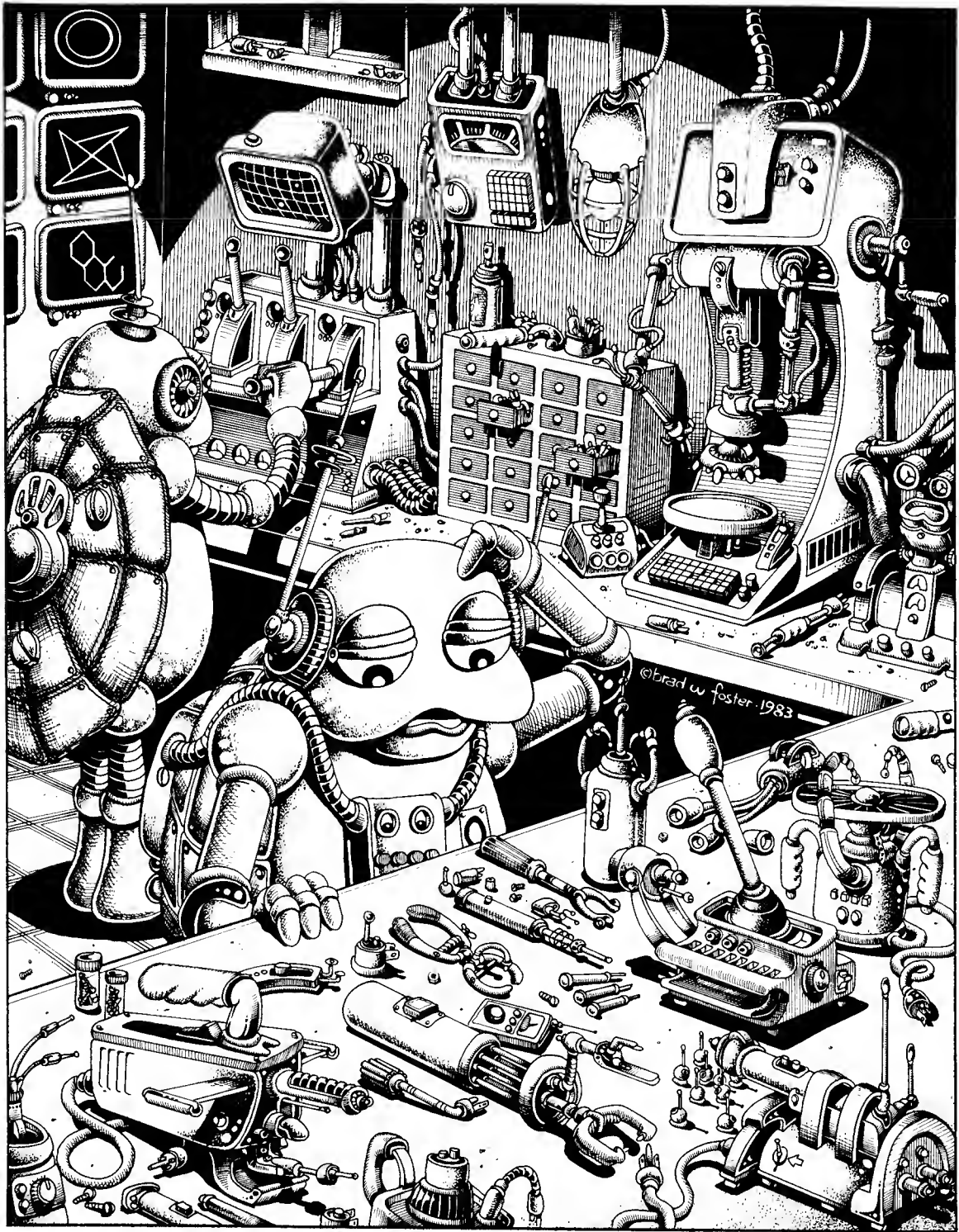
Use the house to draw a wheel. Draw a wheel with one turtle and then figure out how to draw it with two turtles at the same time.

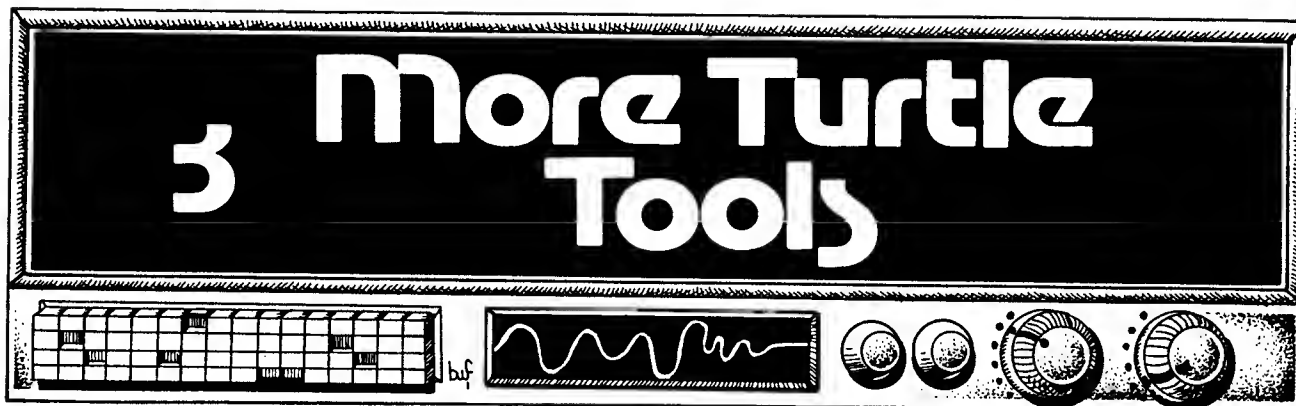


# Doodles and Designs

A page for keeping notes of doodles and designs and how they were made.

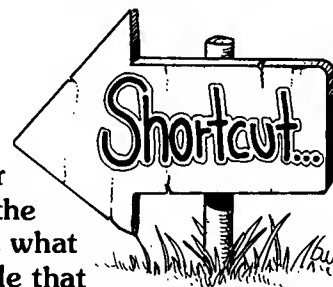






As your students create more and more designs and pictures, they will begin to realize the need to draw many different sizes of a particular shape. By now they may have defined SMALL.SQUARE, LARGE.SQUARE, and MEDIUM. SQUARE. But suppose they need an itsy-bitsy square or a bigger-than-small-but-smaller-than-medium-size square? It gets pretty tedious defining new procedures for the same shape when all you want to do is change the size. It's time for another shortcut.

### Variables

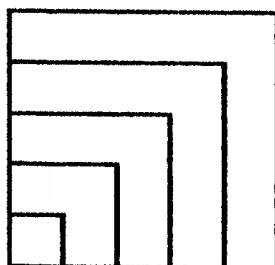


Variables enable you to write one procedure for SQUARE or TRIANGLE or ANYTHING.YOU.WANT. Each time you give the command to carry out that procedure, you'll also tell the turtle what size you want. For example, if you want a SQUARE with a side that measures 27 turtle steps, the command will be SQUARE 27. The number you type following the name of the procedure is called an *input*. The procedure looks like this:

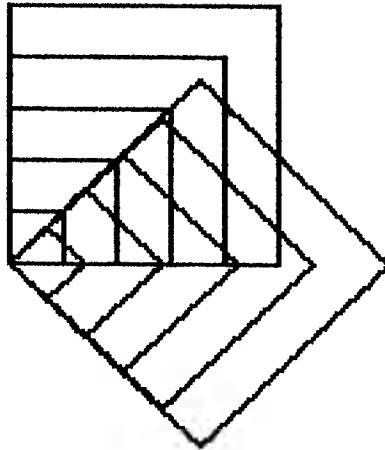
```
TO SQUARE :N
REPEAT 4 [FD :N RT 90]
END
```

The colon, or dots as it's referred to in LOGO, means that whatever follows is a name of a variable. Just as in naming a procedure, we can name variables anything we want. We could say TO SQUARE :SIDE or TO SQUARE :NUMBER.OF.STEPS.I.TELL.YOU. To keep it as simple as possible, use a word that indicates what the variable stands for or single letter. When we get into word games and story writing, we'll use words as names of variables.

In the SQUARE procedure, the :N appears twice: after the name of the procedure and in the FD command since it is the FD command that determines the size of the square. Once your students have defined the procedure with the variable, have them try several inputs. Draw a series of squares each larger than the one before.



What would happen if you draw the series of squares, turned the Turtle RT 45 and repeated the series of squares? Does that give you any ideas?



Remember the house we built in Chapter 2? We could put a variable in that procedure and make all sizes of houses with one blueprint. First we'll have to redefine the triangle with a variable:

```
TO TRIANGLE :X  
  REPEAT 3 [FD :X RT 120]  
  END
```

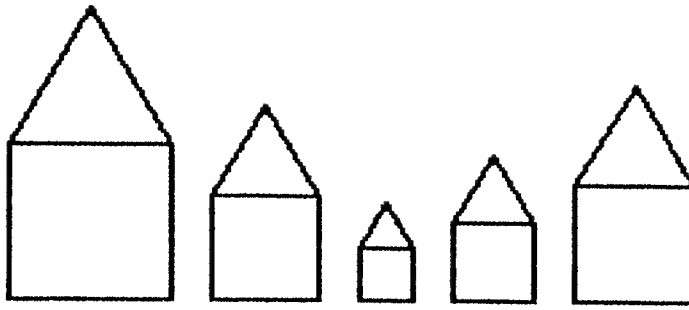
And then the house:

```
TO HOUSE :X  
  SQUARE :X  
  FD :X  
  RT 30  
  TRIANGLE :X  
  END
```



Notice that you can use any letter as the name of the variable.





Can you draw the NEIGHBORHOOD with various sizes of houses? Have your students look at some of the other designs they have created and figure out how to put variables in them to change their size, or draw some new designs that require several different sizes of the same shape.

How could we put a variable into the procedure for a rectangle? In the rectangle we defined earlier, one side was twice as long as the other side. So we'll tell the turtle to make the first side :Y and the second side :Y \* 2. \* is the LOGO multiplication symbol. The computer will do the math for us and the Turtle will make the second side twice as long as the first.

The procedure looks like this:

```
TO RECTANGLE :Y
REPEAT 2 [FD :Y RT 90 FD :Y * 2 RT 90]
END
```

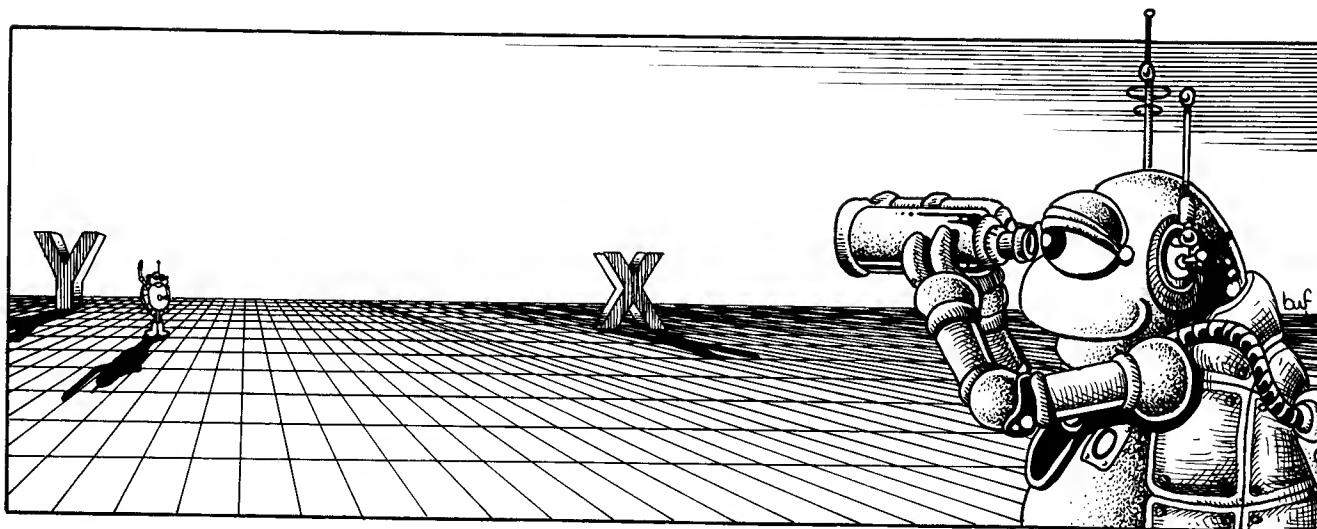
But suppose we want to be able to vary the ratio of the two sides? How about a procedure with two different variables, one for the short side and one for the long side:

```
TO RECTANGLE :A :B
REPEAT 2 [FD :A RT 90 FD :B RT 90]
END
```

In this procedure, anytime we type RECTANGLE, we'll have to follow it with two numbers, once for each of the variables. Now we can draw many different rectangles with one procedure.



Understanding variables and being able to use them is a powerful concept to master. Allow ample time for exploring with this new piece of information. Use a combination of assigned and self-chosen projects requiring several different sizes of the same shape.



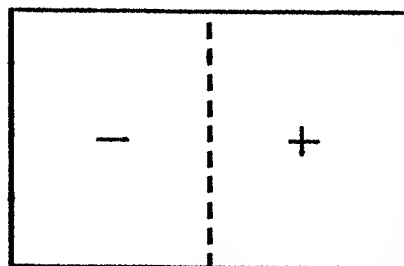
**Turtle Positions**

When you or your students positioned the four turtles to draw the neighborhood, you probably picked up the pens, turned the turtles, moved them, turned them to the correct direction to begin drawing and put the pens back down. In some of the pictures you've created, you probably moved the turtle to some other position besides center-screen to start drawing.

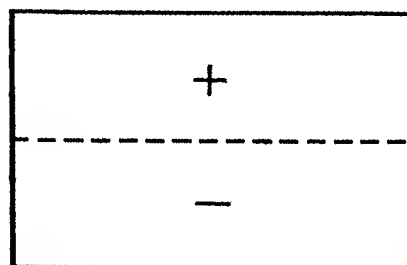
There is another way to move the turtles. It uses the Cartesian coordinate system and offers an excellent way to introduce or practice using the system.

The screen is divided into an invisible grid.

The positions to the left and right of HOME are the X positions.



The positions above and below HOME are the Y positions.

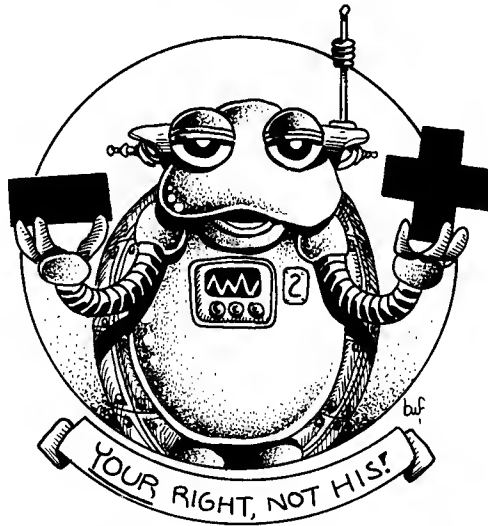


If this is your students' first introduction to the coordinate system, take it one step at a time, first the X positions — both positive and negative, then the Y positions. Later we'll combine them into one command.

In order to move the turtle, first raise the pen (PU). Then give the command:

SETX followed by a number

If the number is positive, the turtle will move to the right-hand side of the screen. If the number is negative, he will move to the left-hand side of the screen.



Have the students figure out the largest positive and negative numbers they can use without having the Turtle wrap to the other side of the screen. This will give them the horizontal dimensions of the screen.

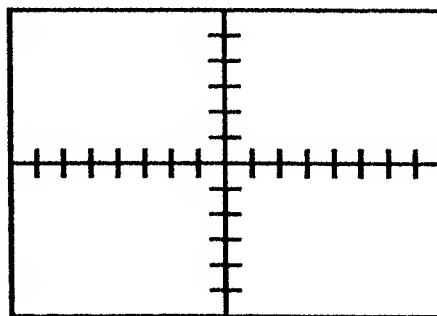
The command for the Y positions is:

SETY followed by a number.

If the number is positive, the turtle will move to a position above HOME. If it is negative, he will move below HOME. Again, have the students figure out the largest positive and negative numbers they can use without having the turtle wrap. This will give them the vertical dimensions of the screen.

If you follow a SETX command with a SETY command, you can position the turtle anywhere on the screen. Try it.

Combine the work on the computer with off-computer activities such as locating specific points on graph paper. Have the students first prepare the graph paper by marking the X and Y axes and numbering them. Then give them a series of points to locate.



A good way to practice with the coordinate system on the computer and to get pretty adept at estimating distances is to play some games. A very easy game to set up is a “target” game. Have one student place small stickers anywhere on the screen. A second student must move the turtle from one sticker to another using the SETX and SETY commands. Limit the number of tries the student may take.

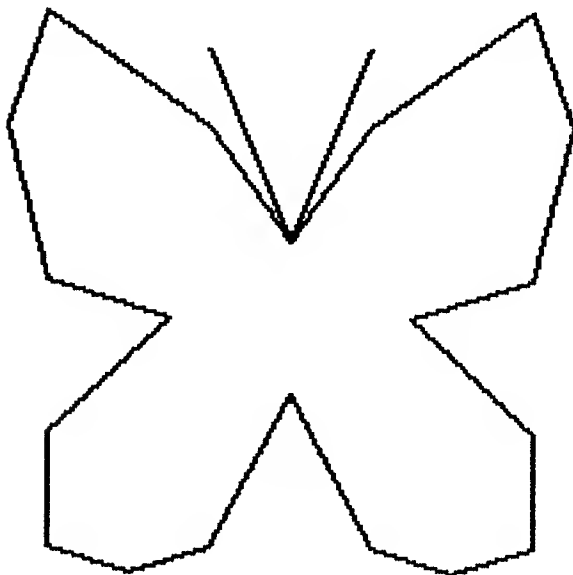
After the students have had plenty of practice with the X and Y commands, they can combine the two commands into one. The SET POSition command looks like this:

SETPOS [50 70]

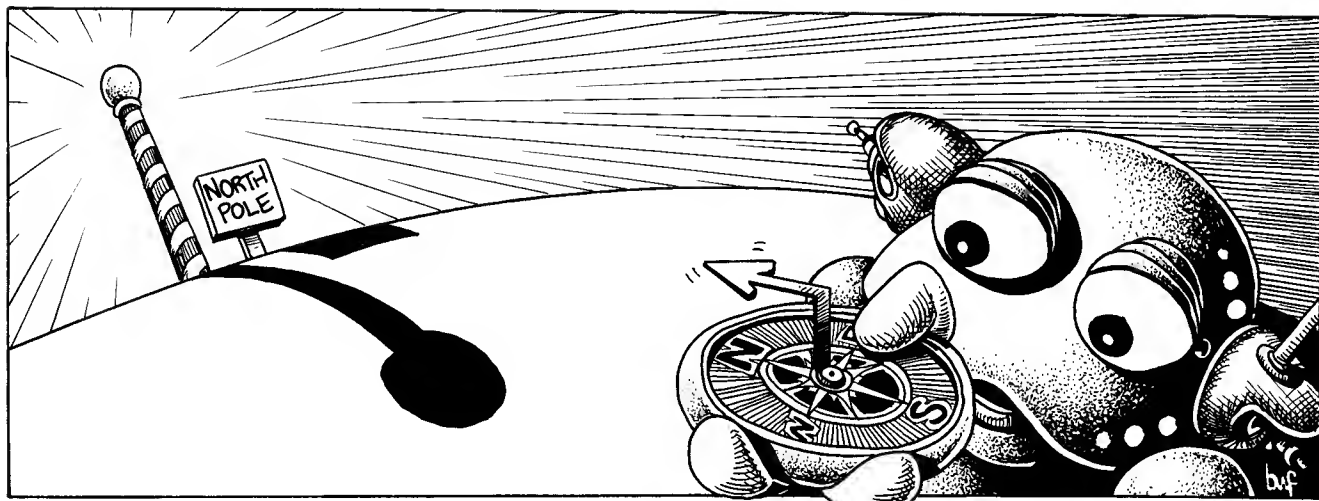
The first number within the brackets is the X coordinate number and the second is the Y coordinate number.

After you introduce SETPOS, have the students play the target game again. This time they will have to judge the X position and the Y position at the same time.

If you leave the pen down when you use the SETX, SETY, and SETPOS commands, the turtle will draw from one position to another. You might be able to draw a picture just by continuing to move the turtle with these commands. Try it with the graph paper first. Plot simple pictures and have the students connect the points to create the picture. Then try it on the computer.

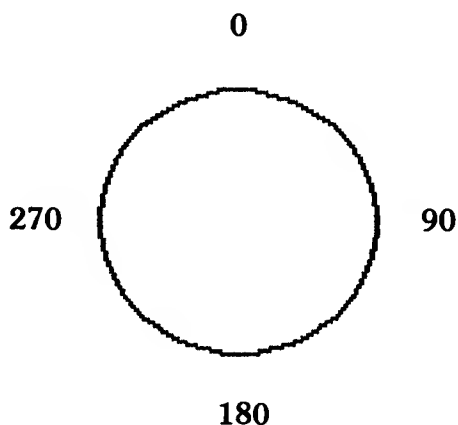


```
TO BUTTERFLY
PU FD 15 PD
SETPOS [30 60]
SETPOS [90 105]
SETPOS [105 60]
SETPOS [90 0]
SETPOS [45 -15]
SETPOS [90 -60]
SETPOS [90 -105]
SETPOS [60 -115]
SETPOS [30 -105]
SETPOS [0 -45]
SETPOS [-30 -105]
SETPOS [-60 -115]
SETPOS [-90 -105]
SETPOS [-90 -60]
SETPOS [-45 -15]
SETPOS [-90 0]
SETPOS [-105 60]
SETPOS [-90 105]
SETPOS [-30 60]
SETPOS [0 15]
SETPOS [30 90]
SETPOS [0 15]
SETPOS [-30 90]
END
```



## Headings

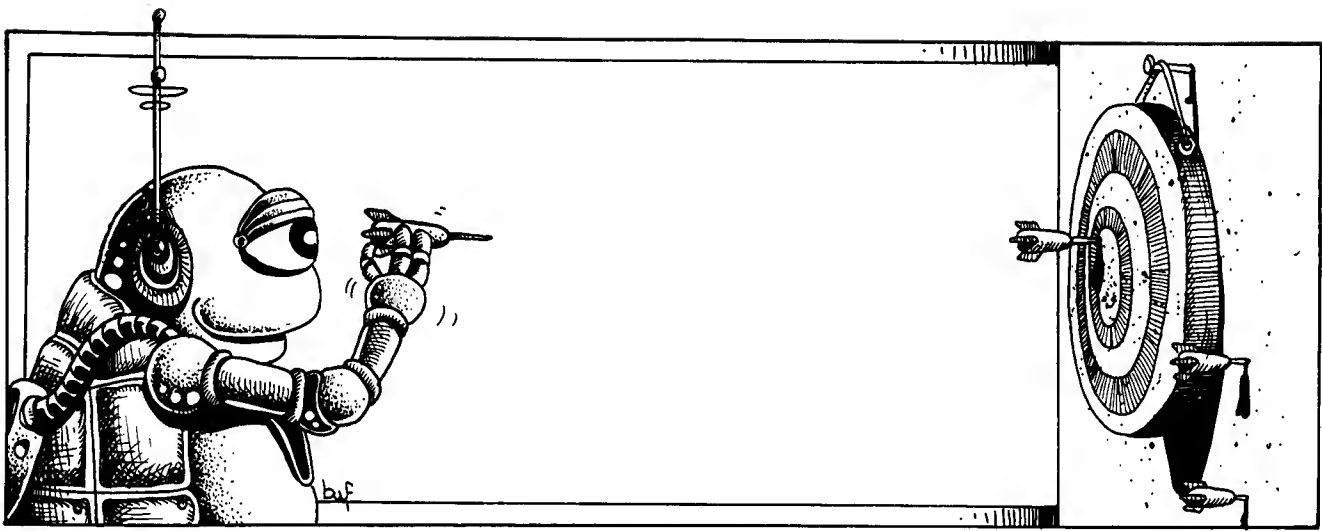
Just as we learned another way to position the turtle, there is another way to turn the turtle besides using **RIGHT** and **LEFT** commands. When we look at a map, we see north, south, east and west. Compasses are marked off in degrees with north being 0, east being 90, south being 180, and west being 270.



The computer has a built-in compass, marked off in degrees. If a turtle faces the top of the screen, he is facing 0. To the right is 90, to the bottom is 180 and to the left is 270. And, of course, he can face any point in between.

If you give the command **SETH** (SET Heading) followed by a number, the turtle will automatically turn and face that direction. Can you figure out what the difference between **RT 90** and **SETH 90** is? What happens if you give the command **SETH 360**? What if you give the command **SETH -90**? Can you give the command **SETH 450**? If you said no, try it. Can you explain what happened?

The target game with stickers can be used again to practice with headings. This time the student will aim at the target using the **SETH** command and will estimate the number of turtle steps needed to reach the target.



How about an automatic target game, one that will draw a target at a random location on the screen and then return the turtle to home? Then the student must use the **SETH** command to aim the turtle and a **FORWARD** command to reach the target. When the student reaches the target, clear the screen and type **TARGET** again. Later you can add sound effects and a test so the computer will congratulate you when you reach the target. For now, let's keep it simple.

```
TO TARGET
CS PU
SETPOS LIST (RANDOM 320) (RANDOM 120) - (RANDOM 70)
PD
SQ
PU HOME
END

TO SQ
REPEAT 4 [FD 10 RT 90]
END
```

**RANDOM** is a new command. If we type **RANDOM 100**, the computer will output a random number between 0 and 100. Look at the line that tells the turtle to set its position. The first number, which sets the X position, is **RANDOM 320**. We are telling the computer to pick any number between 0 and 320. If you experimented much with **SETX** commands, you may have discovered that **SETX 320** will cause the turtle to wrap around the screen and return **HOME**. (This number may differ slightly from screen to screen.) By telling the computer to pick a random number between 0 and 320, we have included all possible X positions, both negative and positive.

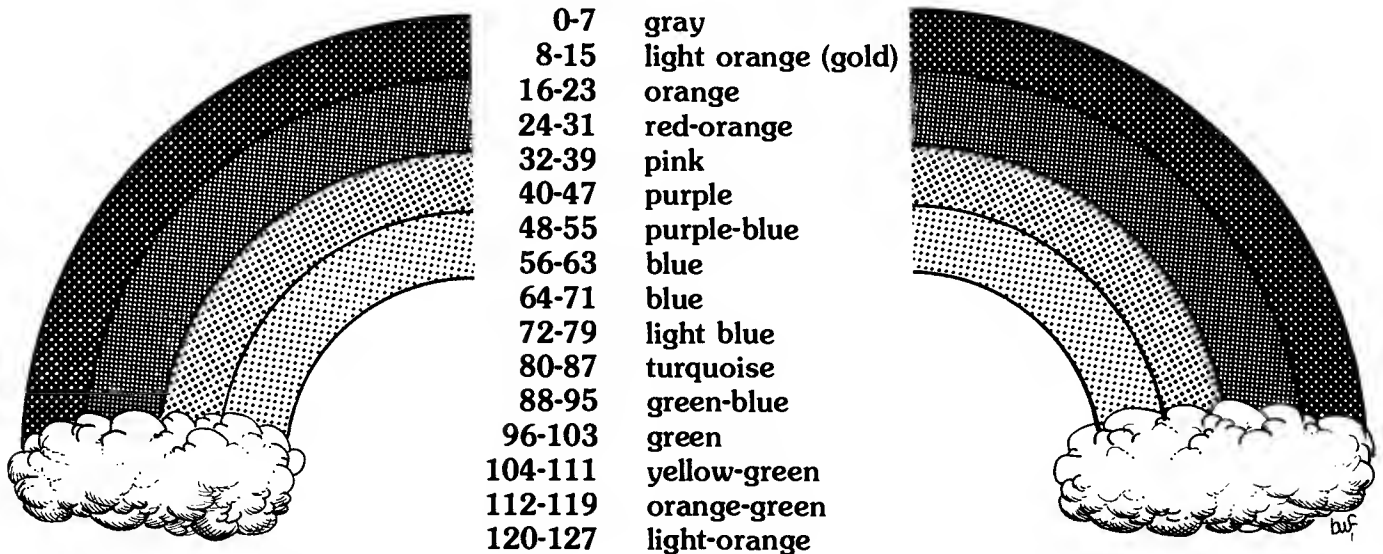
The **Y** command is a little trickier since we don't want the target to be drawn underneath the text lines. Start backing the turtle up and you'll discover he can go about 70 steps before he slips behind the text lines. **SETY -70** is about as far as we want him to go. It's about 120 steps to the top of the screen. So, we'll **SET** the Y position by telling the computer to select a random number between 0 and 120 and then subtract a random number between 0 and 70.

The purpose of the target game is to encourage the students to practice with the **SETH** command and with estimating distances. The procedures of the target game introduce a new command — **RANDOM** — that may open up new avenues of exploration for your students. Challenge them to come up with other ways to use **RANDOM**.

## A Rainbow of Colors

ATARI LOGO has a whole rainbow of colors you can use — 128 of them, in fact. They are divided into sixteen color families, each with eight shades. Just as the turtles are numbered, so are the colors, and we call for them by number. The lowest number in each family is the darkest and the highest number is the lightest. Here is a listing of the color families and their numbers.

### ATARI LOGO Colors



0-7	gray
8-15	light orange (gold)
16-23	orange
24-31	red-orange
32-39	pink
40-47	purple
48-55	purple-blue
56-63	blue
64-71	blue
72-79	light blue
80-87	turquoise
88-95	green-blue
96-103	green
104-111	yellow-green
112-119	orange-green
120-127	light-orange

You can change the color of the turtles, the ink they draw with and the background. You can get pretty colorful if you want to! Let's start with the turtles. After you call up a turtle with TELL 0 or 1 or 2 or 3, use the command SETC (SET Color) followed by the number of the color you select. Maybe you want all four turtles to be blue:

```
TELL [0 1 2 3 ]  
SETC 56
```

Try changing the colors of the turtles one at a time or all at once.

Changing the background (BG) color is very similar. Just type the command:

SETBG and the number of the color.

Changing the pen color is a little more complicated because each of the turtles has three pens he can draw with and each can be a different color. You can draw an entire flower garden with yellow flowers, green stems, and purple butterflies. You don't like yellow flowers? With one command, you can change them all to red even after the picture is drawn on the screen!

Unless you command the turtle to use another pen, he is using pen 0. There are also a pen 1 and a pen 2. To change the pen, use:

SETPN 1 (or 2)

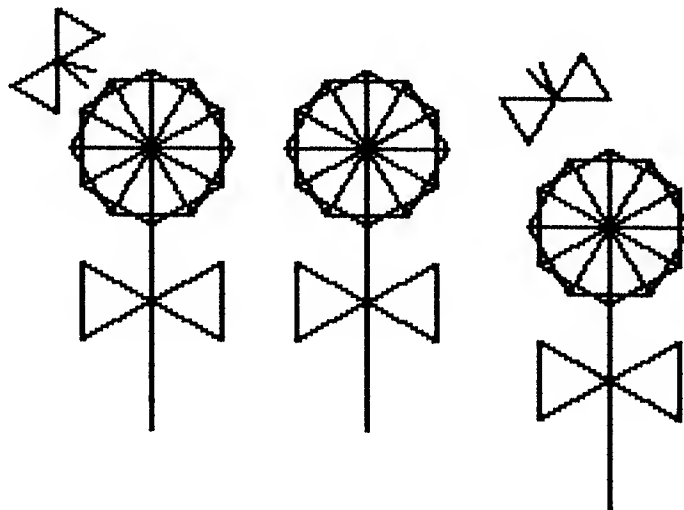
When you turn the computer on, PN 0 draws in color 15 (gold), PN 1 is 47 (purple), and PN 2 is 121 (red). But you can change them to anything you want with the command SETPC (SET Pen Color). The command looks like this:

SETPC 0 59

The first number is the pen number and the second is the number of the color. Once you have changed the pen color, you'll have to tell the turtle to draw with that pen:

SETPN 0

Draw a design using all three pens. Here's an example. For a flower garden, draw the flowers with pen 0, the stems with pen 1, and a butterfly or two with pen 2.



Now try this:

SETPC 0 23  
SETPC 1 87  
SETPC 2 103

Everything you drew with pen 0 will turn orange; everything you draw with pen 1 will turn turquoise; and everything you drew with pen 2 will turn green.

If you ever forget what pen you are using, type

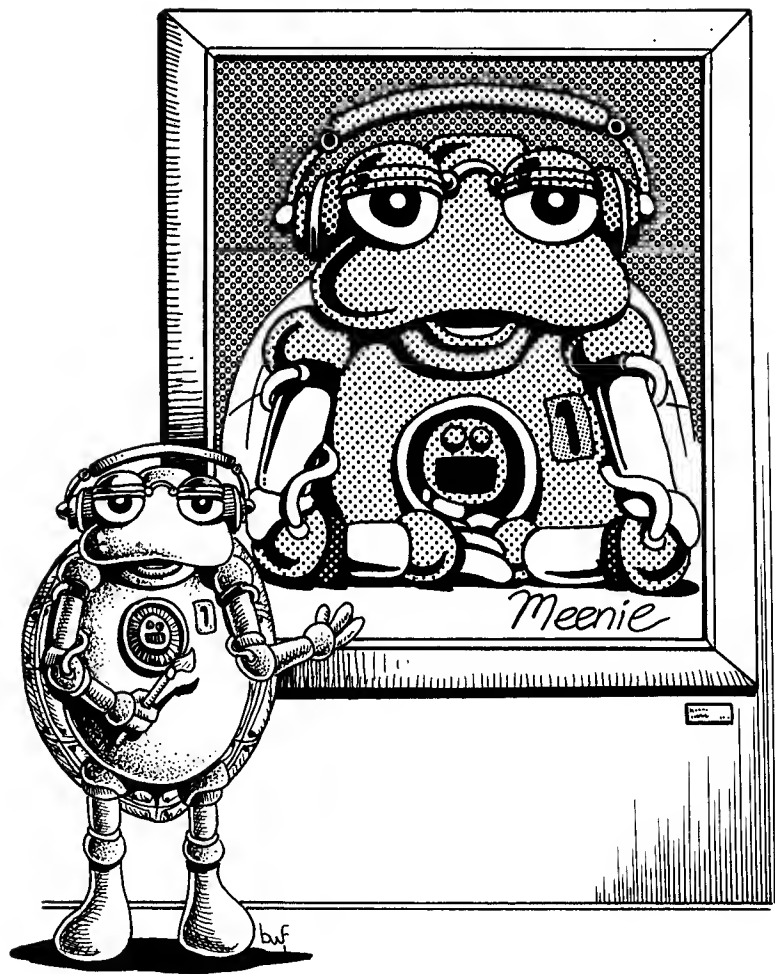
PR PN (which stands for Print Pen)

and the computer will respond by printing the number of the pen. If you want to know what color you are using, type

PR PC (Print Pen Color)

Have your students create designs or pictures using all three of the pens and different color for the pens, turtles, and background.





### Signing Your Work

Any artist who is proud of his work signs it. Any student who wants a grade on a term paper puts her name on it. If you've put hours and hours into creating something, it is important to let people know whose work it is. So let's label our LOGO projects!

PRINT is a command that tells the computer to PRINT something. How logical! Type PRINT and whatever message you want printed enclosed in brackets. For example:

```
PRINT [THIS IS MY GARDEN]
PRINT [DESIGNED BY IVA GREENTHUMB]
PRINT [ON MAY 14, 1984]
```

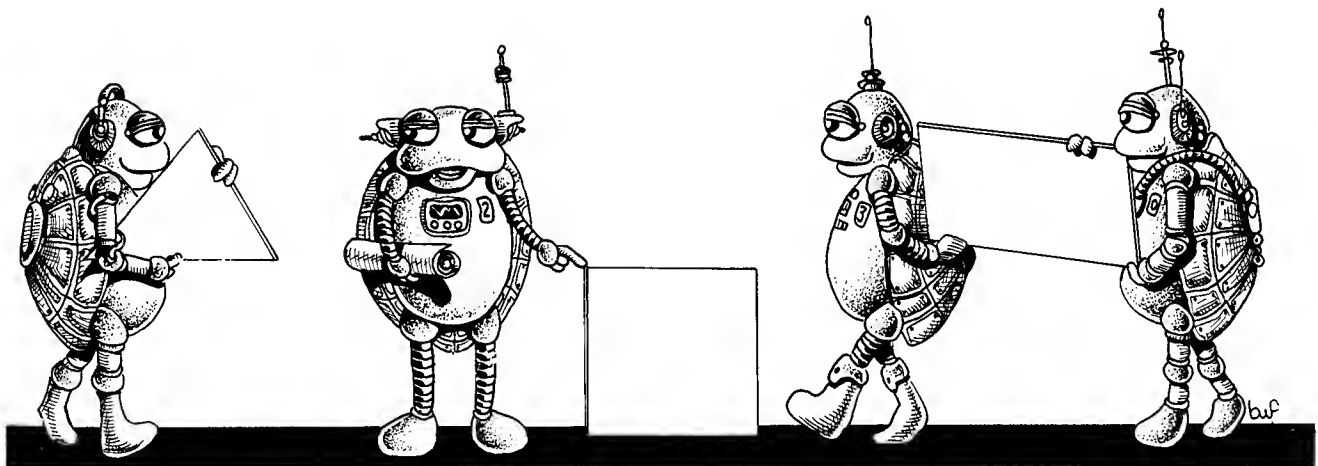
If we add those commands to the beginning of the GARDEN procedure, first the computer will print out the messages and then the turtles will draw the picture. Or, we could add them as the last three commands of GARDEN. The turtles would draw the picture first and then the printed message would appear.

It's such an easy thing to do. If you have the students get in the habit of adding at least their name and the date to whatever designs they create, they will have a chronological record of their work in LOGO. And you will be able to tell immediately whose work it is.

# Variables

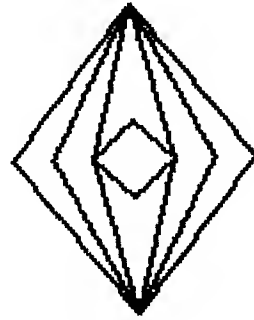
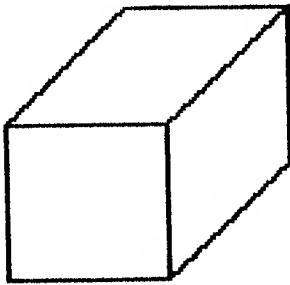
Select one shape, a square, a triangle, or a rectangle. Create a picture or design with it. You may use as many sizes of that shape as you want.

Define a procedure with a variable for the shape you selected. Teach the turtle to draw your design. Write down all the procedures you define.



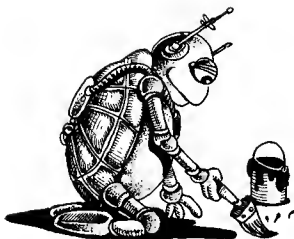
## X and Y Coordinates

Select one of the following simple designs (or make up one of your own) and teach the turtle to draw it using the X and Y coordinates. Keep the pendown and use the command SETPOS [\_\_\_\_\_ \_\_\_\_\_] to move from one position to the next.



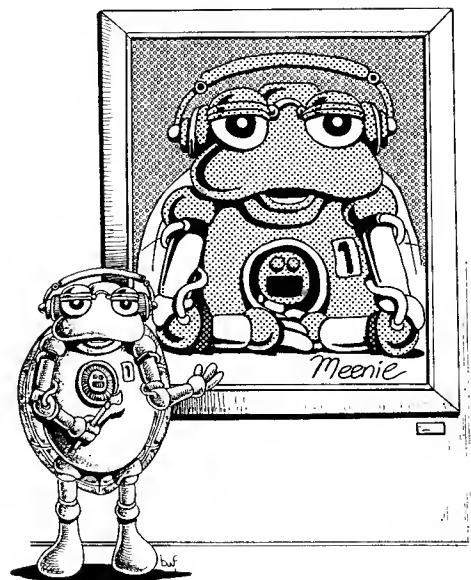
# Designing in Color

Draw a picture using three different pens and three different colors. Then change the colors and choose your favorite color combination. The commands you will need are SETPN 0, SETPN 1, SETPN 2, SETPC 0 \_\_\_\_, SETPC 1 \_\_\_\_, and SETPC 2\_\_\_\_.



# Doodles and Designs

A page for keeping notes of doodles and designs and how they were made.



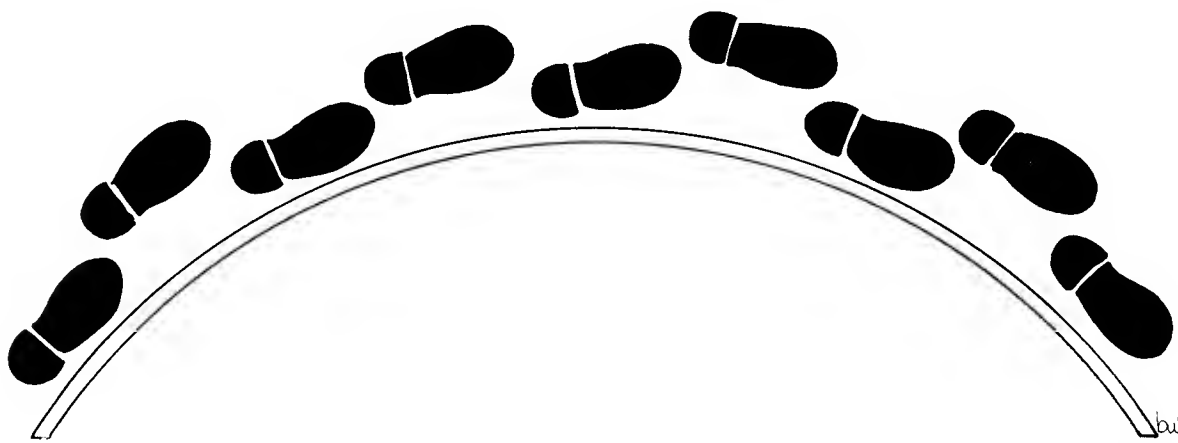


# 4 The Total Turtle Trip



When people of any age begin working with turtle graphics, it doesn't take them long to want to know how to make a circle. Making a square seems fairly obvious, but how do you make curved lines? The best way to figure this puzzle out is to play turtle.

Have the students take turns walking around a large circle, describing what they are doing.



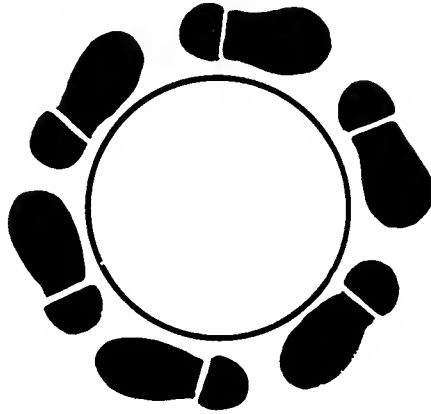
If they can see that to walk in a circle, they will have to take a step, turn a little, take a step, turn a little, and so on, they can begin to see what it will take to tell the turtle to walk in a circle. It will be a succession of FD and RT or LT commands and, since it's a small step and a small turn, perhaps FD 1 RT 1 would be a good place to start. But how many times should that be repeated?

When they begin to seek the answer, they may type in FD 1 RT 1 several times. They will soon realize that this method is too tedious and might take the rest of the day. How about using the REPEAT command? Since it is up to them to discover the correct number of repeats, tell them to start with any number and keep adding up the repeats they use until they get all the way around the circle. If they go too far, clear the screen and try again. If they think they've got it, hide the turtle with HT and see if the circle is completed. (To show the turtle, use ST.)

Someone will soon discover that the magic number is 360. And when they discover it for themselves, it will be much more meaningful than if someone had told them to memorize the fact that there are 360 degrees in a circle. They have experienced those 360 degrees, and as we move through the rest of this chapter, 360 will become more and more important.

Allow them time to enjoy this new bit of information. Someone is bound to ask, "But how do I make a smaller circle?" If not, then you ask! If they're totally stumped, it's time to play turtle again. Have the students walk in a smaller circle. The steps can remain the

same size, but what happens to the turn? The turtle has to make a bigger turn between each FD command.



Suppose we change RT 1 to RT 2? Let them try it. What will happen if we give the command:

```
REPEAT 360 [FD 1 RT 2]
```

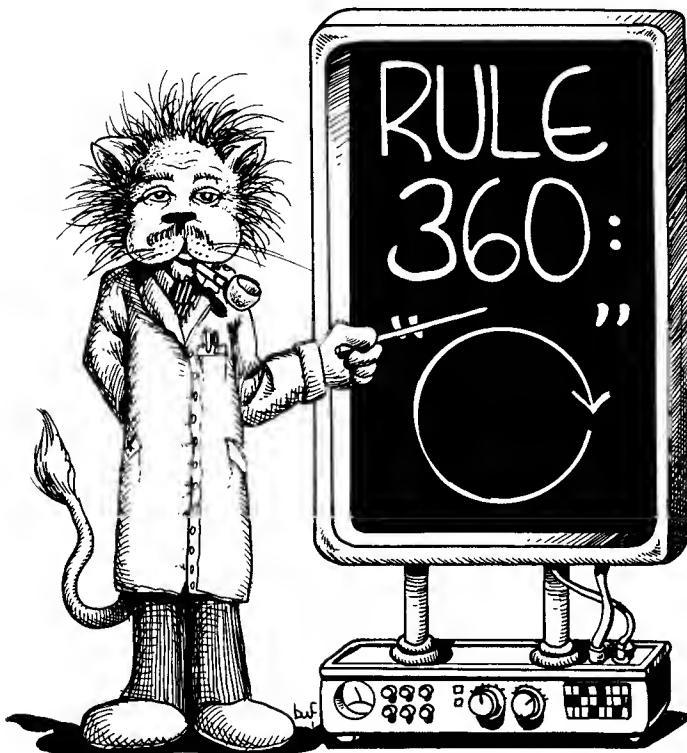
It's a smaller circle, but the Turtle goes around twice! If we change the turn, we'll have to change another number in the procedure. Through experimentation they should discover:

```
REPEAT 180 [FD 1 RT 2]
```

And that brings up another possibility. What if we change the REPEAT 180 to REPEAT 90? What else would we have to change in the command? When they discover REPEAT 90 [FD 1 RT 4], ask if those numbers look familiar. Aha! It's the square procedure backwards! To make a square, we repeat 4 times and turn RT 90. Could we reverse the numbers in the triangle command to make a circle? Try it.

```
TO TRIANGLE  
REPEAT 3 [FD 20 RT 120]  
END
```

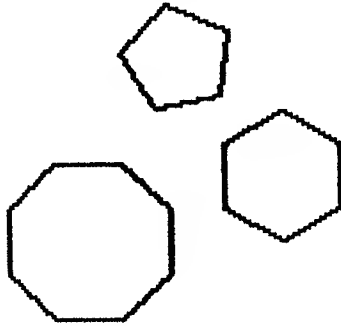
```
TO CIRCLE  
REPEAT 120 [FD 1 RT 3]  
END
```



There seems to be a very important relationship between the REPEAT number and the turn number. Multiplied together, they always equal 360. Any time the turtle rotates through a complete circle, he will rotate through 360 degrees. Any time the turtle draws a polygon and ends up facing the same direction as he started, he will have rotated through 360 degrees. That's called the Total Turtle Trip or The Rule of 360.



Using that piece of information, challenge the students to draw a pentagon, a hexagon and an octagon. Use these new figures to make new designs or add to some already started.



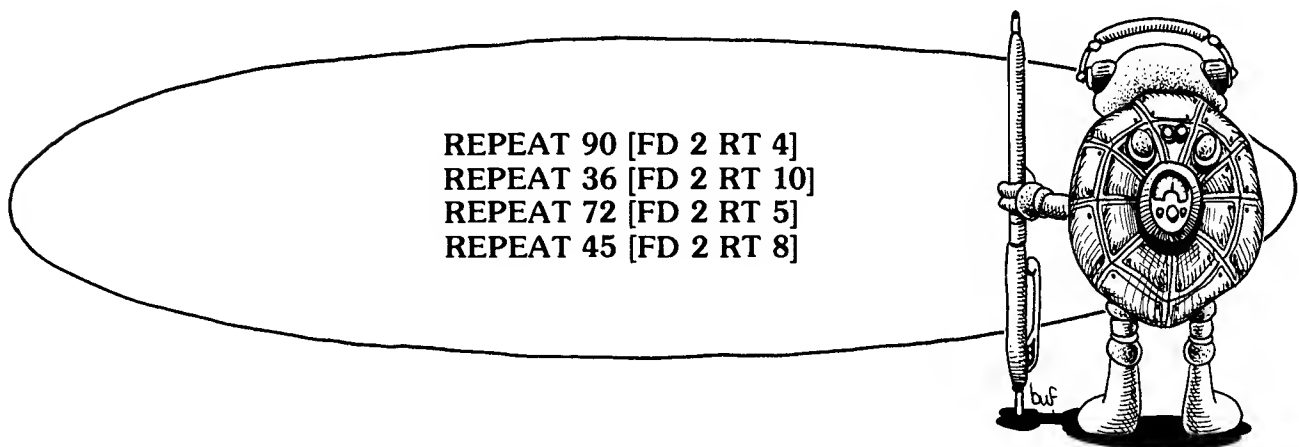
TO HEXAGON  
REPEAT 6 [FD 20 RT 60]  
END

TO PENTAGON  
REPEAT 5 [FD 30 RT 72]  
END

TO OCTAGON  
REPEAT 8 [FD 10 RT 45]  
END

The Rule of 360 is an extremely important mathematical concept. Students should be given time to explore with it and play with it and try numerous things before you go on.

How many different circle procedures can you define?



(There is a way to make circles using the radius to designate the size. For older students, you may want to introduce this method. See page 93 of **Introduction to Programming Through Turtle Graphics.**)

Compare what happens with the different circle procedures. Which ones make large circles and which make small circles?

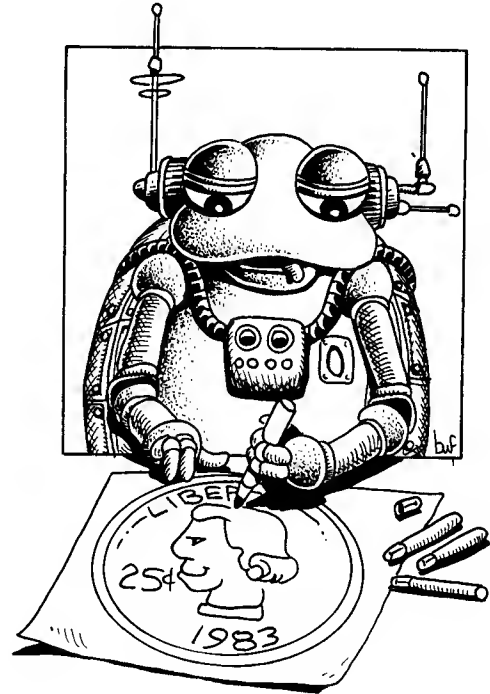
Pick out one circle procedure and define it with a variable so you can draw any size circle you want.

```
TO CIRCLE :N  
  REPEAT 36 [FD :N RT 10]  
END
```

Do the same thing with a pentagon, a hexagon, and an octagon.

Suppose you want to draw only half of a circle. If CIRCLE is defined as REPEAT 36 [FD 5 RT 10], how would we define HALF.CIRCLE?

How would you draw a quarter circle?



Create designs or pictures using various sizes of circles and pieces of circles.



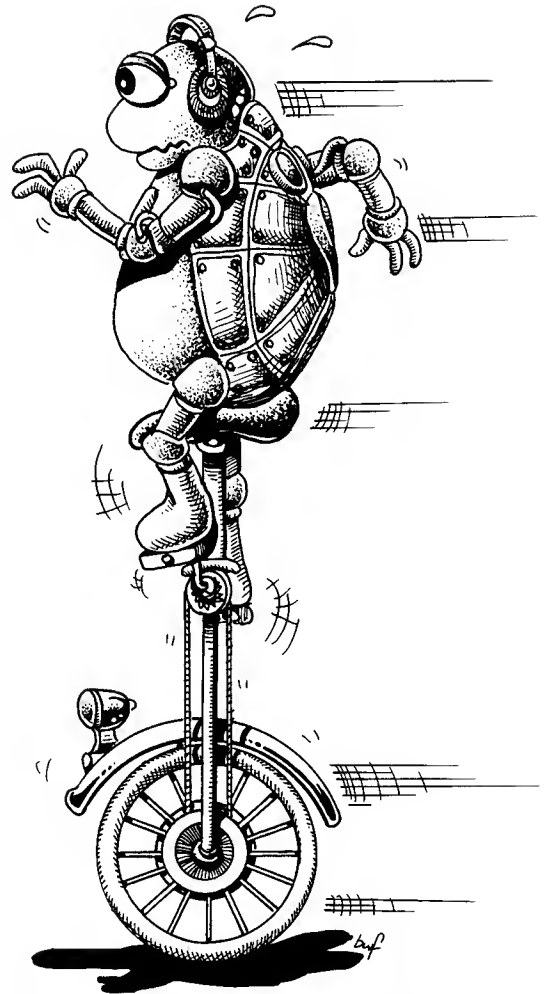
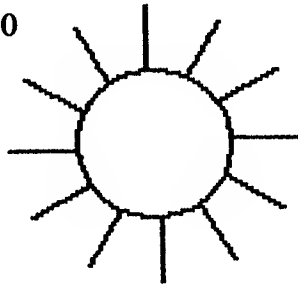
## Spokes on a Circle

Suppose you want to draw a bicycle tire or the sun or some other design in which you have lines coming out from a circle. Think about the Total Turtle Trip and what you have learned so far about drawing parts of a circle. If you want to have twelve spokes coming out from the circle, you could draw one-twelfth of the circle, draw a spoke, draw another twelfth, draw a spoke and so on. Look at the following procedures:

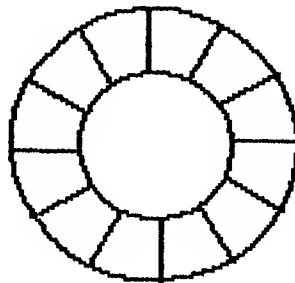
```
TO SUNSHINE  
REPEAT 12 [ARC SPOKE]  
END
```

```
TO ARC  
REPEAT 3 [FD 5 RT 10]  
END
```

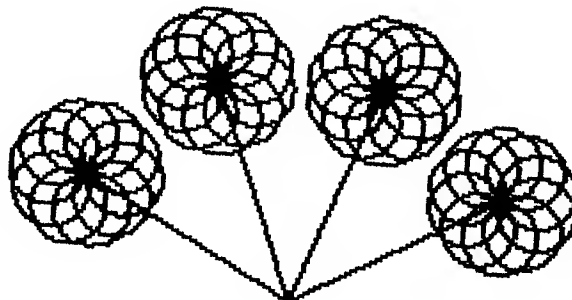
```
TO SPOKE  
LT 90 FD 25 BK 25 RT 90  
END
```



A real challenge is to draw the SUNSHINE, and then draw another circle around the spokes.



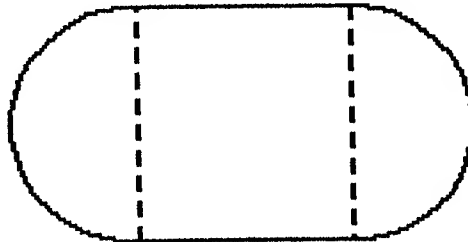
Try some designs with circles using all four turtles starting in different positions. Maybe you can draw a whole bouquet of flowers at once.



## Ovals

What about an oval? There's probably a very difficult way to draw an oval. There's also a very simple way.

Think about ovals and think about circles. How would you transform a circle into an oval? Think about the track around a football field. If you could cut off the two rounded parts around the ends of the field and put them together, you would have a circle.

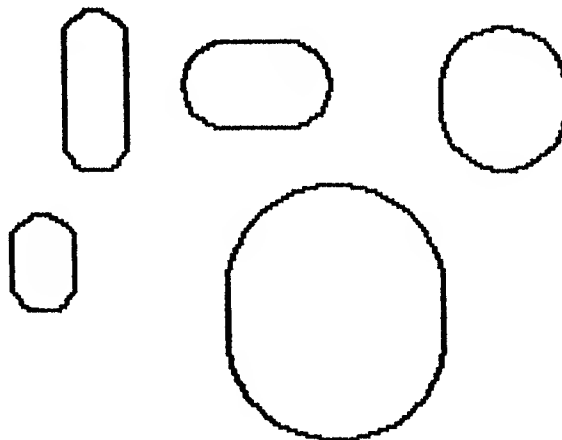


How would you get a turtle to draw the track around the football field? The easiest way would be to draw a half circle, go forward a certain amount, draw another half circle and go forward a certain amount again. Now can you define an oval?

```
TO OVAL  
REPEAT 18 [FD 10 RT 10]  
FD 40  
REPEAT 18 [FD 10 RT 10]  
FD 40  
END
```

Suppose you want to be able to draw different sizes of ovals. You might want to make the length of the side shorter or longer. Or you might want to make the half circles smaller or larger. Looks like we need a procedure with two variables.

```
TO OVAL :A :B  
REPEAT 18 [FD :A RT 10]  
FD :B  
REPEAT 18 [FD :A RT 10]  
FD :B  
END
```

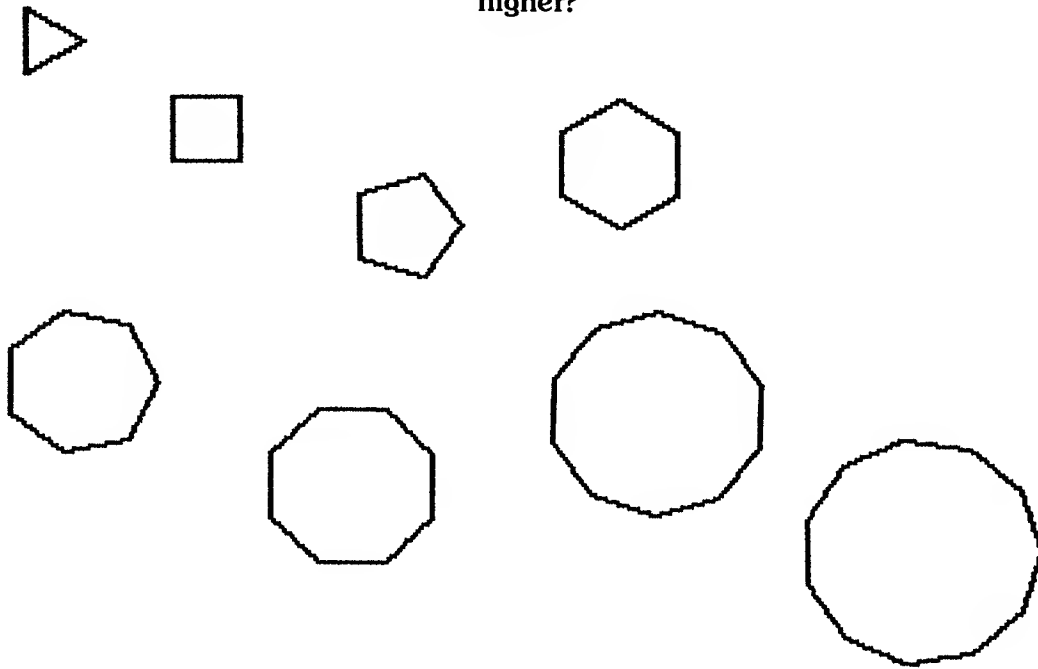


## A General Purpose Polygon Procedure

Just as we learned earlier that the computer can multiply using the \* symbol, it can divide using the / symbol. If we combine some variables with some division and what we have learned about the Total Turtle Trip, we can define a procedure to draw any regular polygon. Since the REPEAT number multiplied by the turning number equals 360, we could tell the turtle to turn RT 360 divided by the REPEAT number. Look at the following procedure:

```
TO POLYGON :R
REPEAT :R [FD 20 RT 360 / :R]
END
```

When we give the command POLYGON, we'll follow it with a number input to indicate how many sides the polygon should have. Try the procedure, increasing the number input by 1 each time: POLYGON 3, POLYGON 4, POLYGON 5, etc. What happens as the number gets higher and higher?



## Interactive Procedures

Earlier, the students used the PRINT command to sign their work. (PRINT can be abbreviated PR). Now let's take a look at:

SE (sentence)  
RL (readlist)  
IF

These commands will allow us to write *interactive* procedures, ones that require some input from the user while the procedure is running.

SE combines words or lists and outputs them as a "sentence." For example, if you type:

```
PRINT SE [I LOVE] [ATARI LOGO]
```

the computer will combine those two lists of words into one sentence.

RL, as we are going to use it, waits for someone to type an input and will then output whatever is typed. For example, here is a simple procedure to ask someone's name and then greet that person by name:

```
TO GREET
PRINT [WHAT IS YOUR NAME]
PRINT SE [HELLO] RL
END
```

Have your students define simple procedures using PRINT, SE, and RL. They can use any questions, but must be able to combine the response with a message from the computer. Here are a few examples:

```
Who is your favorite singer?
  I don't like_____
Where would you like to go on vacation this year?
  I'd love to go to_____
What sport do you enjoy playing?
  They won't let me play_____
```

The GREET procedure could be expanded to include some of the questions.

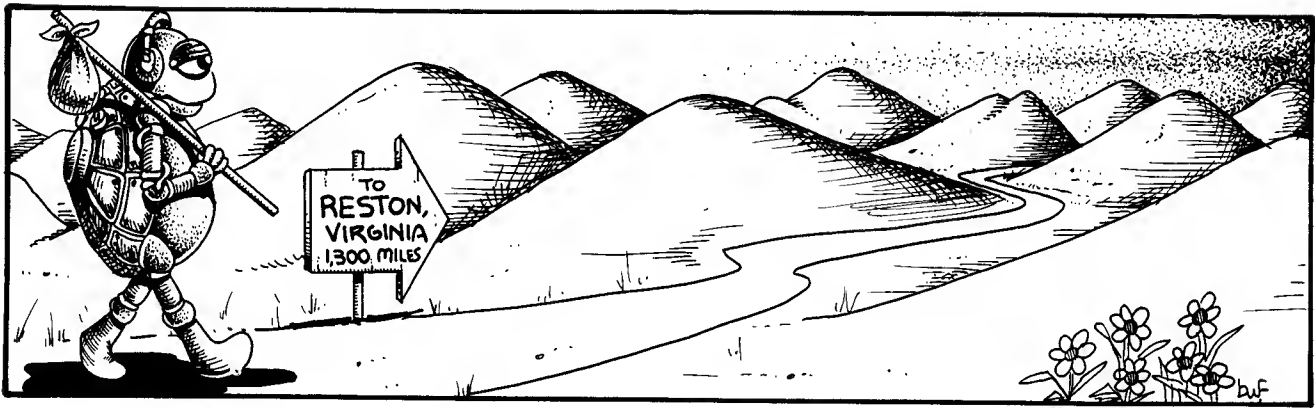
```
TO GREET
PRINT [WHAT IS YOUR NAME?]
PRINT SE [HELLO] RL
PRINT [WHO IS YOUR FAVORITE SINGER?]
PRINT SE RL [IS REALLY GREAT.]
PRINT [WHERE WOULD YOU LIKE TO GO ON VACATION?]
PRINT SE [I'D LOVE TO GO TO] RL
PRINT [WHAT'S YOUR FAVORITE SPORT?]
PRINT SE [I CAN'T PLAY] RT
END
```

Suppose we want to include some yes/no questions? We'll use a *conditional* command that tells the computer IF certain conditions are met THEN do a certain thing. Let's add on IF to our GREET procedure:

```
TO GREET
PRINT [WHAT IS YOUR NAME]
PRINT SE [HELLO,] RL
PRINT [DO YOU LIKE COMPUTERS?]
IF RL = [YES] [PRINT [GREAT!]] [PRINT [TOO BAD, THEY'RE HERE TO STAY]]
END
```

If the user types YES in answer to the question, the computer will respond with GREAT! IF the user types NO, then the answer does not equal YES. The computer will skip over the first possible output and will print the second list: TOO BAD, THEY'RE HERE TO STAY.

Have your students add more yes/no questions to the procedure.

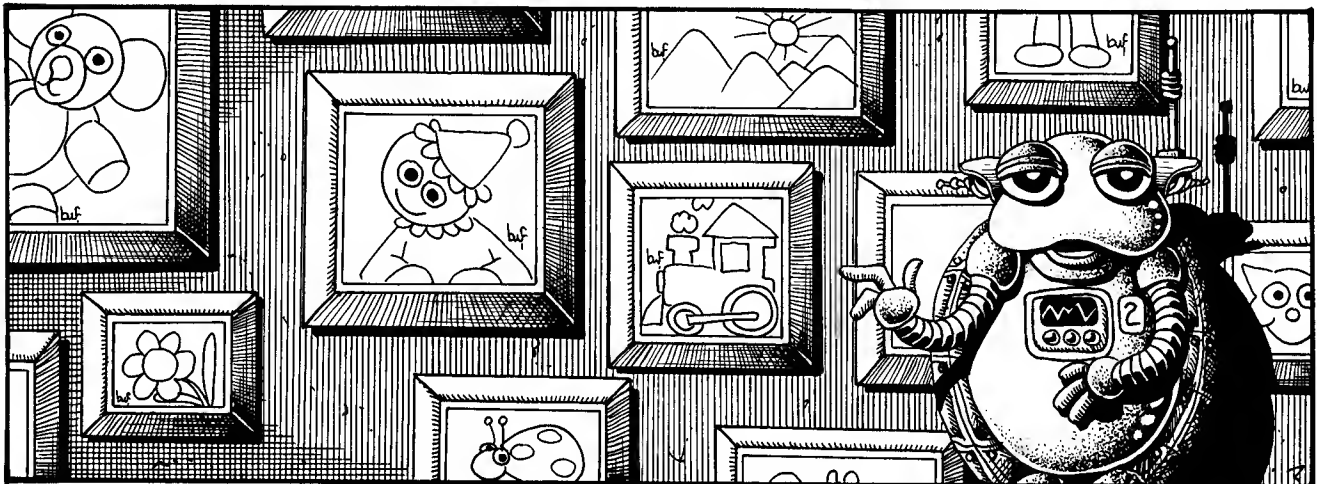


Let's combine some choices with what we've learned about the Total Turtle Trip. We'll start by asking if the user knows what the Total Turtle Trip is. If he doesn't know, we'll try to explain it to him and then test to see whether or not he can use it to describe some polygons. Notice that the third line is very long. Be sure to type the entire line before pressing RETURN.

#### TO TRIP

```
PRINT [DO YOU KNOW WHAT THE TOTAL TURTLE TRIP IS?]
IF RL = [YES] [PRINT [GOOD! LET'S MAKE SOME POLYGONS.] [PRINT [THE TUR-
TLE TURNS THROUGH 360 DEGREES TO MAKE A POLYGON. DIVIDE 360 BY THE
NUMBER OF SIDES AND YOU'LL KNOW HOW MUCH TO TURN THE TURTLE TO
DRAW EACH SIDE.]
PRINT [TRY A SQUARE.]
PRINT [HOW MANY SIDES DOES A SQUARE HAVE?]
IF RL = [4] [PRINT [RIGHT]] [PRINT [NO, IT HAS 4.]]
PRINT [HOW MUCH SHOULD THE TURTLE TURN FOR EACH CORNER?]
IF RL = [90] [PRINT [RIGHT!]] [PRINT [NO, IT'S 90.]]
END
```

Have your students add other commands to describe other polygons.



#### Computer Art Show

By now, your students have probably created many designs. Have them select some of their favorite designs to put into a computer art show. We'll define a procedure that will allow them to show off several of their designs and also give someone a choice about which designs they want to see.

```
TO ART.SHOW
```

```
CS
```

```
PRINT [WOULD YOU LIKE TO SEE A DESIGN?]
```

```
IF RL = [YES] [PRINT [DO YOU WANT TO SEE SQUARES, TRIANGLES, OR  
CIRCLES?]] [PRINT [AW SHUCKS, GOODBYE]] STOP]
```

```
IF RL = [SQUARES] [SQUARES WAIT 120 ART.SHOW]
```

```
IF RL = [TRIANGLES] [TRIANGLES WAIT 120 ART.SHOW]
```

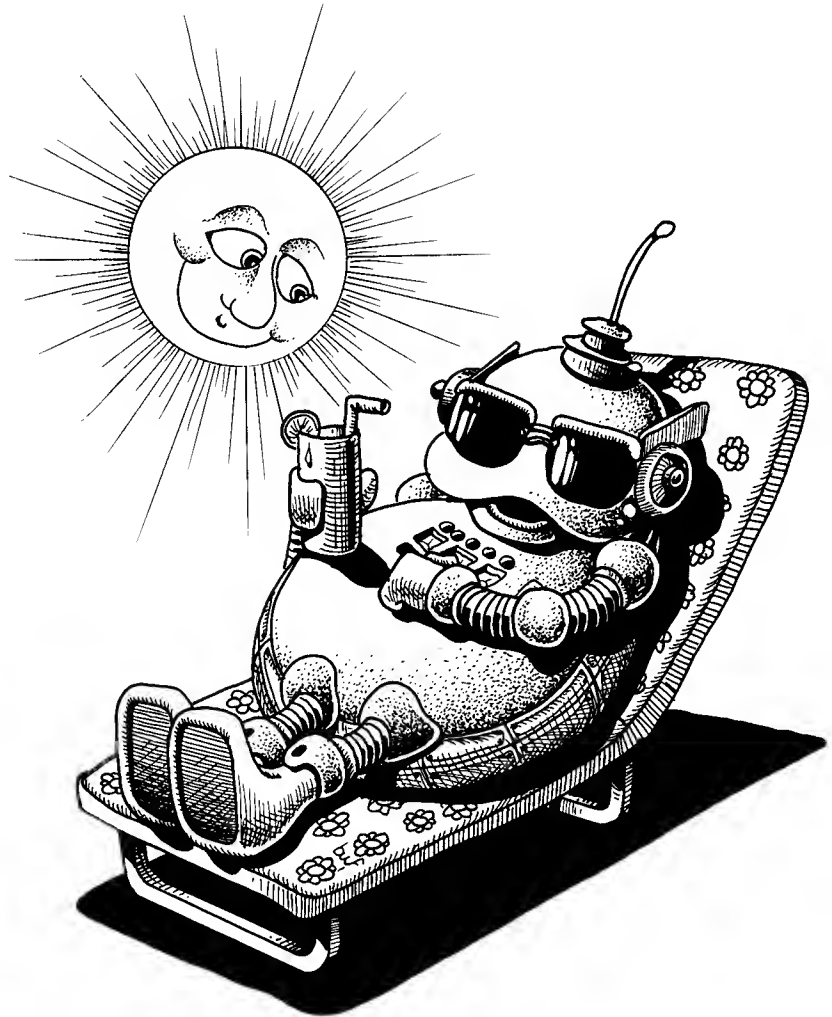
```
IF RL = [CIRCLES] [CIRCLES WAIT 120 ART.SHOW]
```

```
END
```

The first choice is a yes/no question. If the user answers YES, the computer will type DO YOU WANT TO SEE SQUARES, TRIANGLES OR CIRCLES? and the user then has another choice to make. If the user answers NO, the computer will respond with AW SHUCKS, GOODBYE and the procedure will STOP.

The second choice offers the user three selections: SQUARES, which will call up a design named SQUARES; TRIANGLES, which will call up a design named TRIANGLES; and CIRCLES, which will call up a design named CIRCLES. When the user makes his choice and types it in, the design will be carried out. Then the procedure will WAIT 120 (which equals two seconds) and start over again since the next command calls the procedure named ART.SHOW. This is our first use of *recursion*, which we will learn more about in the next chapter.

Have your students define a procedure similar to ART.SHOW that will incorporate some of their own designs.

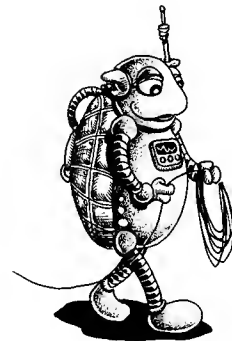




# Circles

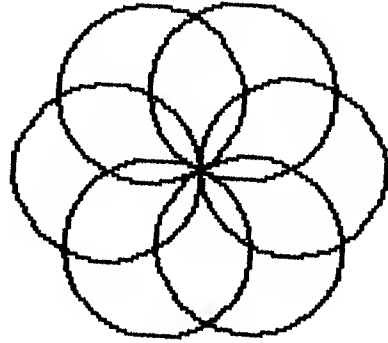
Define 10 ways to make a circle.

Select one procedure and define it with a variable to make different sizes of circles.



## Circle Designs

Select one of these designs or make up your own and teach the Turtle to draw it.



# Polygons

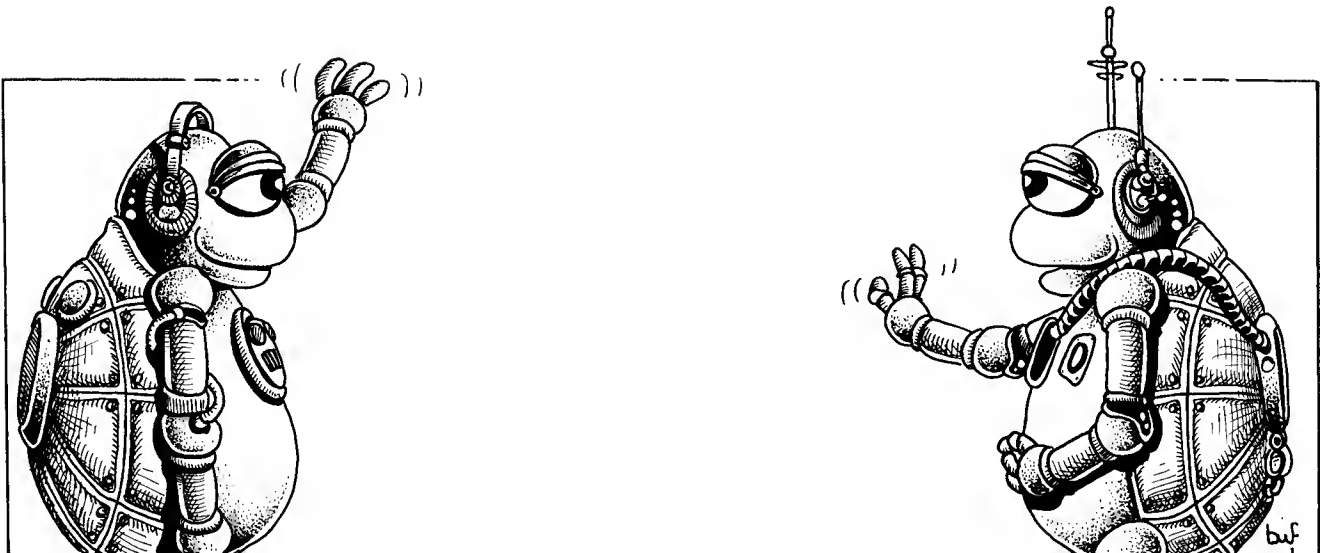
Use the Rule of 360 to help you fill in the following chart.

Name of Polygon	Number of Sides	Turtle Turn

# Greetings

Using PRINT, SE, RL and IF, add at least three more questions and responses to the GREET procedure.

```
TO GREET
PRINT [WHAT IS YOUR NAME?]
PRINT SE [HELLO,] RL
PRINT [DO YOU LIKE COMPUTERS?]
IF RL = [YES] [PRINT [GREAT!]] [PRINT [TOO BAD, THEY'RE HERE TO
STAY]]
END
```



# Total Turtle Trip

Add commands to the TRIP procedure to cover a triangle, a hexagon and an octagon.

TO TRIP

PRINT [DO YOU KNOW WHAT THE TOTAL TURTLE TRIP IS?]

IF RL = [YES] [PRINT [GOOD! LET'S MAKE SOME POLYGONS]] [PRINT [THE  
TURTLE TURNS THROUGH 360 DEGREES TO MAKE A POLYGON. DIVIDE  
360 BY THE NUMBER OF SIDES AND YOU'LL KNOW HOW MUCH TO TURN  
THE TURTLE TO DRAW EACH SIDE.]]

PRINT [TRY A SQUARE.]

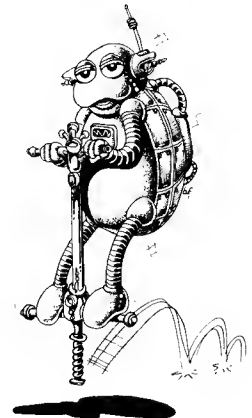
PRINT [HOW MANY SIDES DOES A SQUARE HAVE?]

IF RL = [4] [PRINT [RIGHT]] [PRINT [NO, IT HAS FOUR.]]

PRINT [HOW MUCH SHOULD THE TURTLE TURN FOR EACH CORNER?]

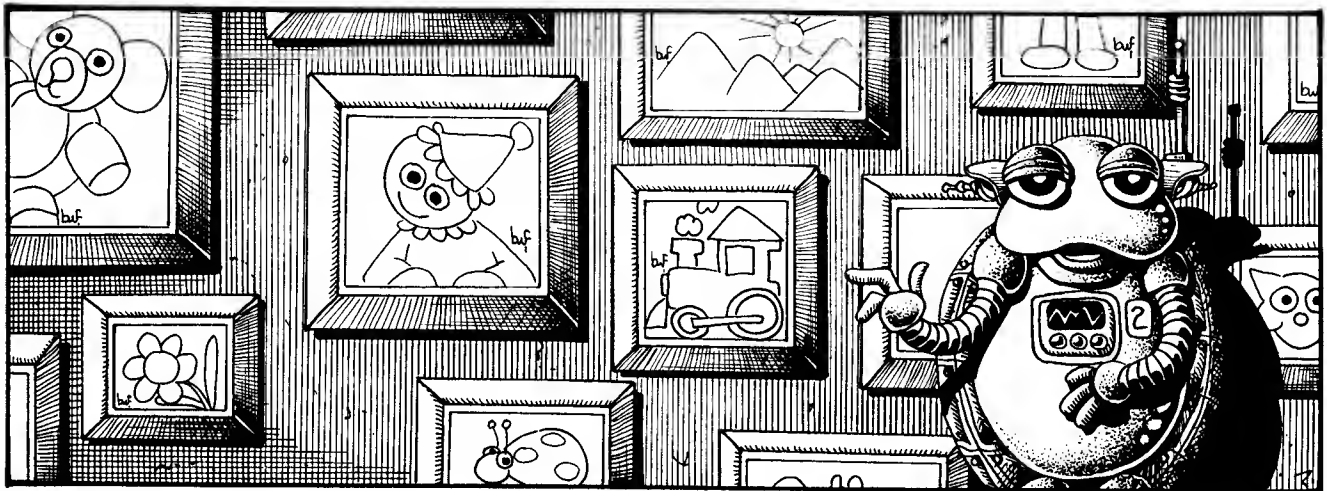
IF RL = [90] [PRINT [RIGHT!]] [PRINT [NO, IT'S 90.]]

END



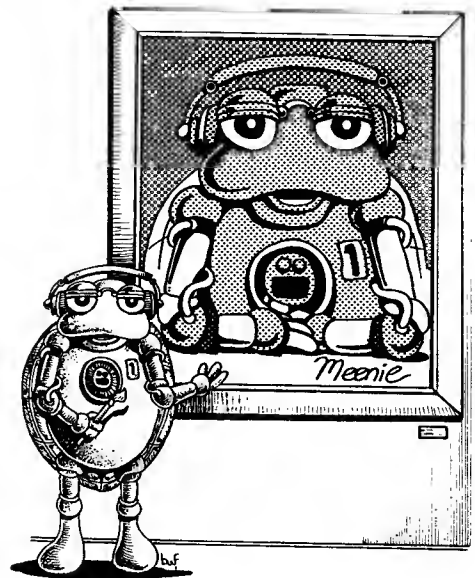
# Computer Art Show

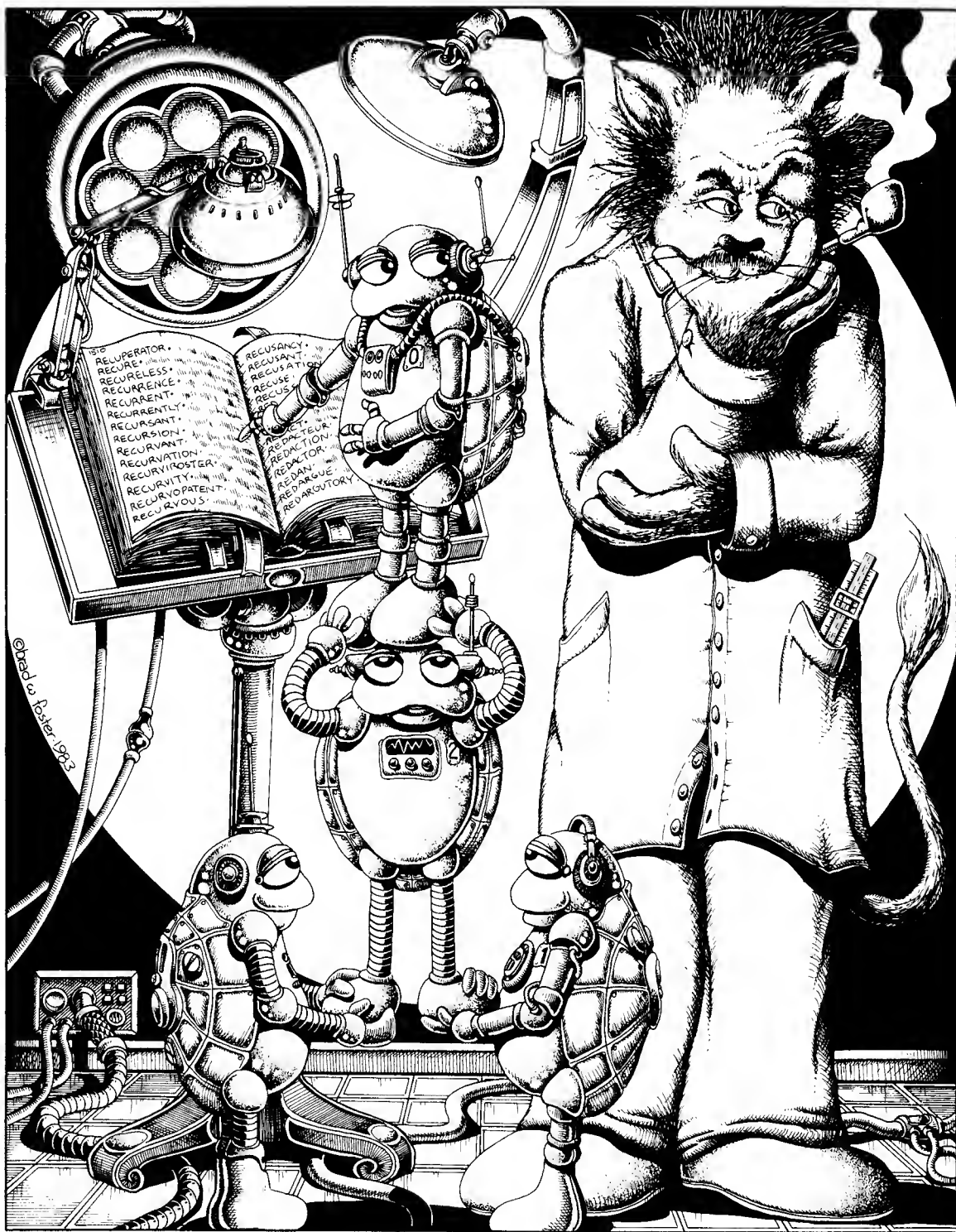
Select three or four of your favorite designs and put them in a LOGO Art Show. Write a procedure that lets someone choose which picture he or she wants to see.



# Doodles and Designs

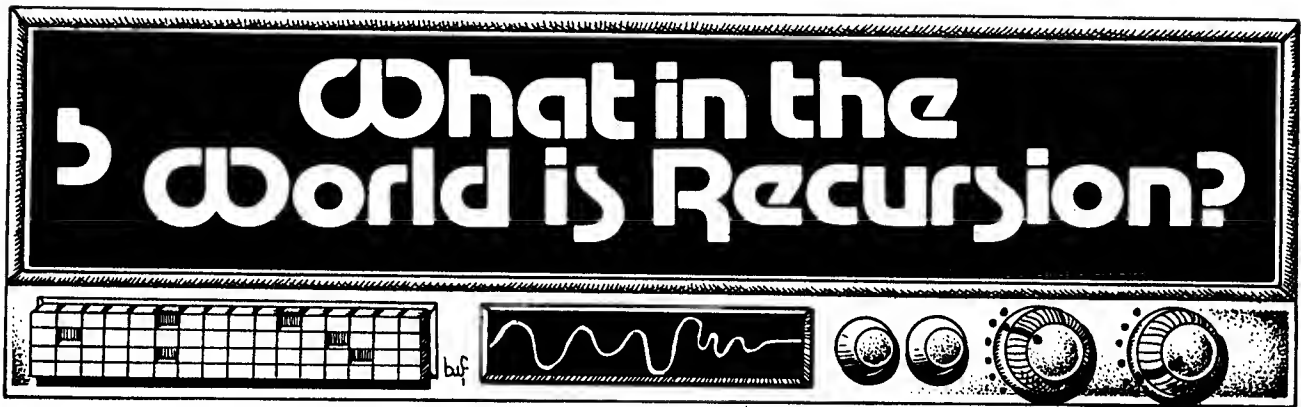
A page for keeping notes of doodles and designs and how they were made.







# What in the World is Recursion?



You have just finished grading a set of papers when you receive another set to grade. You finish grading that set and you get another set. Many of the steps you go through to grade those papers are the same each time. In fact, many of our day-to-day tasks can be broken down into sequences that are repetitive.

Suppose we could write a simple LOGO procedure to grade papers. We want to write one procedure to take care of all the papers for the entire year. The procedure might look like this:

```
TO GRADE.PAPERS
  PICK UP PAPERS
  GET RED PENCIL
  GET CUP OF COFFEE
  DRINK COFFEE
  CHECK ANSWERS
  IF INCORRECT, MARK WITH RED PENCIL
  TOTAL NUMBER OF CORRECT ANSWERS
  GIVE PAPERS BACK TO STUDENTS
  GRADE.PAPERS
END
```



Each command in this procedure calls another procedure that has a series of commands to be carried out. For example, GET CUP OF COFFEE might include WASH CUP, POUR COFFEE, and ADD TEASPOON OF SUGAR. Look at the last command in the GRADE.PAPERS procedure. The last command is GRADE.PAPERS. It calls the GRADE.PAPERS procedure and you start the process all over again.

*Recursion* is the ability of a procedure to call itself. In these first few examples, we'll look at *tail-end recursion* — the recursive call is the last command in the procedure.

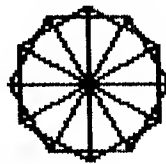
We must have a way to stop recursive procedures or they'll go on forever. At first we'll use the BREAK key. Simply "break" into a procedure to stop the turtle. Later we'll add a conditional command to stop the procedure. For example, in the GRADE.PAPERS procedure we could add:

```
IF JUNE 1 [STOP]
```

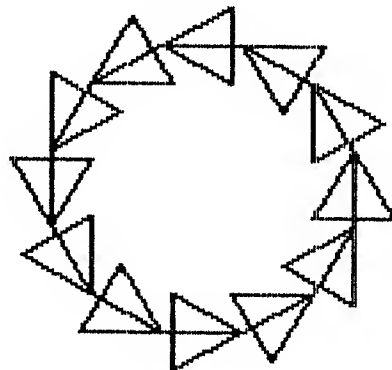
In the ART.SHOW procedure at the end of the last chapter, the procedure will continue until the user responds with NO to the question, "Do you want to see a design?"

Let's try some very simple recursive procedures with the turtles. What would happen if we drew a shape, rotated the turtle slightly and called the procedure again:

```
TO TRIANGLES  
REPEAT 3 [FD 35 RT 120]  
RT 30  
TRIANGLES  
END
```



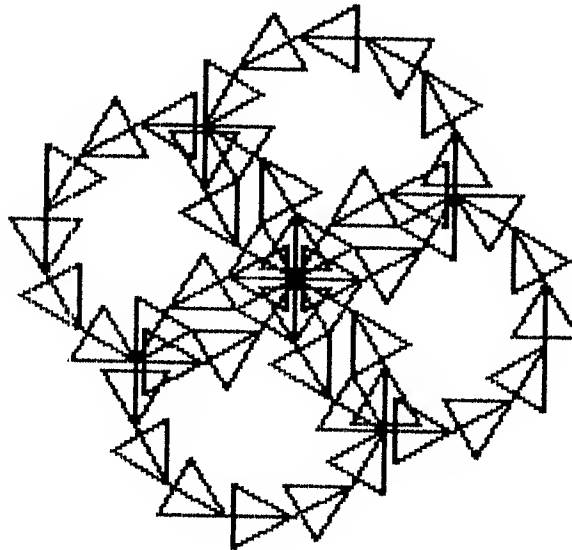
Try it and see what happens. What happens if you change the 30 to 29? What happens if you add FD 30 right after the RT 30?



Here's one using all four turtles and the START procedure we defined earlier. Try it once with the turtles visible. Then hide them (HT) and try it again.

```
TO SURPRISE
START
TRIANGLES
END
```

```
TO TRIANGLES
REPEAT 3 [FD 35 RT 120]
RT 30 FD 35
TRIANGLES
END
```



Have the students develop several simple recursive procedures of their own. Try some with one turtle and some with more than one turtle. Allow plenty of time for them to experiment with this new concept.

Once the students have explored, experimented, and played with simple recursive procedures, you can add another element. Let's try recursion with a procedure that includes a variable. We could do that quite simply in the procedure above by including the variable that lets us draw any size triangle we want. Try it. Notice that you must include the variable in the recursive line.

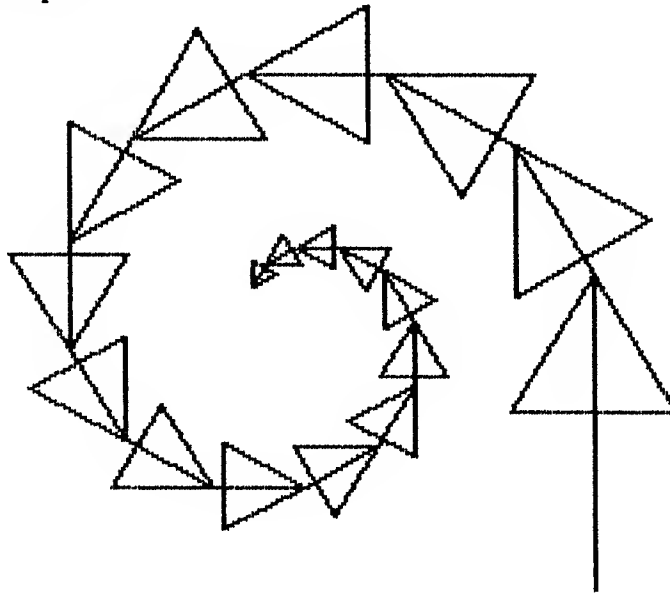
```
TO TRIANGLES :X
REPEAT 3 [ FD :X RT 120]
RT 30 FD :X
TRIANGLES :X
END
```

Adding a variable gives us some flexibility but it doesn't change the procedure significantly. We'll still end up with a ring of triangles if we start with one Turtle and four rings if we use START and all four turtles.

Let's add one small addition and watch an amazing thing happen. In the recursive line, add + 3 like this:

```
TO TRIANGLES :X
REPEAT 3 [FD :X RT 120]
RT 30 FD :X
TRIANGLES :X + 3
END
```

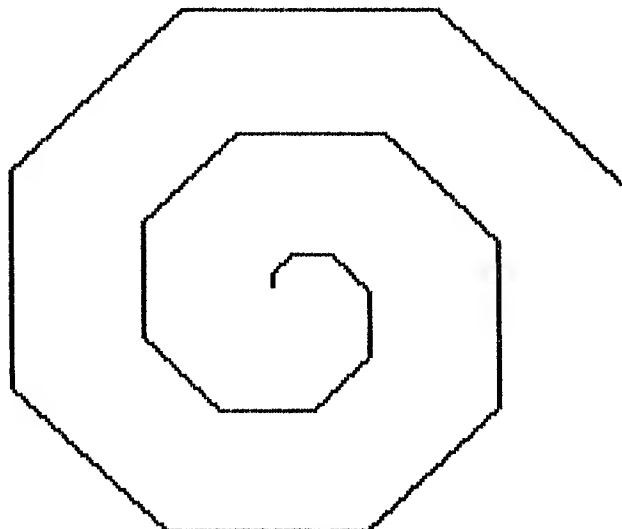
Can you predict what is going to happen? Try it with one turtle. Every time the turtle gets down to the recursive line, the computer will add 3 to the value of X. For example, if you give the command TRIANGLES 10, the Turtle will draw a triangle with 10 steps to a side, turn RT 30, go FD 10, and will read the recursive line. It tells him to add 3 to the value of X, which is presently 10. The next triangle he draws will be 13 steps to a side, then 16, then 19, and so forth. The other thing that happens is that instead of a ring of triangles, you'll see a spiral!



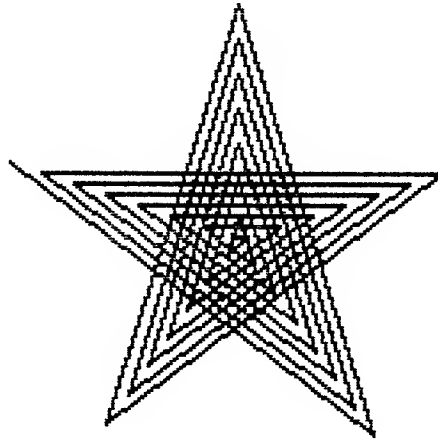
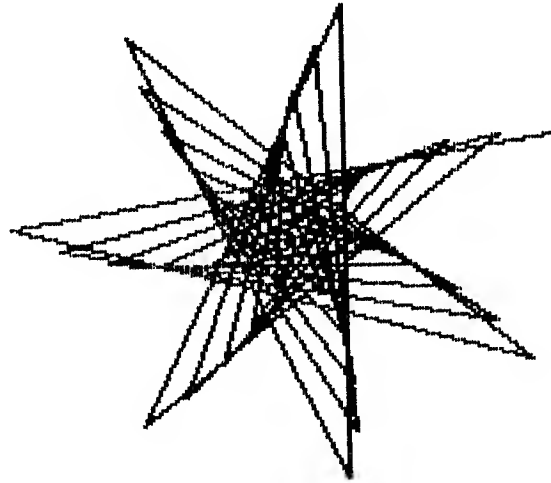
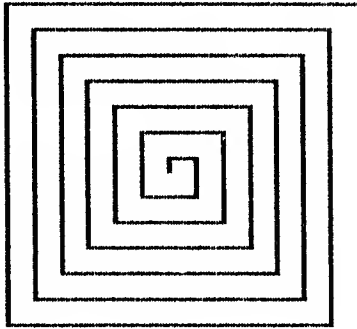
To really watch the procedure grow, press CTRL F for fullscreen. When you stop the procedure with BREAK, it will automatically go back to split screen. You also can go back to the split screen with CTRL S.

What happens if you get all four turtles drawing spirals? Well, if you're using the TRIANGLES procedure, it may look interesting, but it begins to lose the spiral shape pretty quickly. Let's simplify it a bit and make some more pleasing spirals. Try them first with one turtle.

```
TO SPIRAL :X
FD :X
RT 45
SPIRAL :X + 5
END
```



Try turning the Turtle different amounts and adding different amounts in the recursive line. Can you draw a spiral with 90 degree turns? How about a star spiral? How about some other turns?



Once again, something sometimes happens in LOGO that sends you down a whole new path of discovery. Suppose you were drawing a star spiral and wanted to see what would happen if you had all four turtles drawing it at the same time. It's no longer a spiral, but it could be a blossoming flower, or you could be in the cockpit of a space ship getting closer and closer to the sun!

```
TO SUN
START
STAR.SPIRAL 10
END
```

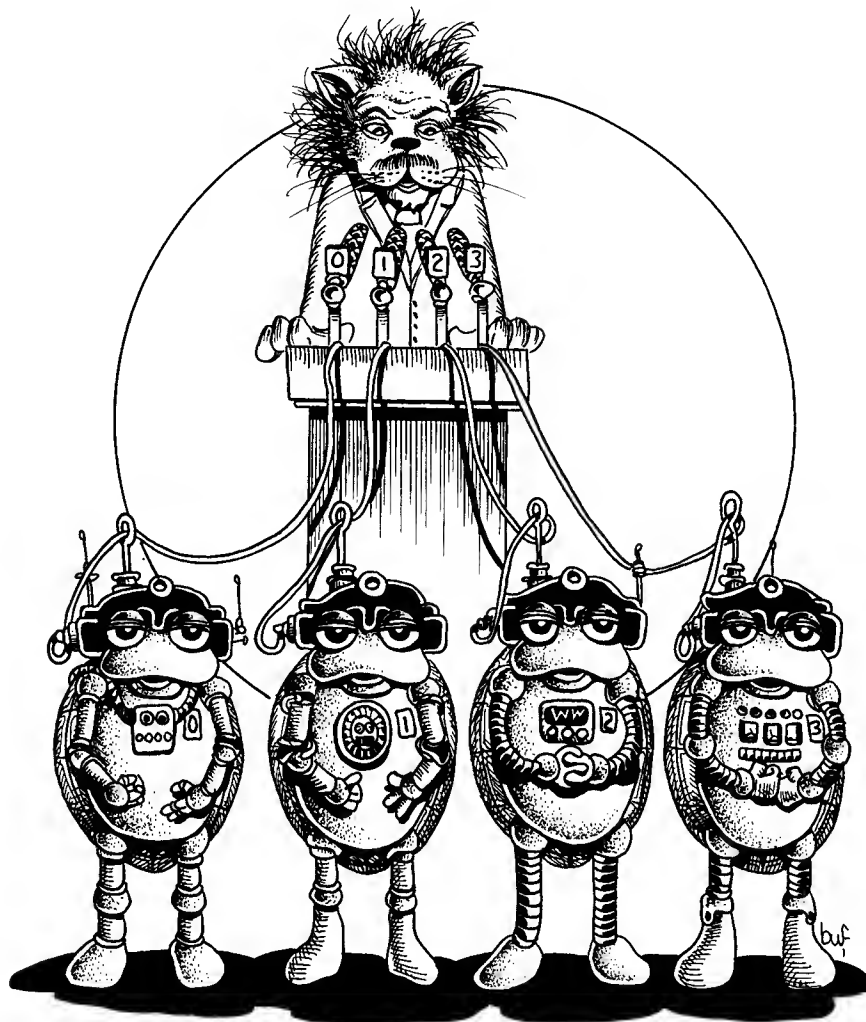
```
TO STAR.SPIRAL :X
FD :X
RT 144
STAR.SPIRAL :X + 3
END
```

Again, allow ample time for your students to explore with each new piece of information before you add another.

Many times you'll want to have the turtle carry out a recursive procedure up to a certain point and then stop or to do something else. By adding a conditional to the recursive procedure, we have more control over what's happening. We could tell the turtle to draw a spiral until it gets to be a certain size. For example:

```
TO SPIRAL :X  
IF :X > 90 [STOP]  
FD :X RT 45  
SPIRAL :X + 5  
END
```

The conditional line reads "If the value of X is greater than 90, stop." The turtle will check the value of X each time the SPIRAL procedure is called. As soon as the value becomes greater than 90, the procedure will stop. Try adding conditional statements to the recursive procedures you have developed.



### Using EACH and TELL with SPIRALS

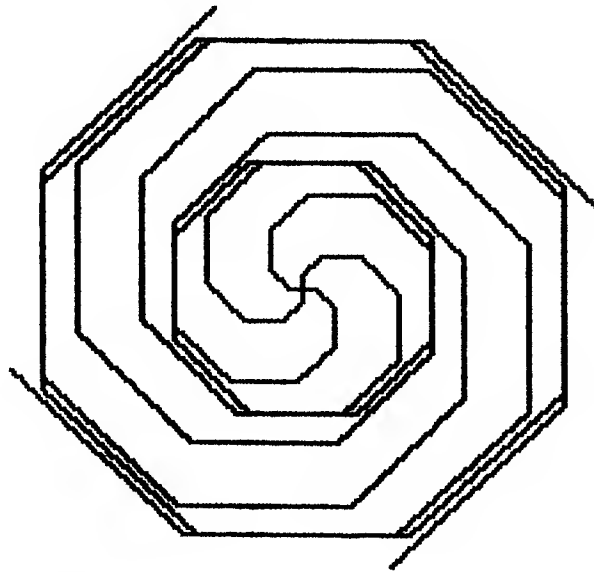
So far we have "talked" to the turtles using TELL. We can also use EACH and a very interesting thing happens, a sort of follow-the leader effect. TELL causes all four turtles to carry out commands at the same time. EACH tells them to carry them out one at a time.

In other words, Turtle 0 will carry out all the commands contained within the brackets. Then Turtle 1 will do the same, then Turtle 2 and then Turtle 3.

The spiral procedure is a fun one to experiment with. Clear your screen and give the command:

EACH [SPIRAL 5]

How about starting the turtles in different directions with our START procedure and then giving the command?

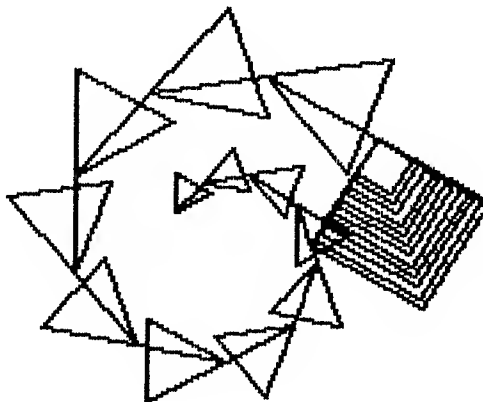


Embedded Recursion

All of the recursive procedures we've used so far are examples of tail-end recursion. Each procedure called itself just before the END statement. Let's add another command or two right after the recursive line and see what happens. Modifying the TRIANGLES procedure, we'll add a conditional and another line to have the turtle draw a square after it completes a spiral of triangles:

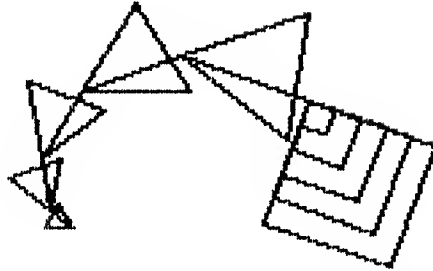
```
TO TRIANGLES :X
IF :X > 50 [STOP]
REPEAT 3 [FD :X RT 120]
RT 40 FD :X
TRIANGLES :X + 3
REPEAT 4 [FD :X RT 90]
END
```

Unless you are already thoroughly familiar with recursion, the results of this procedure probably surprised you. Giving the command TRIANGLES 15 caused the turtle to draw 12 triangles of increasing sizes and then several squares of decreasing sizes, 12 to be exact. Why?



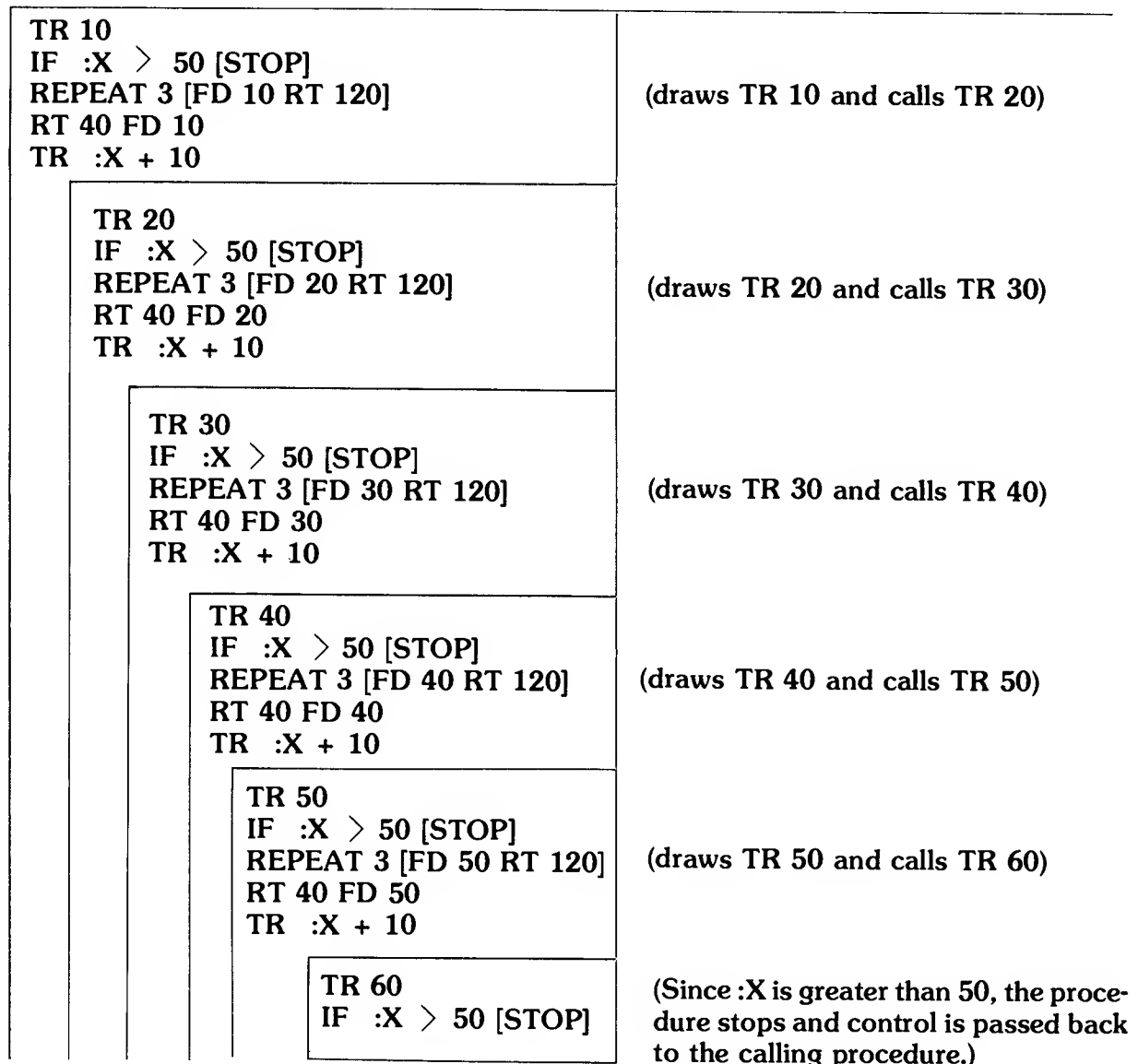
For the sake of illustration, let's edit TRIANGLES slightly:

```
TO TR :X
IF :X > 50 [STOP]
REPEAT 3 [FD :X RT 120]
RT 40 FD :X
TR :X + 10
REPEAT 4 [FD :X RT 90]
END
```

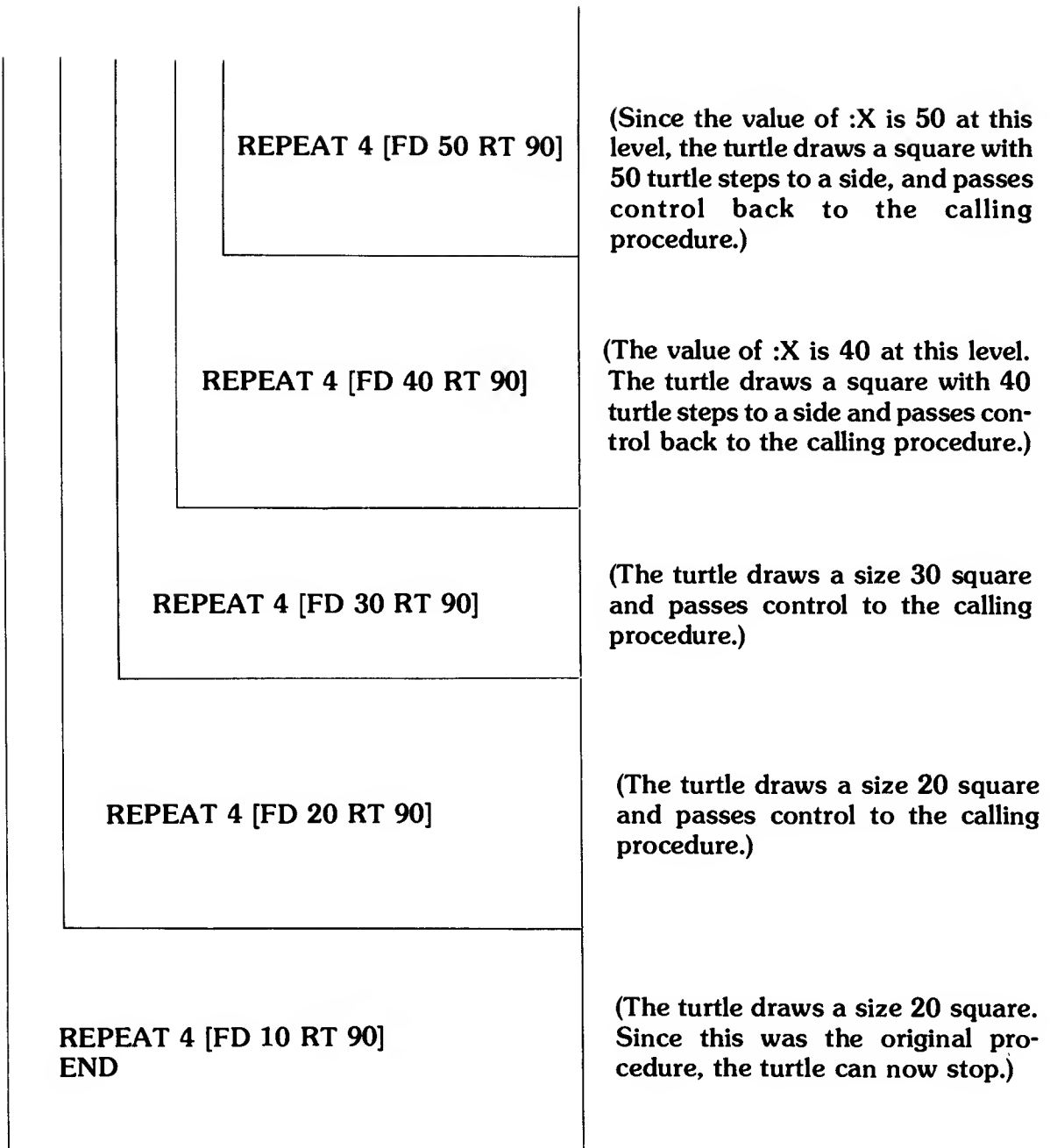


TR 10 causes the turtle to draw five triangles of increasing sizes and then five squares of decreasing sizes. (Rotate the turtle LT 90 before you give the command TR 10 to keep the illustration from wrapping on the screen.)

Let's take it step by step. When a LOGO procedure calls another procedure, the control is passed to the second procedure. When the second procedure is completed, the control is passed back to the first procedure (the calling procedure.) This is true whether you build a simple house with HOUSE calling SQUARE and TRIANGLE or designing an intricate pattern with a recursive procedure.







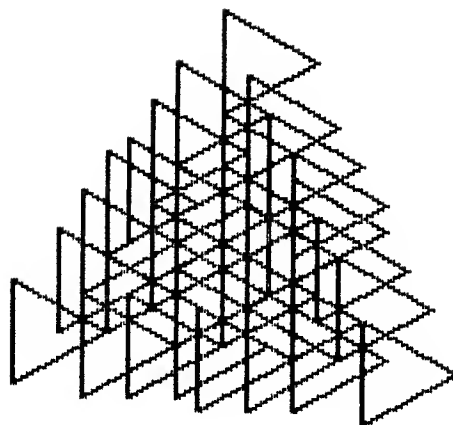
Have your students experiment with placing additional commands after the recursive line. Ask them to predict what will happen before they actually give the turtle the commands. Diagramming the procedures, as in the previous example, may be helpful in tracing the control and following the turtle's movements.

Let's look at another example of embedded recursion. This is another variation of the original TRIANGLES procedure.

```
TO TRI :X
IF :X > 40 [STOP]
REPEAT 3 [FD :X TRI :X + 10 RT 120]
END
```

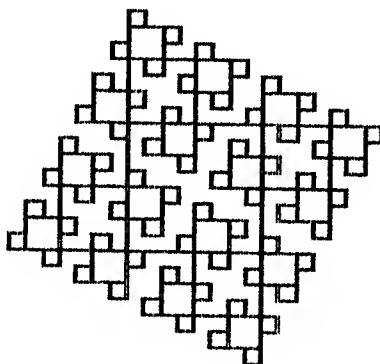
In this case, the recursive call is placed within the brackets of the REPEAT command. Can you predict what will happen? Try it! Then turn the page.

Were you right?



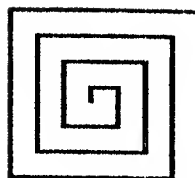
The only way to really get a handle on recursion is to experiment, experiment, experiment. Have your students explore with other polygons. Try using some of the other arithmetic operations in the recursive line ( $:X * 2$ ). Change the IF command to IF  $:X < 5$  [STOP]. Then start with large numbers and subtract or divide in the recursive line ( $:X - 5$  or  $:X / 2$ ).

```
TO SQUARES :X
IF :X < 5 [STOP]
REPEAT 4 [FD :X SQUARES :X / 2 RT 90]
END
```

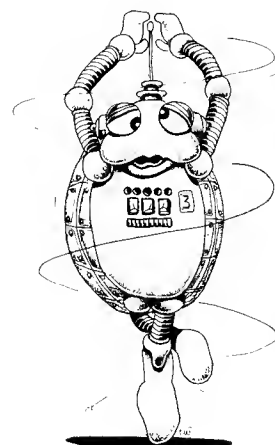
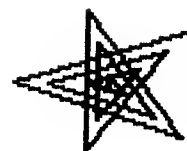


# Spirals

Define procedures to draw the following spirals:



Design a spiral of your own and teach the turtle to draw it.



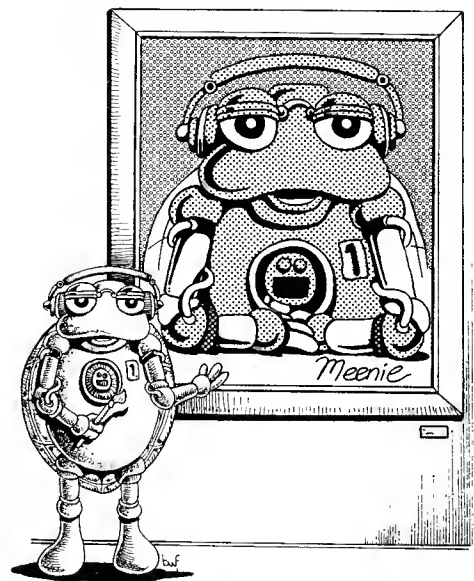
# Recursion

Write a simple recursive procedure and diagram it to trace the control.



# Doodles and Designs

A page for keeping notes of doodles and designs and how they were made.





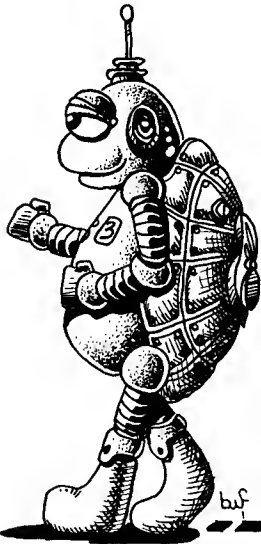
# 6 Collision Detection



ATARI LOGO has a built in collision detection capability that provides an opportunity to create some very exciting, animated programs. When we think of collisions, we think of things bumping into each other. To bump into something requires motion.

So far we have moved the turtles with FORWARD and BACK commands or by using the X and Y coordinates. We can also move the turtles by setting them in motion, giving them a speed. They will continue to move until we either clear the screen or give them a speed of 0. Use the command:

SETSP followed by a number from -199 to 199



What do you think will happen if you give a turtle a negative number? Try it. Try several speeds and watch what happens to the line the turtle draws. At what speed does the line become broken? How would you draw a line of dots across the screen?

If you give a turtle a speed and then tell him to draw something, he'll draw the design and then continue moving at the given speed. Define a square and then try:

SETSP 25 SQUARE

Add a WAIT command and put it in a recursive procedure and you can make the turtle draw and climb a ladder:

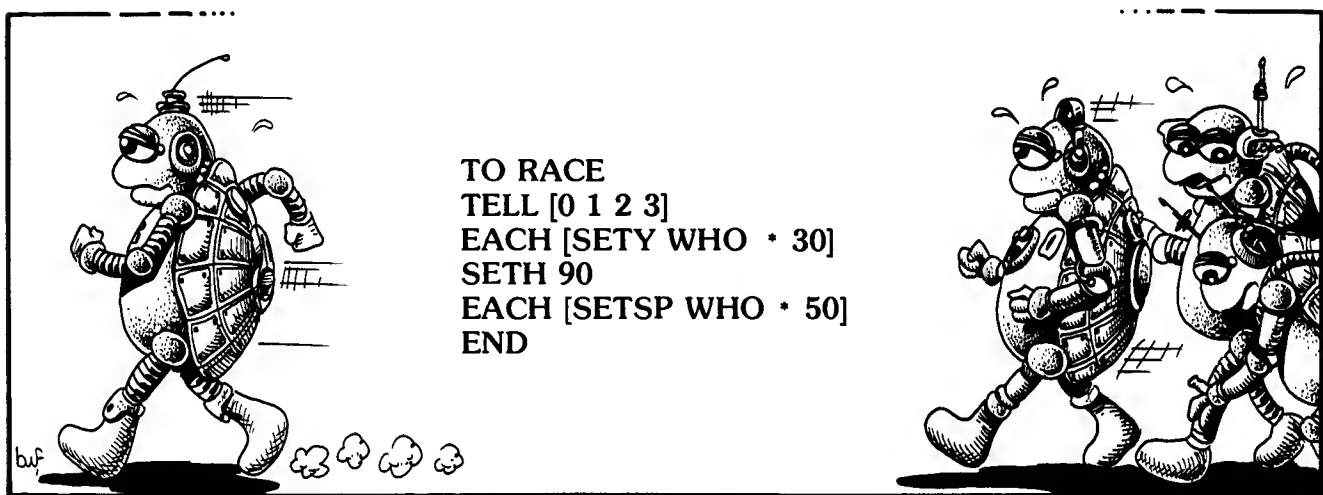
```
TO LADDER
SETSP 25
SQUARE
WAIT 50
LADDER
END
```

```
TO SQUARE
REPEAT 4 [FD 20 RT 90]
END
```

During the WAIT command, the turtle continues to move at the given speed. WAIT doesn't cause the turtle to stop, it only causes a pause before the next command in the procedure.

Let's bring up all the turtles and have a turtle race. WHO outputs the number of the turtle. We can use WHO \* a number to give each turtle a different position and a different speed. For example if we're talking to Turtle 0 and we say WHO \* 30, the computer will multiply 0 \* 30. If we're talking to Turtle 1, the computer will multiply 1 \* 30. We could say EACH [SETY WHO \* 30] and each turtle would go to the Y coordinate of 30 \* its own number.

In the RACE procedure, we've used WHO twice, once to get the turtles into position and once to get them moving.



And the winner is Turtle 3 by a long shot! Poor Turtle 0 never leaves the starting block! Can you explain why?

Experiment with having the turtles move at various speeds and in various directions.

After your students have had time to experiment with motion, introduce the collision detection capability.

### Setting up Collisions

There are 22 types of collisions, numbered 0 to 21 depending on which turtles or which turtle and which drawings are involved in the collision. By using the collision detection capability, we can cause all kinds of things to happen. It is another form of the IF-THEN conditionals we covered earlier, only this time you'll use the command:

### WHEN

Think of the collisions as the WHEN's. **You** will define the THEN's. For example, you could say "WHEN Turtle 0 bumps into Turtle 1, then turn around and go the opposite direction." Since a collision between Turtle 0 and Turtle 1 is collision number 19, you could write the command as WHEN 19 [RT 180].



Look at the following Table of Collisions and Events. It shows the possible collisions. (For now, don't worry about the special events — we'll cover them later.)

## Table of Collisions and Events

Code Number		Inputs		
Collision	Special event	Turtle number	Pen number	Description of event
0		0	0	
1		0	1	
2		0	2	
	3			Button on Joystick is pressed
4		1	0	
5		1	1	
6		1	2	
	7			Once per second
8		2	0	
9		2	1	
10		2	2	
11				Not used
12		3	0	
13		3	1	
14		3	2	
	15			Joystick position is changed
Collision number		Turtle number	Turtle number	
16		3	0	
17		3	1	
18		3	2	
19		0	1	
20		0	2	
21		1	2	

Collision 0 occurs when Turtle 0 bumps into something drawn with pen number 0. Collision 1 occurs when Turtle 0 bumps into something drawn with pen number 1. Collision 16 occurs when Turtle 3 bumps into Turtle 0. If you wanted to set up a collision between Turtle 2 and something drawn with pen number 2, what collision would you use?

Let's set up a very simple collision using collision number 0. Have Turtle 0 draw a vertical line with Pen 0. Then face him RT 90 and set him in motion. WHENever he touches the line, make him turn around and go the other way. You can do that by saying WHEN 0 [RT 180].

```
TO FRUSTRATE
  TELL 0
  SETPN 0 PD
  FD 50 BK 100 FD 50
  RT 90
  PU
  SETSP 30
  WHEN 0 [RT 180]
  END
```

The procedure will keep running until you type CS. CS will not only clear the screen and return the turtle to HOME, it will also reset the turtle's speed to 0.

Experiment with some of the other possible collisions. Can you capture a turtle inside a box? Here's one possible solution:

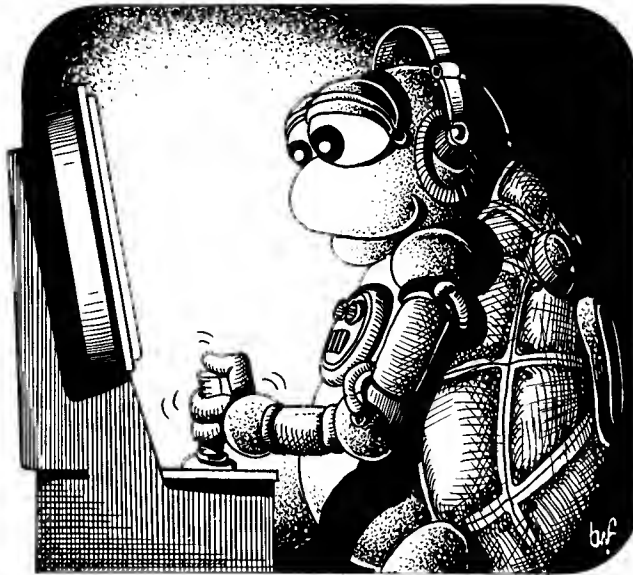
```
TO BOX
  TELL 1 SETPN 1
  PD REPEAT 4 [FD 60 RT 90]
  PU RT 45 FD 30
  SETSP 30
  WHEN 5 [SETH HEADING + 160]
  END
```

What happens if you set up a collision and then talk to a different turtle? Try it. What happens if you set up a collision and talk to two turtles at the same time? Set up a collision and talk to two turtles who are moving at different speeds.

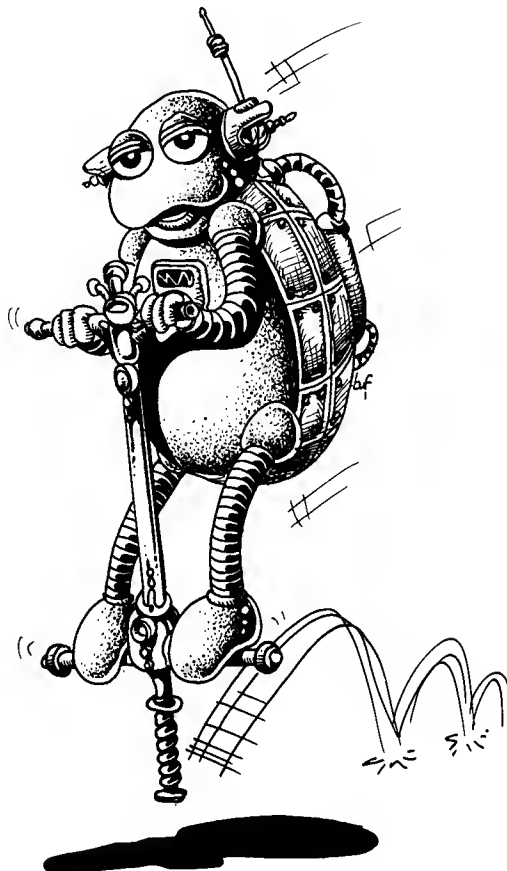
Here is an example of "capturing" one turtle and having the other three mimic its motions at different speeds. It results in a graphic demonstration of differences in speed. The turtles each draw the same design, but each design is a different size since the turtles are moving at different speeds.

```
TO CAPTURE
  TELL 0 SETPN 0 PD
  REPEAT 6 [FD 50 RT 60]
  PU RT 45 FD 10
  TELL 1 PU SETX -80 RT 45 SETSP 30
  TELL 2 PU SETPOS [-100 60] RT 45 SETSP 20
  TELL 3 PU SETPOS [-20 80] RT 45 SETSP 10
  TELL 0 SETSP 40K
  TELL [0 1 2 3] SETPN 2 PD
  WHEN 0 [RT 145]
  END
```

There are two possible events (rather than collisions) that involve using a joystick. An example of event 15 is included in the **Atari Reference Manual** on page 115. The procedure enables you to draw using the joystick. It might be especially fun for pre-readers who want to work with the turtle.

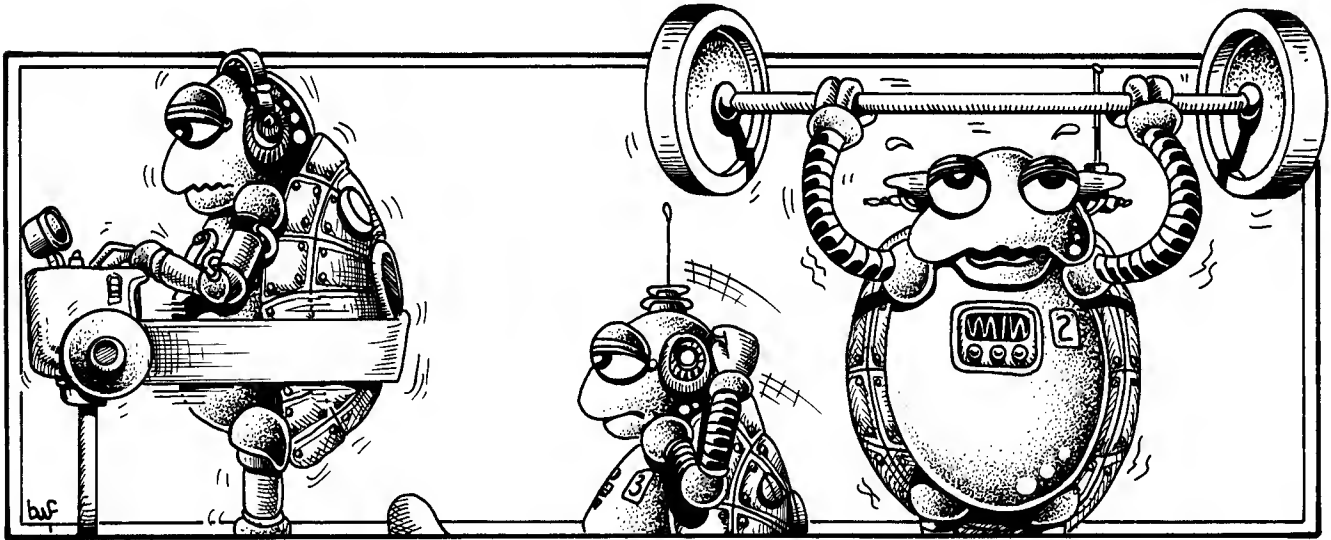


An example of the joystick button event (3) is on page 116 of the **Atari Reference Manual**. When you press the joystick button, the turtle acts like a spring.



Advanced students might take a look at the ATARI LOGO Car Race in Appendix C of **Introduction to Programming Through Turtle Graphics.**

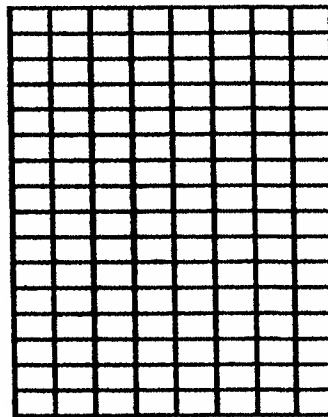
Before we go on to other collisions or events, let's look at another capability of ATARI LOGO.



### Changing Turtle Shapes

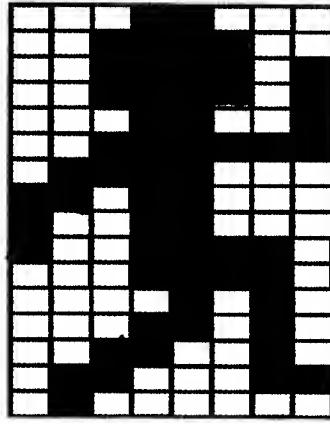
So far, we have used the turtles as turtles. We have had them do all sorts of things either visibly or invisibly. But they've always been turtles. Now we're going to learn to change them into something else.

To EDit a SHape, type EDSH 1 and press return. You will see a grid on the screen with the cursor in the upper left hand corner.



Use CTRL and the four arrow keys to move the cursor. If you want to fill in a rectangle, press the space bar. (Or if you want to erase a filled rectangle, press the space bar.) Try moving the cursor and filling and erasing rectangles to make a shape.

Here's a design for a KID:

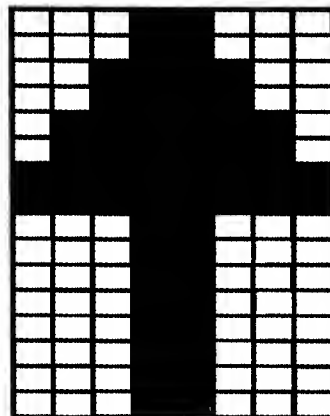


When you have a shape designed, pressed the ESC key. Now type:

```
TELL 0  
SETSH 1
```

and you should see your new shape on the screen. You can command this new shape just as you did the turtle. There's one big difference between using the shape you have defined and using the turtle. The turtle will turn and face the direction you tell him. Your new shape will always remain in the same position it was when you designed it.

For example, design an arrow pointing up:



As you give right and left commands, the arrow will remain pointing toward the top of the screen, but its "drawing mechanism" will rotate and it will draw in the direction you tell it to.

Allow time for the students to experiment with designing new shapes and using them to draw or simply move around the screen by setting the speed. As many as fifteen different shapes can be defined, but of course there are only four turtles, so you can only use four of them at a time.

If you want to save the shapes you define, you'll use the GETSH (GET SHape) command. Just as you do with procedures, you'll have to give your shape a name. For example, if you want to save the KID we designed, you say:

MAKE "KID GETSH 1

Then you can save it, along with any procedures, on cassette or diskette. When you load the procedures from the diskette or cassette, you'll have to PUT the shape back into a shape number by using:

PUTSH 1 :KID

If you do not save your shapes by giving them names and using the GETSH operation, they will be lost when you turn off the computer.

Let's use the KID shape we defined in a procedure. Event 7 causes something to happen every second. perhaps we could use it to make some KIDs dance:

```
TO DANCE
PUTSH 1 :KID
TELL [0 1 2 3] ST
SETSH 1 PU
EACH [SETH WHO * 90]
SETSP 40
WHEN 7 [EACH [SETH HEADING + 45]]
END
```

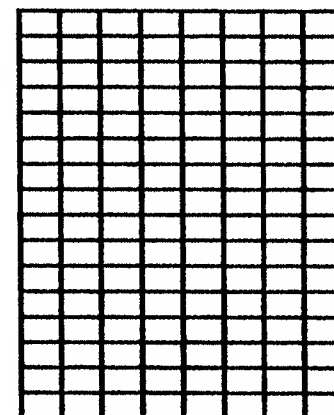
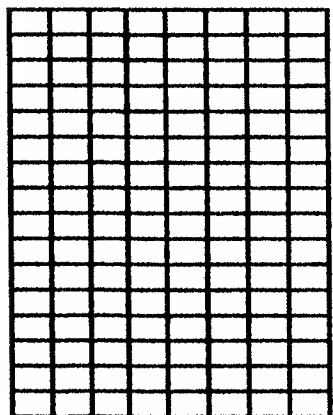
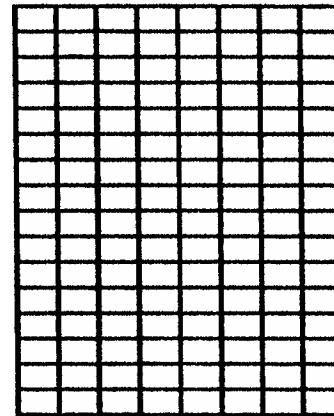
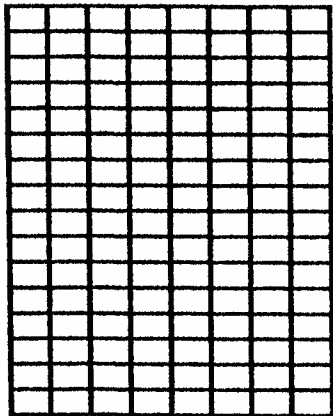
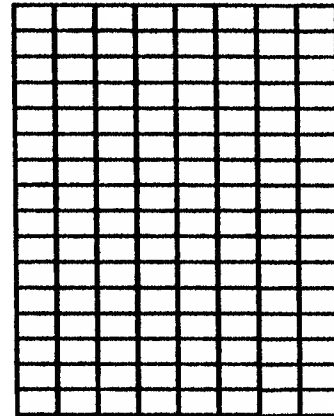
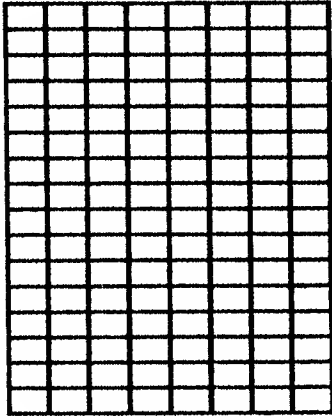


Allow a lot of time to experiment with collisions and events. This is an opportunity for imaginations to run wild. If the student can define the problem he or she is trying to solve, there is probably a way to solve it. Encourage students to define their own projects. They might pick a subject or hobby they are interested in and somehow illustrate it on the screen. Redefine the turtles as stars in outer space and have them collide, causing an explosion with the screen flashing different colors. Draw a racquetball court, redefine a turtle as a ball and have it bounce around all day within the court. Design a new video game. The possibilities are endless, limited only by your imagination.



# Redesigning the Turtle

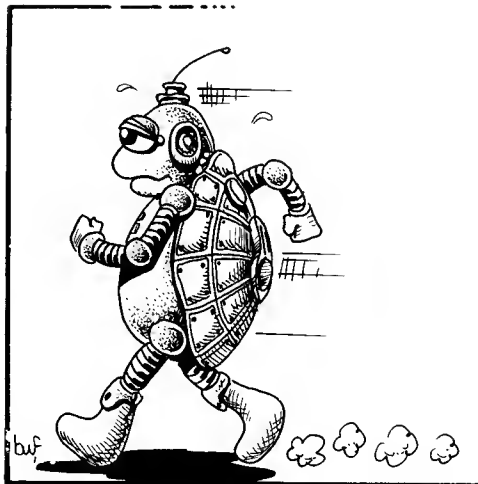
Using the following grids, sketch several new designs for the turtle. Using the EDSH command, redesign the turtle. Have the turtle carry the new shape with the command SETSH followed by the number of the shape. Define a procedure with a collision using your new shape.





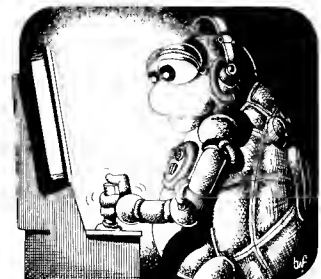
# Collisions

Using the collision detection commands, write a procedure to capture a moving turtle inside a cage. Write down your procedures.



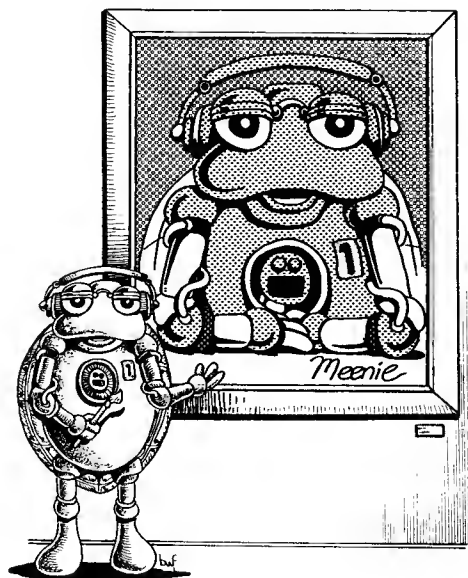
# Collisions

Design a simple game or maze using one or more of the collision detection commands.

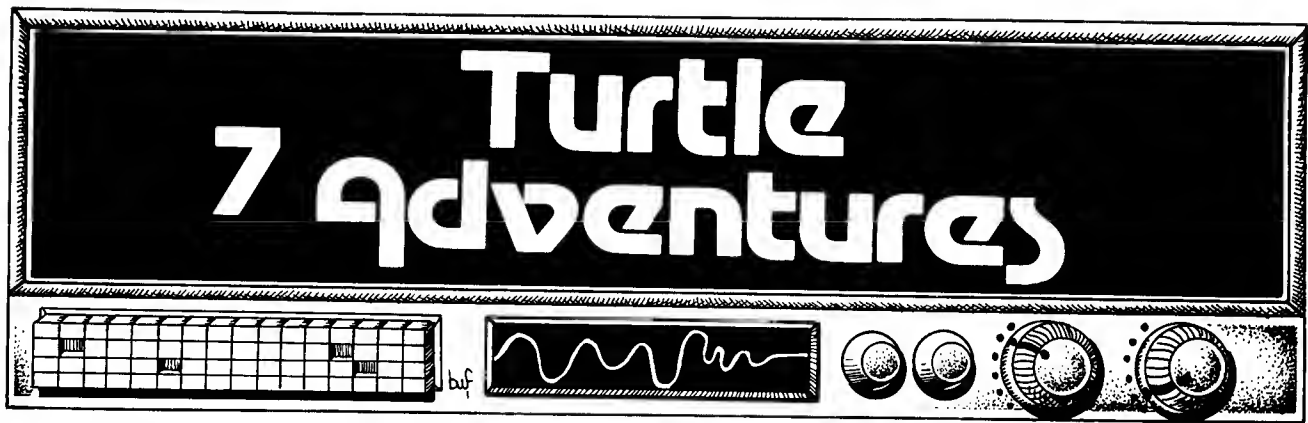


# Doodles and Designs

A page for keeping notes of doodles and designs and how they were made.







Let's take a look at another feature of ATARI LOGO. Then we'll expand many of the things we've learned and use them to develop an interactive story or adventure.

### Music

What better way to introduce an adventure than with a little music!

ATARI LOGO can play two part harmony. The command for a musical tone is TOOT followed by four numbers. The first number is either 0 or 1 and designates which "voice" you're talking to. The second number is the frequency, specified in Hertz (cycles per second abbreviated Hz.) Frequency can range from 14 to 64,000 Hz. Following is a chart showing frequency values and notes for four octaves.

Frequency	Note	Frequency	Note
110	A	440	A (above middle C)
117	A <sup>#</sup> , B <sup>b</sup>	466	A <sup>#</sup> , B <sup>b</sup>
123	B	494	B
131	C (low C)	523	C (high C)
139	C <sup>#</sup> , D <sup>b</sup>	554	C <sup>#</sup> , D <sup>b</sup>
147	D	587	D
156	D <sup>#</sup> , E <sup>b</sup>	622	D <sup>#</sup> , E <sup>b</sup>
165	E	659	E
175	F	698	F
185	F <sup>#</sup> , G <sup>b</sup>	740	F <sup>#</sup> , G <sup>b</sup>
196	G	784	G
208	G <sup>#</sup> , A <sup>b</sup>	831	G <sup>#</sup> , A <sup>b</sup>
220	A (below middle C)	880	A (above high C)
220	A (below middle C)	880	A (above high C)
233	A <sup>#</sup> , B <sup>b</sup>	932	A <sup>#</sup> , B <sup>b</sup>
247	B	988	B
262	C (middle C)	1047	C
277	C <sup>#</sup> , D <sup>b</sup>	1109	C <sup>#</sup> , D <sup>b</sup>
294	D	1175	D
311	D <sup>#</sup> , E <sup>b</sup>	1245	D <sup>#</sup> , E <sup>b</sup>
330	E	1319	E
349	F	1397	F
370	F <sup>#</sup> , G <sup>b</sup>	1480	F <sup>#</sup> , G <sup>b</sup>
392	G	1568	G
415	G <sup>#</sup> , A <sup>b</sup>	1661	G <sup>#</sup> , A <sup>b</sup>
440	A (above middle C)	1760	A

The third number in the TOOT command is the volume. It can range from 0 to 15. Duration, the fourth number, can range from 0 to 255. It is measured in units of one-sixtieth of a second.

Try a few tones:

```
TOOT 0 440 15 60
TOOT 1 30 B 120
TOOT 0 1000 15 20
```

Even a simple melody can be pretty complicated to program. However, by now you and your students should be in the habit of breaking problems down into their simplest elements. Suppose we want to play "California, Here I Come." Pick it out on the piano or hum it and you can find some musical phrases that are repeated. We can play the melody with only eight notes. So, let's write a procedure for each of those notes, with a variable for the duration of the note. Then we'll write procedures for each musical phrase and put the whole song together. The eight notes we'll need and procedures for them are:

```
TO C# :DURATION
TOOT 0 277 15 :DURATION
END
```

```
TO D :DURATION
TOOT 0 294 15 :DURATION
END
```

```
TO E :DURATION
TOOT 0 330 15 :DURATION
END
```

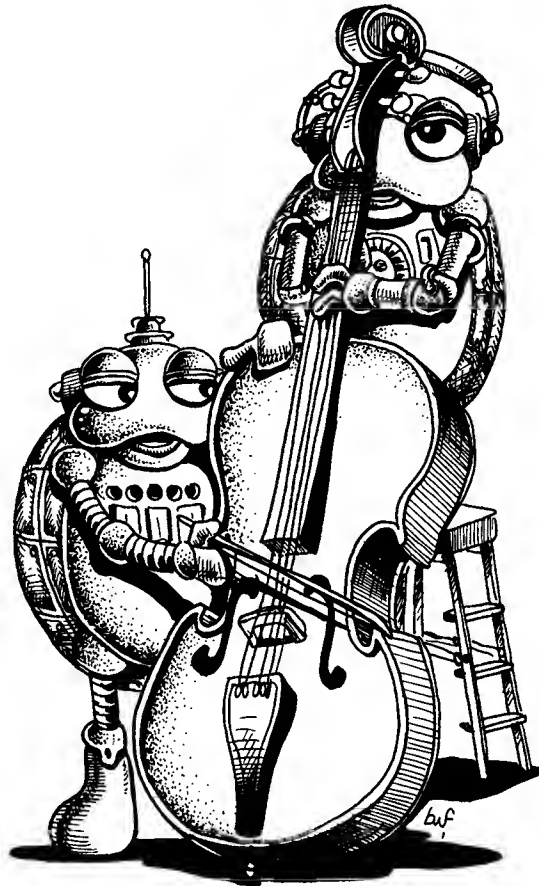
```
TO F# :DURATION
TOOT 0 370 15 :DURATION
END
```

```
TO G# :DURATION
TOOT 0 415 15 :DURATION
END
```

```
TO A :DURATION
TOOT 0 440 15 :DURATION
END
```

```
TO B :DURATION
TOOT 0 494 15 :DURATION
END
```

```
TO H1.C# :DURATION
TOOT 0 554 15 :DURATION
END
```



Now, instead of typing TOOT followed by four numbers, we can simply type the actual note we need and the duration for that note.

The first musical phrase begins with the same note repeated four times, which presents a problem. Try repeating a tone four times: REPEAT 4 [A 30]. Rather than four distinct notes, they melt into one long note. If we could just put a tiny little pause between the notes...Aha!

```
TO REST
TOOT 0 14 0 1
END
```

We've put the volume at 0 and the duration at 1, just long enough to put a little break between notes. Since we have to include a frequency, we used 14. It doesn't really matter what frequency we use since the volume is at 0.

Now we can define the four musical phrases and the superprocedure CALIFORNIA:

TO CALIFORNIA

LINE1

LINE2

LINE3

LINE1

LINE2

LINE4

END

TO LINE1

REPEAT 4 [A 30 REST]

B 20 A 30 F# 60

END

TO LINE2

REPEAT 4 [E 30 REST]

F# 20 E 30 C# 60

END

TO LINE3

E 20 F# 20 E 30

F# 20 G# 20 F# 30

G# 20 A 20 B 20 E 30 REST

E 20 F# 20 G# 20

END

TO LINE4

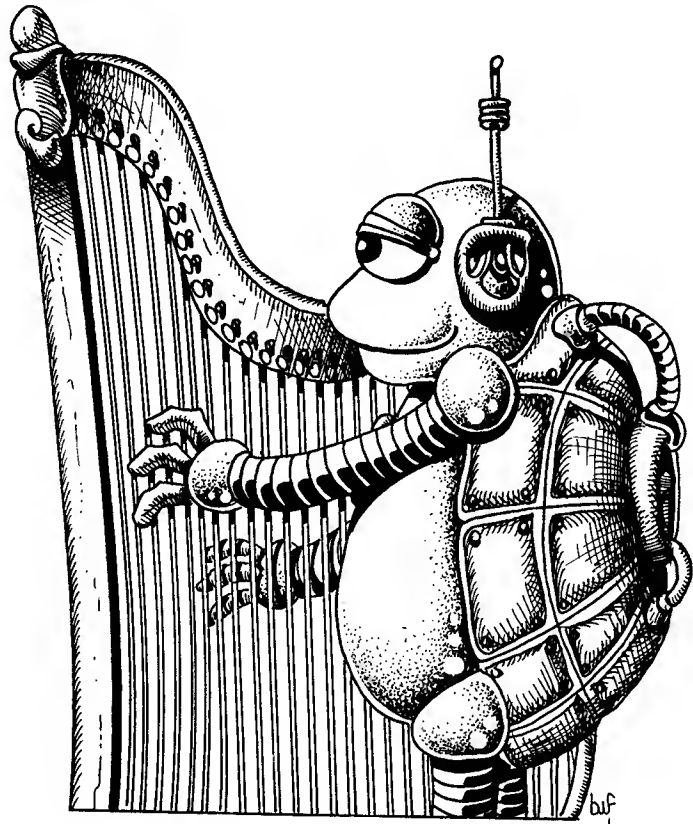
D 30 E 30 F# 30 A 30

H!C# 20 B 30 A 40

F# 20 G# 20 A 30 F# 30

A 20 B 30 A 60

END



Play around with it; improve it; add harmony. Program some other songs or make up your own.

## Another Look at RANDOM

Remember when we used RANDOM in the target game to select a random position on the screen? Let's explore a little more with RANDOM. As you will recall, RANDOM tells the computer to pick a number, any number up to whatever limit you put on it. What would happen if you defined a square with RANDOM for the size of each side? Can you predict what will happen with the following procedure? Try it. Were you right?

```
TO SQUARE  
  REPEAT 4 [FD RANDOM 80 RT 90]  
END
```

Have you ever seen a picture of a computer chip? Try REPEAT 150 [SQUARE]

You could have the computer pick a random number between 0 and 100:

```
TO PICK  
  PRINT RANDOM 100  
END
```

Can you design a "Guess my Number" game using RANDOM?



How about creating some random music! The first number following TOOT must be 0 or 1. The second can be anything from 14 to 64,000. The third can be anything up to 15, and the fourth can be anything up to 255.

The third and fourth numbers in TOOT can be anything from 0 to the delimiter. The frequency number, however, cannot be below 14, so we have to ask for a RANDOM number and then add 14 to it. Let's use 2000 as the delimiter and keep the notes within a comfortable hearing range.



```
TO TONE
TOOT 0 RANDOM 2000 + 14 RANDOM 15 RANDOM 255
END
```

To generate some random music, simply repeat TONE several times:

```
TO RAN.MUSIC
REPEAT 20 [TONE]
END
```

Want some especially creepy music for a haunted house? Define TONE with low and slow notes:

```
TO TONE
TOOT 0 RANDOM 500 + 14 RANDOM 15 RANDOM 255 + 100
END
```

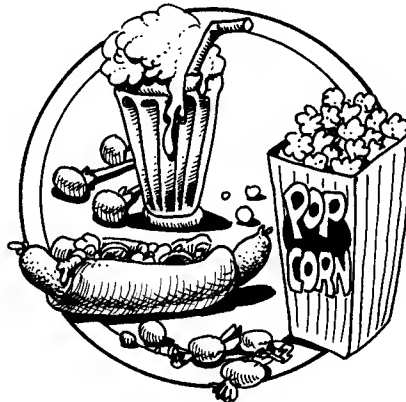
How would you define TONE if you want some random carnival music that's light and lively?

### More on Interactive Procedures

How could we combine some music, random or otherwise, with some graphics and some text to write a story? How could we involve someone else in the story by providing choices for them to make? Let's take a look at a few more text commands that will help make a story like this possible.

Earlier we learned how to use RL to allow a user to type an input. The computer waited for the input and then combined it with a preprogrammed message. Instead of repeating it right away, let's have the computer remember the input and use it later as part of a story. We might ask a series of questions like:

What is your name?  
What is your favorite food?

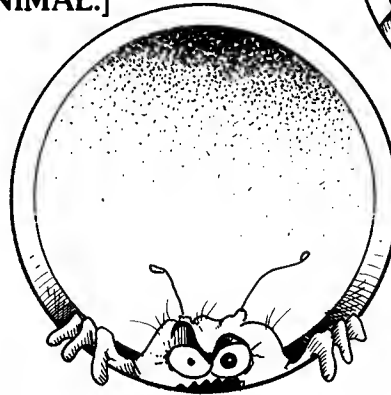


What is your least favorite animal?  
Who is your best friend?  
If you could go anywhere in the world, where would you go?

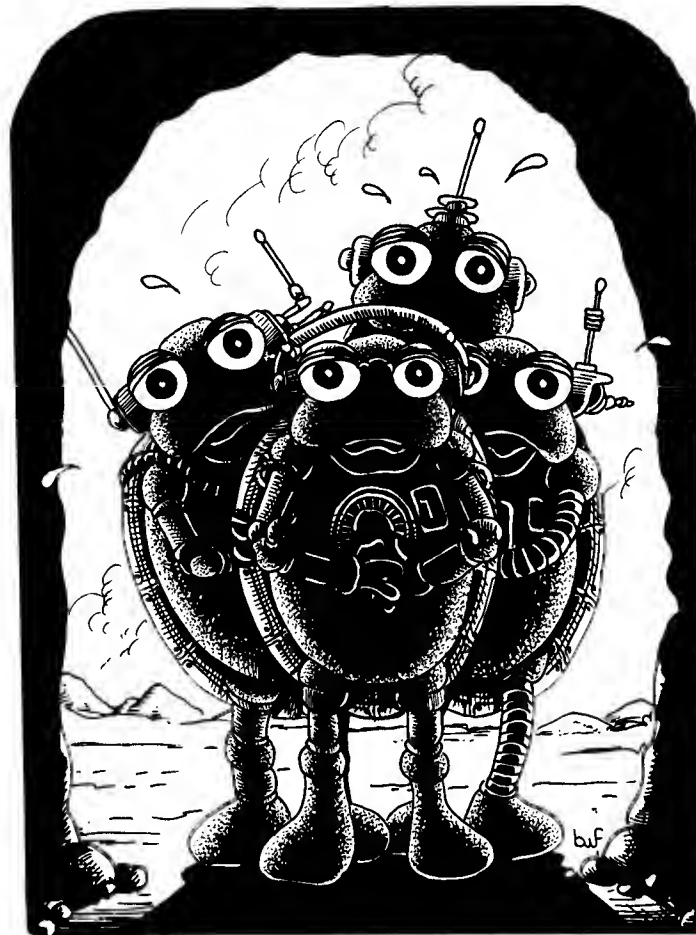
Then we would use the answers to the questions as part of a story. Here's the beginning of an interactive story:

```
TO STORY
QUESTIONS
TELL.IT
END
```

TO QUESTIONS  
 PRINT [WHAT IS YOUR NAME?]  
 MAKE "NAME RL  
 PRINT [ WHO IS YOUR BEST FRIEND?]  
 MAKE "FRIEND RL  
 PRINT [WHAT COUNTRY WOULD YOU LIKE TO VISIT?]  
 MAKE "COUNTRY RL  
 PRINT [NAME A FRIGHTENING ANIMAL.]  
 MAKE "ANIMAL RL  
 END



TO TELL.IT  
 PRINT SE [ONCE ON A DAY LAST SUMMER,] :NAME  
 PRINT SE [AND] :FRIEND  
 PRINT SE [WERE VISITING] :COUNTRY  
 PRINT [ ]  
 PRINT [THEY CAME UPON A LARGE CAVE]  
 PRINT [AND DECIDED TO GO IN.]  
 PRINT [AS THEY ENTERED THE CAVE,]  
 PRINT SE [THEY TRIPPED OVER THE] :ANIMAL  
 PRINT [WHO LAY SLEEPING IN THE ENTRANCE.]  
 END



MAKE gives a name to a variable, enabling the computer to store it and recall it later. In the first example, whatever the user types in as a response to the question "What is your name?" is stored under the name NAME. Later it is called up and combined as part of a sentence:

```
PRINT SE [ONCE ON A DAY LAST SUMMER,] :NAME
```

Try the story as far as it goes and then have your students add to it.

Develop other interactive procedures using PRINT, SE, RL, and MAKE.

### An Illustrated, Interactive, Musical Mad-Lib

Often in LOGO, one idea leads to another. Thus, a project may never be finished, but always in progress. For example, let's start with a simple song that might be illustrated. How about "Jack and Jill?"

The melody has a total of 28 notes — that's simple enough. In order to play the tune, we'll need procedures for all the notes from B below middle C to E above high C. Let's add simple harmony by playing the same note an octave lower. If you experiment a little, you will find that the lower note will overshadow the higher one unless you "turn its volume down." Here are the tone procedures we'll need for "Jack and Jill." As we did earlier, we'll make the duration a variable. We'll also need a REST to place between repeated notes.

```
TO B :D
TOOT 0 247 15 :D
TOOT 1 123 10 :D
END
```

```
TO A :D
TOOT 0 440 15 :D
TOOT 1 220 10 :D
END
```

```
TO C :D
TOOT 0 262 15 :D
TOOT 1 131 10 :D
END
```

```
TO HLB :D
TOOT 0 494 15 :D
TOOT 1 247 10 :D
END
```

```
TO D :D
TOOT 0 294 15 :D
TOOT 1 147 10 :D
END
```

```
TO HLC :D
TOOT 0 523 15 :D
TOOT 1 262 10 :D
END
```

```
TO E :D
TOOT 0 330 15 :D
TOOT 1 165 10 :D
END
```

```
TO HLD :D
TOOT 0 587 15 :D
TOOT 1 294 10 :D
END
```

```
TO F :D
TOOT 0 349 15 :D
TOOT 1 175 10 :D
END
```

```
TO HLE :D
TOOT 0 659 15 :D
TOOT 1 330 10 :D
END
```

```
TO G :D
TOOT 0 392 15 :D
TOOT 1 196 10 :D
END
```

```
TO REST
TOOT 0 14 0 1
END
```

Let's try it!

TO SONG

HI.C 30 G 15 A 30

G 15 HI.C 30 G 15 A 30

G 15 HI.E 30 HI.D 15

HI.C 30 HI.B 15 A 45 G 45

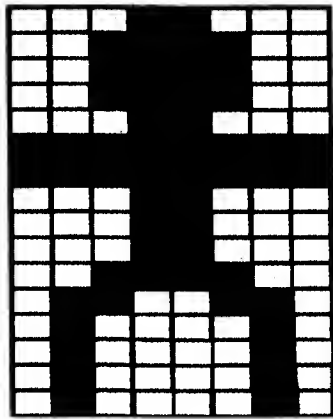
F 30 REST F 15 HI.D 45

F 15 E 30 REST E 15 HI.C 45

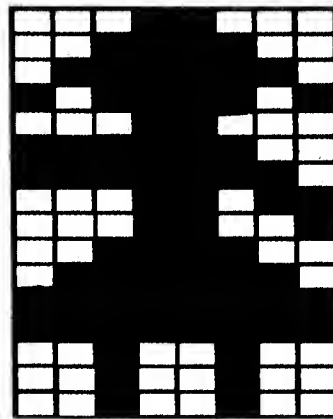
E 15 D 30 A 15 G 30 B 15 D 45 C 60

END

How about redesigning one of the turtles to be Jack and another turtle to be Jill?



EDSH1



EDSH2

TO CHARACTERS

MAKE "JACK GETSH 1

MAKE "JILL GETSH 2

PUTSH 1 :JACK

TELL 0 SETSH 1

PUTSH 2 :JILL

TELL 1 SETSH 2 SETC 40

END

Play around with Jack and Jill. Set their heading and speed and see if you can figure out home to make them climb a hill. Perhaps you might add the following commands to be previous procedure:

TELL [0 1] PU SETPOS [-40 -30] ST

TELL 1 RT 90 FD 20

TELL [0 1] SETH 45 SETSP 15

Oops! We need a hill, and a well! Here's one possibility:

TO HILL  
TELL 3 PU SETPOS [-70 -30]  
PD SETPC 0 96  
RT 90 FD 30 LT 45  
REPEAT 2 [FD 30 RT 10]  
FD 30 RT 5 FD 40  
RT 10 FD 40  
END

TO WELL  
TELL 2 PU  
SETPOS [50 20] PD  
SETPN 1 SETPC 1 16  
BASE  
TOP  
END

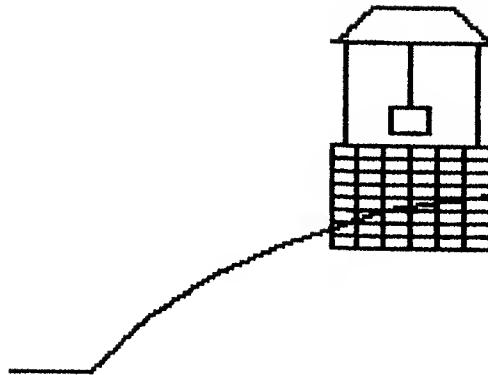
TO BASE  
REPEAT 8 [BRICKS LT 90 FD 50 RT 90 FD 5]  
END

TO BRICKS  
REPEAT 5 [REC RT 90 FD 10 LT 90] REC  
END

TO REC  
REPEAT 2 [FD 5 RT 90 FD 10 RT 90]  
END

TO TOP  
RT 90 FD 5 LT 90 FD 40  
RT 90 BK 5 FD 30 BUCKET FD 25  
RT 90 FD 40 BK 40 LT 90 FD 5  
LT 135 FD 20 LT 45 FD 30  
LT 45 FD 20  
END

TO BUCKET  
RT 90 FD 25 LT 90 FD 7  
RT 90 FD 10 RT 90 FD 14  
RT 90 FD 10 RT 90 FD 7  
LT 90 FD 25 RT 90  
END



But, we need to have Jack and Jill climbing up the hill during the first part of the song and coming down, one at a time, during the second part of the song. Hmm... Suppose we break the song down into three parts called FETCH, JACK, and JILL to represent the three parts of the story. Then we could include the motions of the characters with the appropriate parts of the song. FETCH could be defined with the first four lines from SONG. JACK would contain the next two lines of SONG, and should also contain commands to tell the

characters to stop and Jack to turn around. JILL would include the last line of SONG, as well as command Jill to go back down the hill.

TO FETCH  
H.I.C 30 G 15 A 30  
G 15 H.I.C 30 G 15 A 30  
G 15 H.I.E 30 H.I.D 15  
H.I.C 30 H.I.B 15 A 45 G 45  
END

TO JACK  
SETSP 0  
TELL 0 SETH -135 SETSP 25  
F 30 REST F 15 H.I.D 45  
F 15 E 30 REST E 15 H.I.C 45  
END

TO JILL  
SETSP 0 TELL 1 SETH -135 SETSP 25  
E 15 D 30 A 15 G 30 B 15 D 45 C 60  
SETSP 0  
END

So, we could now put it all together like this:

TO J.AND.J  
TELL [0 1 2 3] HT PU CS  
HILL  
WELL  
CHARACTERS  
FETCH  
JACK  
JILL  
END

Suppose, however, that we want to make the entire sequence into a “mad-lib,” offering our audience a chance to participate in the story. Using PR, SE MAKE, and RL, we’ll ask for some inputs and plug them into the story in the appropriate places.

TO QUESTIONS  
PR [NAME A BOY.]  
MAKE “BOY RL  
PR [NAME A GIRL.]  
MAKE “GIRL RL  
PR [NAME SOMETHING TO CLIMB ON.]  
MAKE “CLIMB RL  
PR [WHAT IS YOUR FAVORITE DRINK?]  
MAKE “DRINK RL  
PR [NAME A PART OF THE BODY.]  
MAKE “BODY RL  
PR [NAME A VERB ENDING IN ING.]  
MAKE “VERB RL  
END

Now add the following commands to the beginning of the J.AND.J procedure:

## CT TS QUESTIONS

And these commands to the beginning of the FETCH procedure:

```
PR (SE :BOY [AND] :GIRL)
PR SE [WENT UP THE] :CLIMB
PR SE [TO FETCH A PAIL OF] :DRINK
```

These should be added to the JACK procedure:

```
CT
PR SE :BOY [FELL DOWN]
PR SE [AND BROKE HIS] :BODY
```

This should be added to the JILL procedure:

```
PR (SE [AND] :GIRL [CAME] :VERB [AFTER.] )
```

And that's what you call an illustrated, interactive, musical mad-lib! (The entire procedures are included in the appendix.)

## Quizzes and Word Games

By adding conditional commands, you could develop quizzes and word games and have the computer check the answers to see whether or not they're correct.

Suppose, for example, you are studying the Panama Canal. Have the students make up questions and put them into procedures. Then put all the question procedures into a CANAL.QUIZ.

```
TO CANAL.QUIZ
QUESTION1
QUESTION2
QUESTION3
END
```

```
TO QUESTION1
CT TS      (Clear Text, Text Screen)
PRINT [WHEN DID THE FIRST SHIP SAIL]
PRINT [THROUGH THE PANAMA CANAL?]
PRINT [ ]  (This will result in a blank line.)
PRINT [A. 1900]
PRINT [B. 1914]
PRINT [C. 1928]
PRINT [ ]
IF RL = [B] [PRINT [RIGHT!]] [PRINT [NO, IT WAS AUGUST 15, 1914]]
END
```

TO QUESTION2

CT TS

PRINT [HOW MANY YEARS DID IT TAKE]

PRINT [THE U.S. TO BUILD THE CANAL?]

PRINT [ ]

IF RL = [10] [PRINT [VERY GOOD!]] [PRINT [IT TOOK 10 YEARS.]]

END

TO QUESTION3

CT TS

PRINT [WHICH ENTRANCE IS WEST OF THE OTHER ONE?]

PRINT [TYPE ATLANTIC OR PACIFIC.]

PRINT [ ]

IF RL = [ATLANTIC] [PRINT [RIGHT!]] [PRINT [IT MAY SEEM STRANGE, BUT  
THE ATLANTIC ENTRANCE IS WEST OF THE PACIFIC ENTRANCE.]]

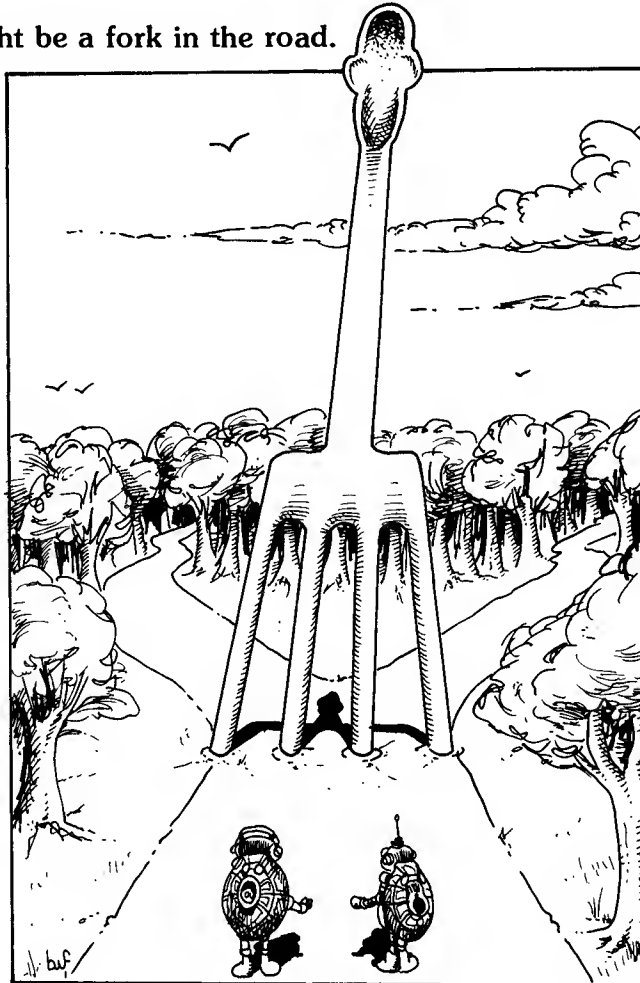
END

Select a topic you are studying and have your students develop a quiz. How about a vocabulary or spelling test?

### Adventure Games

This is a good class project on which several students could work cooperatively. It offers an opportunity to combine many different features of LOGO, including graphics, text, recursion, music, collisions, etc. An interactive story can be developed similar to adventure games, offering the user choices to make and consequences of those choices.

For example, there might be a fork in the road.





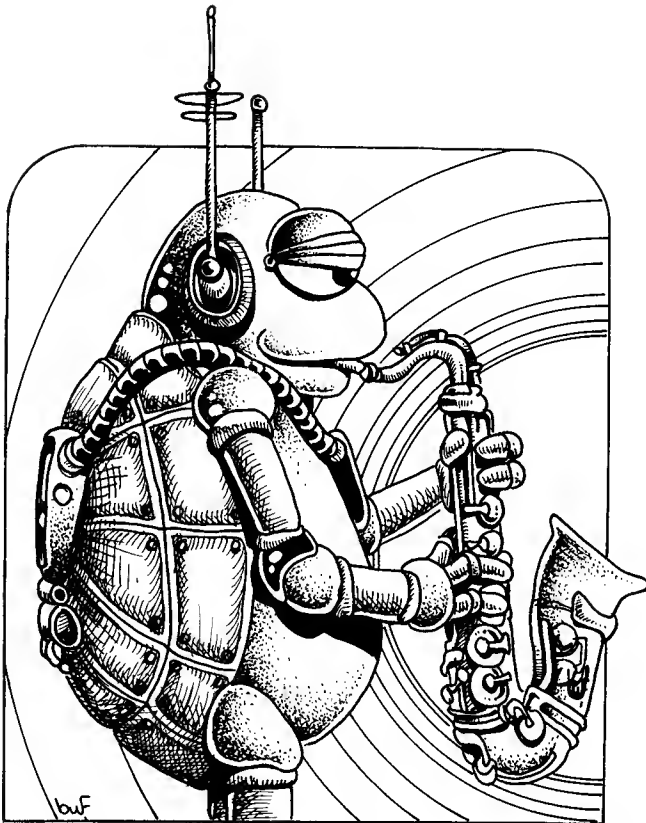
```
TO FORK
PRINT [THERE'S A FORK IN THE ROAD.]
PRINT [DO YOU WANT TO GO NORTH OR EAST?]
IF RL = [NORTH] [NORTH] [EAST]
END
```

If the user types NORTH, the computer goes to the NORTH procedure. If he doesn't type NORTH, it goes to the EAST procedure. Now you have a fork in your program and you'll have to develop two different adventures or have one end in a dead end. Perhaps NORTH could take the user to a haunted house complete with spooky music. East might take him to a planetarium where he looks through a giant telescope at stars, spirals, the milky way, and black holes.

Discuss some ideas with your students and have them work in pairs or teams to develop portions of the adventure. Then put the pieces together into one fantastic story or adventure, one that includes contributions from every student in your class.

# Music

Select a tune and write procedures to have the computer play it.



# Writing Stories

Write a short interactive story using PRINT, SE, RL, and MAKE. Add some graphics to your story.

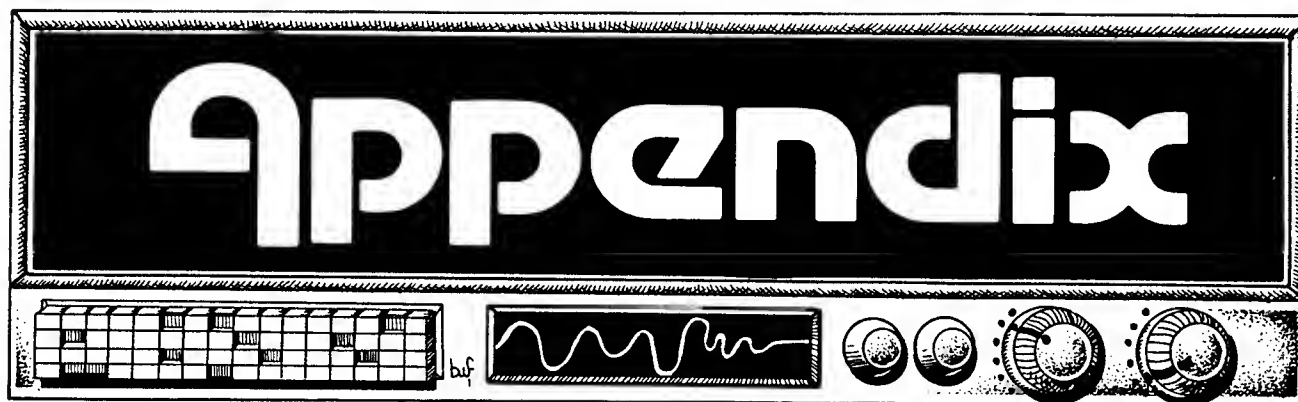
# Mad-libs

Select a nursery rhyme and turn it into a mad-lib. Here's an example:

TO MAD.LIB  
QUESTIONS  
ANSWERS  
END

TO QUESTIONS  
PR [NAME A BOY.]  
MAKE "BOY RL  
PR [NAME A GIRL.]  
MAKE "GIRL RL  
PR [NAME SOMETHING TO CLIMB ON.]  
MAKE "CLIMB RL  
PR [WHAT IS YOUR FAVORITE DRINK?]  
MAKE "DRINK RL  
PR [NAME A PART OF THE BODY.]  
MAKE "BODY RL  
PR [NAME A VERB ENDING IN ING.]  
MAKE "VERB RL  
END

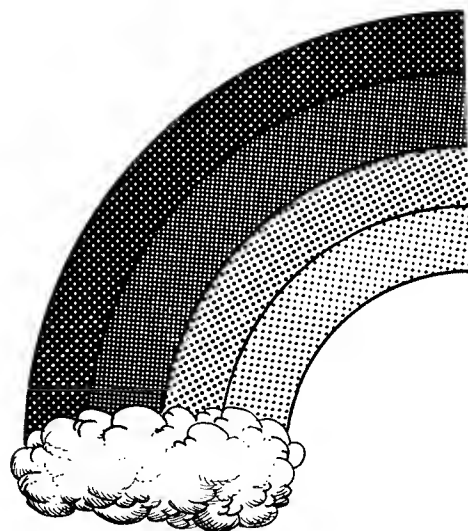
TO ANSWERS  
CT  
PR (SE :BOY [AND] :GIRL)  
PR SE [WENT UP THE] :CLIMB  
PR SE [TO FETCH A PAIL OF] :DRINK  
PR SE :BOY [FELL DOWN]  
PR SE [AND BROKE HIS] :BODY  
PR (SE [AND] :GIRL [CAME] :VERB [AFTER])  
END



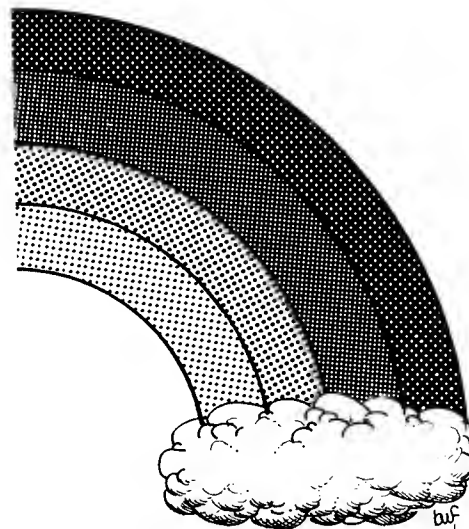
# ATARI Editing Features

<b>BREAK</b>	Aborts whatever LOGO is doing. If editing, changes made in edit buffer will be ignored. Clears the line currently being typed at the top level.
<b>CTRL →</b>	Moves the cursor one position to the right.
<b>CTRL ←</b>	Moves the cursor one position to the left.
<b>CTRL ↑</b>	Moves the cursor up to the previous line.
<b>CTRL ↓</b>	Moves the cursor down to the next line.
<b>CTRL 1</b>	Makes LOGO stop scrolling until CTRL 1 is typed again.
<b>CTRL A</b>	Moves the cursor to the beginning of the current line.
<b>CTRL CLEAR</b>	Deletes text from the cursor position to the end of the current line.
<b>CTRL DELETE BACK S</b>	Erases the character at the cursor position.
<b>CTRL E</b>	Moves the cursor to the end of the current line.
<b>CTRL INSERT</b>	Opens a new line at the position of the cursor.
<b>CTRL V</b>	Scrolls screen to next page in editor.
<b>CTRL W</b>	Scrolls screen back to previous page in editor.
<b>CTRL X</b>	Moves the cursor to the beginning of editor.
<b>CTRL Y</b>	In the editor, inserts the contents of the delete buffer. Outside the editor, inserts the last command typed.
<b>CTRL Z</b>	Moves the cursor to the end of the editor.
<b>DELETE BACK S</b>	Erases the character to the left of the cursor.
<b>ESC</b>	Completes editing and exits to top level.
<b>SHIFT DELETE BACK S</b>	Deletes text from the cursor position to the end of the current line.
<b>SHIFT INSERT</b>	Opens a new line at the position of the cursor.

# Color Chart



0-7	gray
8-15	light orange (gold)
16-23	orange
24-31	red-orange
32-39	pink
40-47	purple
48-55	purple-blue
56-63	blue
64-71	blue
72-79	light blue
80-87	turquoise
88-95	green-blue
96-103	green
104-111	yellow-green
112-119	orange-green
120-127	light-orange



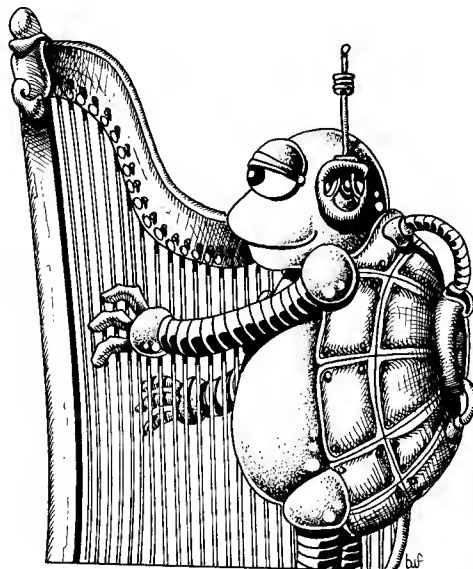
# Table of Collisions and Events

Code Number		Inputs		Description of event
Collision	Special event	Turtle number	Pen number	
0		0	0	
1		0	1	
2		0	2	
	3			Button on Joystick is pressed
4		1	0	
5		1	1	
6		1	2	
	7			Once per second
8		2	0	
9		2	1	
10		2	2	
11				Not used
12		3	0	
13		3	1	
14		3	2	
	15			Joystick position is changed
Collision number		Turtle number	Turtle number	
16		3	0	
17		3	1	
18		3	2	
19		0	1	
20		0	2	
21		1	2	



# Musical Tone Frequencies

<i>Frequency</i>	<i>Note</i>	<i>Frequency</i>	<i>Note</i>
110	A	440	A (above middle C)
117	A <sup>#</sup> , B <sup>b</sup>	466	A <sup>#</sup> , B <sup>b</sup>
123	B	494	B
131	C (low C)	523	C (high C)
139	C <sup>#</sup> , D <sup>b</sup>	554	C <sup>#</sup> , D <sup>b</sup>
147	D	587	D
156	D <sup>#</sup> , E <sup>b</sup>	622	D <sup>#</sup> , E <sup>b</sup>
165	E	659	E
175	F	698	F
185	F <sup>#</sup> , G <sup>b</sup>	740	F <sup>#</sup> , G <sup>b</sup>
196	G	784	G
208	G <sup>#</sup> , A <sup>b</sup>	831	G <sup>#</sup> , A <sup>b</sup>
220	A (below middle C)	880	A (above high C)
220	A (below middle C)	880	A (above high C)
233	A <sup>#</sup> , B <sup>b</sup>	932	A <sup>#</sup> , B <sup>b</sup>
247	B	988	B
262	C (middle C)	1047	C
277	C <sup>#</sup> , D <sup>b</sup>	1109	C <sup>#</sup> , D <sup>b</sup>
294	D	1175	D
311	D <sup>#</sup> , E <sup>b</sup>	1245	D <sup>#</sup> , E <sup>b</sup>
330	E	1319	E
349	F	1397	F
370	F <sup>#</sup> , G <sup>b</sup>	1480	F <sup>#</sup> , G <sup>b</sup>
392	G	1568	G
415	G <sup>#</sup> , A <sup>b</sup>	1661	G <sup>#</sup> , A <sup>b</sup>
440	A (above middle C)	1760	A



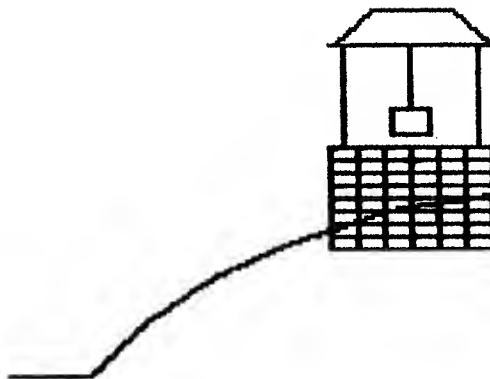
# Procedures for Jack and Jill

(Chapter 7)

TO J.AND.J  
CT TS  
QUESTIONS  
TELL [0 1 2 3] HT PU CS  
HILL  
WELL  
CHARACTERS  
FETCH  
JACK  
JILL  
END

TO QUESTIONS  
PR [NAME A BOY.]  
MAKE "BOY RL  
PR [NAME A GIRL.]  
MAKE "GIRL RL  
PR [NAME SOMETHING TO CLIMB ON.]  
MAKE "CLIMB RL  
PR [WHAT IS YOUR FAVORITE DRINK?]  
MAKE "DRINK RL  
PR [NAME A PART OF THE BODY.]  
MAKE "BODY RL  
PR [NAME A VERB ENDING IN ING.]  
MAKE "VERB RL  
END

TO HILL  
TELL 3 PU SETPOS [-70 -30]  
PD SETPC 0 96  
RT 90 FD 30 LT 45  
REPEAT 2 [FD 30 RT 10]  
FD 30 RT 5 FD 40  
RT 10 FD 40  
END



TO WELL  
TELL 2 PU  
SETPOS [50 20] PD  
SETPN 1 SETPC 1 16  
BASE  
TOP  
END

TO BASE  
REPEAT 8 [BRICKS LT 90 FD 50 RT 90 FD 5]  
END

TO BRICKS  
REPEAT 5 [REC RT 90 FD 10 LT 90] REC  
END

TO REC  
REPEAT 2 [FD 5 RT 90 FD 10 RT 90]  
END

TO TOP  
RT 90 FD 5 LT 90 FD 40  
RT 90 BK 5 FD 30 BUCKET FD 25  
RT 90 FD 40 BK 40 LT 90 FD 5  
LT 135 FD 20 LT 45 FD 30  
LT 45 FD 20  
END

TO BUCKET  
RT 90 FD 25 LT 90 FD 7  
RT 90 FD 10 RT 90 FD 14  
RT 90 FD 10 RT 90 FD 7  
LT 90 FD 25 RT 90  
END

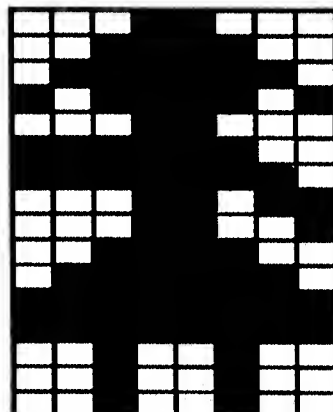
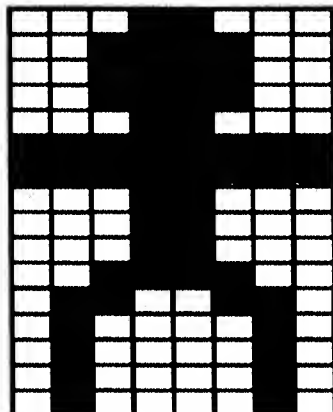
TO CHARACTERS  
MAKE "JACK GETSH 1"  
MAKE "JILL GETSH 2"  
PUTSH 1 :JACK TELL 0 SETSH 0  
PUTSH 2 :JILL TELL 1 SETSH 2  
SETC 40  
TELL [0 1] PU SETPOS [-40 -30] ST  
TELL 1 RT 90 FD 20  
TELL [0 1] SETH 45 SETSP 15  
END

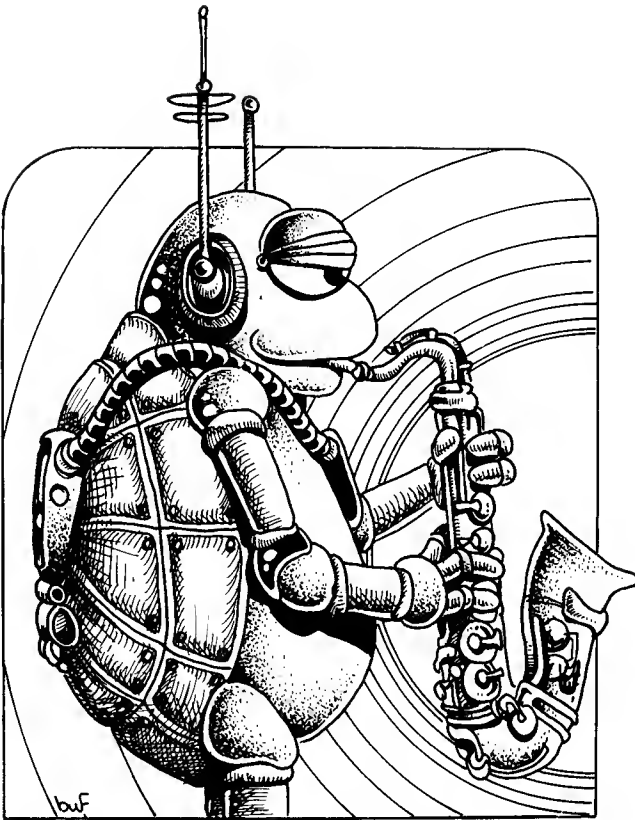
\*Remember that if you want to save the shapes you create, you must give each shape a name using the MAKE and GETSH commands.

TO FETCH  
 PR (SE :BOY [AND] :GIRL)  
 PR SE [WENT UP THE] :CLIMB  
 PR SE [TO FETCH A PAIL OF] :DRINK  
 HI.C 30 G 15 A 30  
 G 15 HI.C 30 G 15 A 30  
 G 15 HI.E 30 HI.D 15  
 HI.C 30 HI.B 15 A 45 G 45  
 END

TO JACK  
 CT  
 PR SE :BOY [FELL DOWN]  
 PR SE [AND BROKE HIS] :BODY  
 SETSP 0  
 TELL 0 SETH -135 SETSP 25  
 F 30 REST F 15 HI.D 45  
 F 15 E 30 REST E 15 HI.C 45  
 END

TO JILL  
 PR (SE [AND] :GIRL [CAME] :VERB [AFTER.])  
 SETSP 0 TELL 1 SETH -135 SETSP 25  
 E 15 D 30 A 15 G 30 B 15  
 D 45 C 60  
 SETSP 0  
 END





TO B :D  
TOOT 0 247 15 :D  
TOOT 1 123 10 :D  
END

TO C :D  
TOOT 0 262 15 :D  
TOOT 1 131 10 :D  
END

TO D :D  
TOOT 0 294 15 :D  
TOOT 1 147 10 :D  
END

TO E :D  
TOOT 0 330 15 :D  
TOOT 1 165 10 :D  
END

TO F :D  
TOOT 0 349 15 :D  
TOOT 1 175 10 :D  
END

TO G :D  
TOOT 0 392 15 :D  
TOOT 1 196 10 :D  
END

TO A :D  
TOOT 0 440 15 :D  
TOOT 1 220 10 :D  
END

TO H1.B :D  
TOOT 0 494 15 :D  
TOOT 1 247 10 :D  
END

TO H1.C :D  
TOOT 0 523 15 :D  
TOOT 1 262 10 :D  
END

TO H1.D :D  
TOOT 0 587 15 :D  
TOOT 1 294 10 :D  
END

TO H1.E :D  
TOOT 0 658 15 :D  
TOOT 1 330 10 :D  
END

TO REST  
TOOT 0 14 0 1  
END

# LOGO Resources

## Books

Abelson, Harold. Logo for the Apple II. New Hampshire: BYTE/McGraw-Hill, 1982.

Abelson, Harold. TI Logo. New York: McGraw-Hill Book Company, 1984.

Bearden, Donna; Martin, Kathleen; and Muller, Jim. The Turtle's Sourcebook. Reston: Reston Publishing Company, 1983.

Bearden, Donna. 1, 2, 3, My Computer and Me. Reston: Reston Publishing Company, 1983.

Bearden, Donna. A Bit of Logo Magic. Reston: Reston Publishing Company, 1984.

Martin, Kathleen; and Bearden, Donna. Primarily Logo. Reston: Reston Publishing Company, Inc., 1984.

Papert, Seymour. Mindstorms: Children, Computers and Powerful Ideas. New York: Basic Books, 1980.

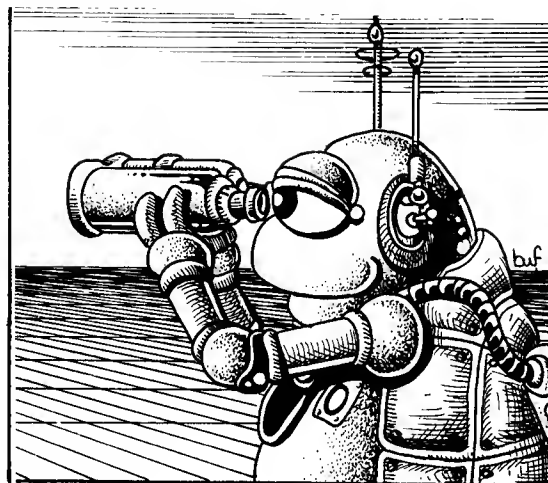
Thornburg, David D. Discovering Apple Logo, An Invitation to the Art and Pattern of Nature. Reading, Mass.: Addison-Wesley Publishing Company, 1983.

Watt, Daniel. Learning with LOGO. Highstown: BYTE/McGraw-Hill, 1983.

## Organizations

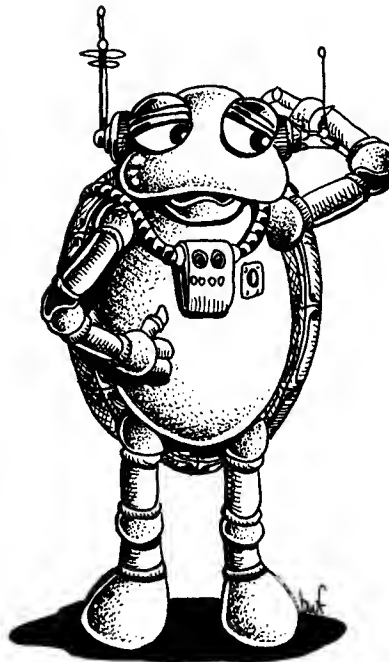
National LOGO Exchange, Box 5431, Charlottesville, VA 22905. Monthly Newsletter.

Young People's LOGO Association, 1208 Hillsdale, Richardson, TX 75081. Monthly Newsletter. Software Exchange.



# Index

Adventure Games	98	PN	34
BACK, BK	1	Polygon Procedure	47
Changing Turtle Shapes	78	Positions	28
Circles	41	Primitive	11
Collision Detection	73	PRINT, PR	35
Colors	33	PUTSH	80
Combining Shapes	49	Quizzes and Word Games	97
Computer Art Show	49	RANDOM	32, 90
CS	2	Rectangles	15
EACH	64	Recursion	59
Editing Procedures	12	RIGHT, RT	1
EDSH	78	Rule of 360	42
Embedded Recursion	65	SE	47
FORWARD, FD	1	SETBG	33
Frequency Values	87	SETC	33
GETSH	80	SETH	31
Headings	31	SETPC	34
HOME	1	SETPN	33
HT	7	SETPOS	30
IF	47	SETSH	79
Interactive Procedures	47, 91	SETSP	73
LEFT, LT	1	SETX	29
Mirror Images	6	SETY	29
Music	87	Squares	13
Musical Mad Lib	93	ST	1
Ovals	46	TELL	3
PC	34	Triangles	16
PE	2	TOOT	88
PENDOWN, PD	2	Variables	25
PENUP, PU	2	WHEN	74
Playing Turtle	2	WHO	74



After several years in the field of aging and a few more in health, **Donna Bearden** left a “real” job to fulfill a life-time dream of writing full time — almost. In addition to writing, she trains teachers and develops computer curriculum for school districts. **ATARI LOGO in the Classroom** is her fourth LOGO book published by Reston. She has also published a children’s book, **Monica, the Computer Mouse**, Sybex, 1984. (Illustrations by Brad W. Foster.) As of this writing, she is still healthy, aging, working hard and enjoying life. She lives with her husband, three children and two computers in Irving, Texas.



**Brad W. Foster** (or, as he is known in the truly trendy art circles of the world, “Who?”) tends to yield a pen with the same deft flourish most people show with blunt instruments, and usually with the same haphazard results. To get past this admittedly difficult obstacle in his path toward being an illustrator, he has come up with a remarkable solution. Basing his idea on the famous theory that “an infinite number of monkeys at an infinite number of typewriters will, given time, write the complete works of Shakespeare,” he attained a research grant from a large Eastern university, bought several million monkeys and supplied them with drawing tables, pens and paper. They are housed in a warehouse in West Texas. Brad has discovered that, indeed, the law of averages supplies him with enough recognizable artwork for him to sign and pass off as his own. He now plans to apply for a government grant to buy several million typewriters and get them started on his first novel. Quite clever for a college graduate! His motto is: “If it doesn’t move, draw on it.”



## More books for ATARI microcomputers . . .

**1,2,3 MY COMPUTER AND ME** (Muller). A fun book for Logo beginners, filled with activities and enticing drawings for third graders and up. Teaches turtle graphics commands and procedures for making shapes, writing, editing and saving, using color and more. A great introduction to Logo concepts.

**A+ PROGRAMMING IN ATARI BASIC** (Reisinger). Teaches BASIC programming to beginners. The emphasis on sound and graphics programs is meant to excite young learners and make teaching easier. Includes a section on *flowcharting*.

**ATARI PILOT ACTIVITIES AND GAMES** (Kohl, Kahn, Disharoon). PILOT, it's a simple language, but put enough of these little programs together and you've got a valuable tool for fun and learning. Includes math, logic, word, language, sound and graphics activities (PEEKs and POKEs, too).

Cut out this order form and mail it to:

V-0937-8F(2)

Customer Service  
Reston Publishing Company, Inc.  
11480 Sunset Hills Road  
Reston, VA 22090

Enclosed is my ☐ check or money order

☐ official school purchase order number \_\_\_\_\_

Please ship

\_\_\_\_\_ copies of 1,2,3 MY COMPUTER AND ME @ \$12.95 each  
Order number: 0-8359-5243-6/R5243-2, paperback \$ \_\_\_\_\_

\_\_\_\_\_ copies of A+ PROGRAMMING IN ATARI BASIC @ \$15.95 each  
Order number: 0-8359-0004-5/R0004-3, paperback \$ \_\_\_\_\_

\_\_\_\_\_ copies of ATARI PILOT ACTIVITIES AND GAMES @ \$14.95 each  
Order number: 0-8359-0321-4/R0321-1, paperback \$ \_\_\_\_\_

Local tax \$ \_\_\_\_\_

Total amount enclosed \$ \_\_\_\_\_

To \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_

Zip \_\_\_\_\_

Please allow 4-6 weeks for delivery.  
Reston Publishing will pay shipping  
and handling costs up to 5 books. All  
orders are subject to approval by the  
publisher. Prices are subject to  
change without notice.

☐ Please put me on your mailing list for a FREE Computers  
in education catalog.





# ATARI LOGO in the CLASSROOM

As a teacher or parent, you know how creative young minds can be, and how eager they are to learn new ideas. *ATARI® LOGO IN THE CLASSROOM* shows you how to capture this excitement and enthusiasm as you introduce children to logical thinking, problem solving, and mathematics with Logo.

*ATARI LOGO IN THE CLASSROOM* presents Logo in clear, easy-to-follow instructional sequences. Each key point of Logo is beautifully illustrated and accompanied by suggestions, questions, and short, self-directed projects. This guide lets students learn Logo at their own speed—but provides you with numerous tips and ideas to help them when they run into problems. Difficult commands are noted and illustrated with practical examples, and interactive exercises are presented to give children a hands-on understanding of fundamental math (without calling it math) and computer concepts. Worksheet space is provided.

Children will learn to create their own drawings and designs, compose music with two-part harmony, experiment with over 120 different colors, write stories, and design and save shapes to use with different programs. They'll discover that Logo is exciting, challenging, and *fun!* And so will you!

ATARI® is a registered trademark of Atari, Inc.  
Illustration: Churchmouse Graphics

A Reston Computer Group Book  
**Reston Publishing Company, Inc.**  
*A Prentice-Hall Company*  
Reston, Virginia