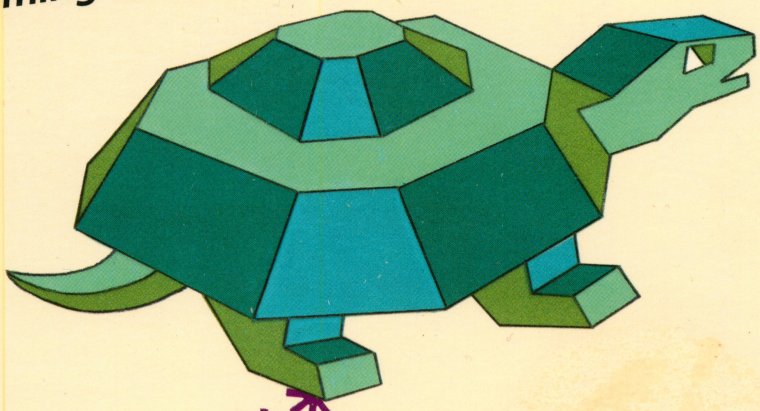
# TURTLE TRACKS™

## Learn Programming As You Create Colorful Computer Graphics





Scholastic
The Most Trusted Name in Learning

FUN SKILLS SERIES:
Computer Literacy

Ages 9 and up

Scholastic Wizware™ / Apple® II Plus 48K, DOS 3.3, Disk

# TURTLE TRACKS

*Learn to write programs by creating your own designs.*

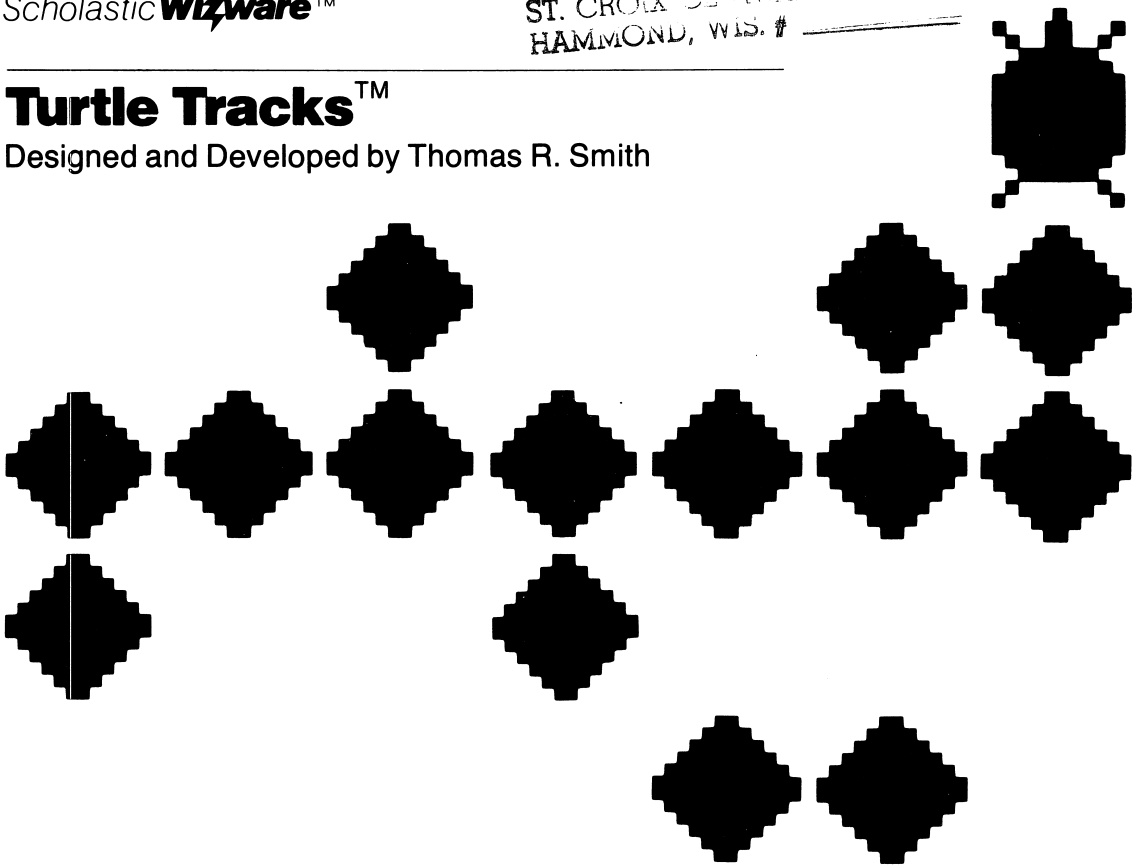


Ages 9 and up

Scholastic **Wizware**™

# Turtle Tracks™

Designed and Developed by Thomas R. Smith



**Scholastic Inc.** New York Toronto London Auckland Sydney Tokyo

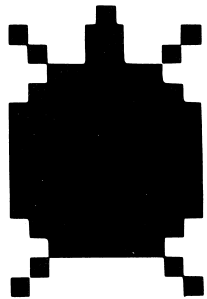
| | |
|---|---|
| **Designer, Developer, Programmer** | Thomas R. Smith |
| **Software Research** | This program was field-tested at Park School in Brookline, Massachusetts |
| **Publisher** | Scholastic *Wizware* ™ |
| **Creative Director** | Deborah Kovacs |
| **Project Manager** | Stephen Gass |
| **Software Editor** | Robert Neumann |
| **Handbook Authors** | Thomas R. Smith/Robert Neumann |
| **Art Editor** | Dennis Niswander |
| **Production** | Barbara Kellogg, Director<br>Maryellen Kohn, Editor |
| **Manufacturing** | Louis deRienzo, Manager<br>Deborah Honig, Assistant Manager |
| **Package and Graphic Design** | Robert P. Gersin Associates, Inc. |





ISBN: 0-590-95110-6 12 11 10 9 8 7 6 5 4 4 5 6 7 8/8

# Introduction



## Welcome to *Turtle Tracks*

You are about to team up with a talented and unusual creature — the turtle. This turtle lives inside your computer, and paints, draws, and sings with great ability.

The trouble is, the turtle doesn't know what to do with its talents — and that's where you come in. Your job: to provide the brains and the imagination that the turtle so sorely needs to put its talents to good use. You become the artist, the musician, and the architect. You use the turtle as your paintbrush, musical instrument, and set of building blocks.

Turn your imagination loose. Do you want to paint a tropical island sunset — mango trees under a red sun? Design it and use the turtle as your paintbrush. Do you want to compose a song? Write it and let the turtle play it.

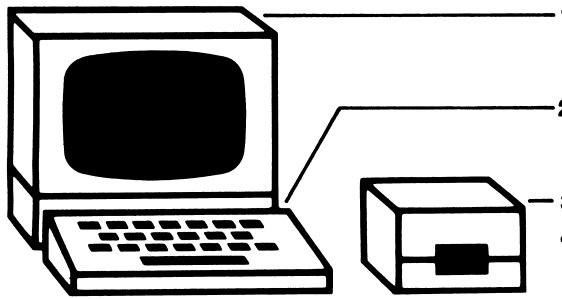
This handbook will teach you to talk to the turtle. And as you learn to do that, you'll also be picking up another valuable skill — computer programming. *Turtle Tracks* is actually a miniprogramming language. As such, it shares lots of features with Logo —another language that uses turtle graphics— and BASIC, the world's most popular language for microcomputers.

Have fun!

# Contents

# How to Load *Turtle Tracks* Into Your Computer

## Equipment You Need



1 TV or monitor (black and white will work but color is better)

2 48K Apple II®, II Plus®, or *IIe*®

3 Disk drive

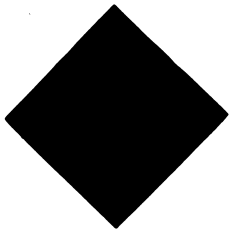
4 *Turtle Tracks* disk

Apple is a registered trademark of Apple Computer, Inc.

## Disk Loading Instructions

Your computer should be turned off.

1. Turn on your TV or monitor.
2. Insert your *Turtle Tracks* disk into the disk drive, with the label facing up and out. (If you have two disk drives, place the *Turtle Tracks* disk in drive 1.)
3. Turn on your computer. The disk drive whirs, and then a Scholastic *Wizware*™ screen appears, followed by a *Turtle Tracks* title screen. Then, you'll see TURTLE TRACKS in the upper-left corner of your screen. Welcome!
4. Before you start learning about the turtle, have a look at a sample program. Type LOAD DEMO. Press RETURN. When the computer asks which disk drive to use, type 1. The sample program will quickly load. You'll see some writing on the screen. Then you'll see > □. Type R and press RETURN. After the turtle runs through the program, press RETURN, type N, and press RETURN again.



# How to Use This Handbook

Work through the 10 easy-to-follow lessons in this handbook in order. After each lesson, play with the challenges we've suggested. They'll help you practice and apply what you have learned. Then, take as much time as you want to experiment with the turtle. That's really the only way to get comfortable with it. You'll learn as much from your mistakes as you will from your successes!

More information about *Turtle Tracks* is in the Appendices at the back of this handbook. In Appendix A (page 56), you'll learn how to save your hard work in *Turtle Tracks* on disk or paper. Read Appendix A when you're ready to save some of your work.

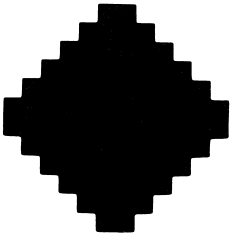
Appendix B (page 60) contains a list of all the instructions the turtle understands, and the lessons where you can find more information about those instructions. Consult Appendix B if you've forgotten how to use a certain instruction.

If the computer doesn't understand something you type, it will tell you so in a screen message. Correct your mistake and continue working. Appendix C (page 62) contains more information about common mistakes.

If you find yourself hopelessly lost in the middle of a lesson, type N, press the RETURN key, and start over at the beginning of the lesson.

And now — let's begin!

# Section I: Getting on Track



## Lesson 1: Talking to the Turtle

If you've loaded *Turtle Tracks* into your computer, you should see a small square at the top left of your screen. That's the *cursor.* The cursor shows where your typing will appear. Type your name. For example:

THOMAS SMITH↵

As you can see, the cursor moves along as you type.

### Erasing

On the right side of your keyboard you'll find a key that has an arrow pointing left. It looks like this: ←. Press this key. When you do, the cursor moves back one space and gobbles up the last letter of your name. You can now type a new letter in that space. Try it. Then keep pressing the ← key until the cursor erases all the letters in your name.

### Numbers and Letters

Typists sometimes use lower-case L (l) to represent the number one (1) and upper-case O to represent the number zero (0). Computers don't let you do that. You must use the number keys only for numbers and the letter keys only for letters.

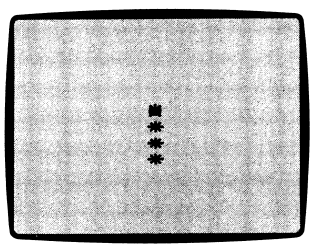
*NOTE:* Do not press the CTRL key on the Apple II and Apple II+, or the CONTROL key on the Apple //e. These keys can disrupt *Turtle Tracks.*

## Draw Forward

Let's start talking to the turtle. Type in:

1 DF3

You have just typed an instruction that tells the turtle to Draw Forward three steps. Tell the turtle that you have finished giving it an instruction by pressing the RETURN key. The cursor will move to the next line.



## Run

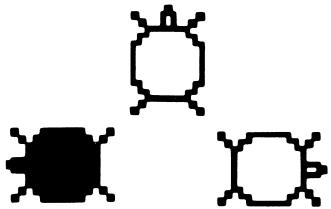
Now, type R, for Run, and press RETURN. This tells the turtle to carry out any instructions that you've given it so far. The turtle should Draw Forward three steps. It leaves behind a trail of three asterisks.

## Line Numbers

The number one (1) that begins instruction 1 DF3 is the instruction's line number. Line numbers determine the order in which instructions will be carried out.

## Two Screens

The screen you are looking at now is called the *turtle screen.* Press RETURN. The turtle disappears, and the cursor shows up again. This screen is called the *blackboard.* You'll be using the blackboard to give the turtle instructions.

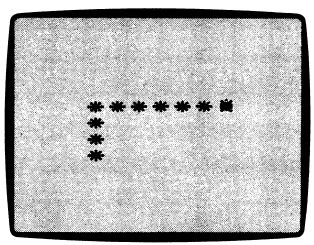


## Turns

Type the following three instructions carefully. If you make a mistake, correct it with the ← key. Press RETURN after each instruction. (If the turtle finds a mistake, it will send you a screen message. Compare what's on the screen with what's here in the handbook, and then type the line over correctly.)

Type:

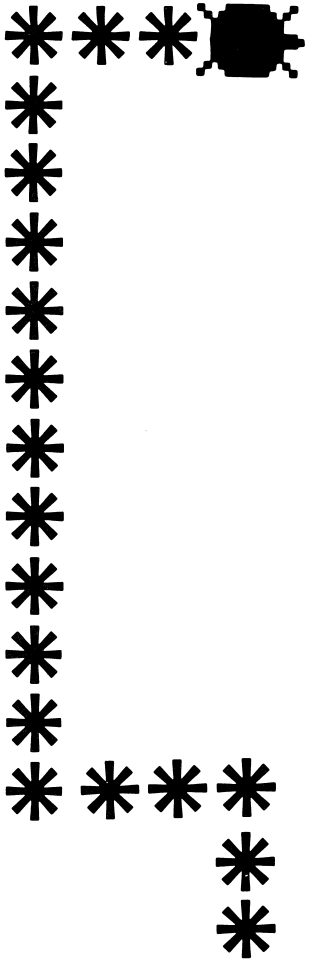
2 TR (Press RETURN.)
3 DF6 (Press RETURN.)
4 TL (Press RETURN.)



Type R and press RETURN. Here's what should happen. The turtle remembers line 1, DF3, and Draws Forward three steps. The instruction in line 2, TR, tells the turtle to Turn Right, and the turtle does so. The turtle then performs line 3, DF6, and Draws Forward six steps. Finally, the turtle obeys line 4, TL, and Turns Left.

## List

Return to the blackboard. (Press RETURN.) Now type L, for List, and press RETURN. The instructions you've given the turtle so far will appear neatly, in line number order.



## New

Now type N, for New, and press RETURN. N tells the turtle to erase old instructions, and prepare for new ones. If you type L now and press RETURN, you'll see that the turtle doesn't have any instructions to List. If you type R and press RETURN, you'll see that the turtle doesn't have any instructions to Run.

## Challenges

Now is the time to start experimenting with the turtle. Return to the blackboard. (Press RETURN.) Give the turtle some instructions to carry out. Remember to begin instructions with line numbers, and to finish them by pressing the RETURN key. When you want to run your instructions, type R and press RETURN. (The command R does not use a line number.)

If you get a screen message and you don't know why, turn to Appendix C (page 62).

1. Draw a square. (In *Turtle Tracks,* a square means a figure with the same number of asterisks — or other characters — on each side. If you used a ruler, you'd find that the lengths aren't exactly the same.)

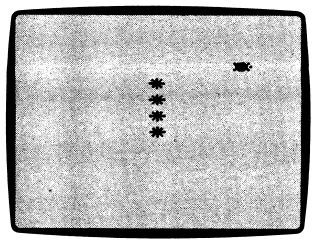
2. Draw a square with exactly five asterisks on each side. (You may be a bit surprised by the answer!)

3. Draw a rectangle whose length is seven and width is three asterisks. (Another surprise!)

4. Draw a capital letter of your choice.

(Suggested Solutions on page 63.)

# Lesson 2: Jumping Turtle



### Jump Forward

Return to the blackboard (press RETURN). Type N, press RETURN, then type:

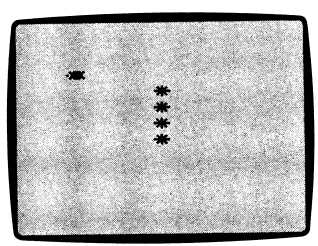
```
1 DF4
2 TR
3 JF5
```

Don't forget that you must press RETURN after each instruction, including the last one. Now type R and press RETURN. The JF instruction in line 3 tells the turtle to Jump Forward five steps. (Believe it or not, DF, JF, TR, and TL are the only instructions you'll need to move the turtle about the screen.)

### Changing Instructions

Return to the blackboard. Type L, press RETURN, and the following instructions should appear:

```
L
1 DF4
2 TR
3 JF5
```



Here's how you can change an instruction. Change line 2 so that the turtle Turns Left by typing:

```
2 TL
```

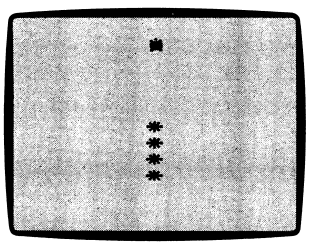
Press RETURN after you type in the instruction, of course. Then type R and press RETURN. The final screen should look like the one at right:

What happened to the old instruction in line 2 (TR)? The turtle replaced it. The turtle automatically throws out old instructions when it sees new ones with the same line number. Return to the blackboard, type L, and press RETURN. As you can see, line 2 now contains the new instruction.

**Erasing Instructions**

To erase an instruction, simply type the line number and press RETURN. For example, type:

2

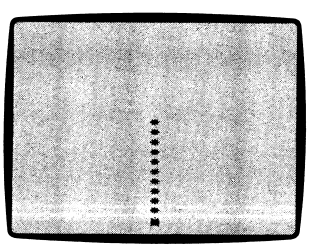


Press RETURN. The computer says DELETED, which means erased. Type L and press RETURN, and you'll see that line 2 has indeed vanished! Run the program (type R and press RETURN) and you'll see that the turtle doesn't care that line 2 no longer exists. The turtle goes straight from line 1 to line 3.

**Drawing Backward**

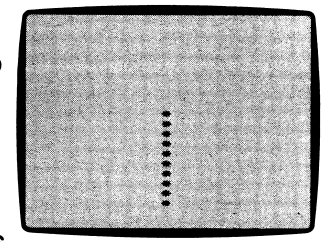
Use numbers with minus signs (−) in front of them to make the turtle draw backward. Return to the blackboard (press RETURN), type N, press RETURN again, then type:

1 DF − 10 (Press RETURN.)



Run this program (type R and press RETURN) and you'll see the turtle draw backward 10 steps.

## Jumping Backward



The turtle can jump backward as well as draw backward. Tell it to do so by using a minus sign before the number. Return to the blackboard, then type:

```
2 TR              (Press RETURN.)
3 JF – 8          (Press RETURN.)
```

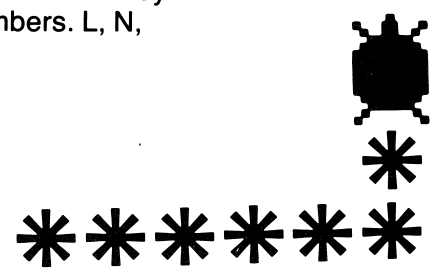
Run this program. The turtle carries out line 1 again and draws backward 10 steps. It turns right in line 2, then jumps backward eight steps in line 3.

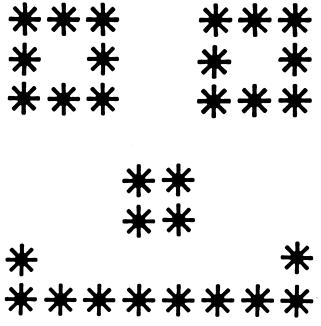
## Commands and Statements

You may be wondering why some instructions use line numbers and others don't. The answer is that there are two kinds of instructions.

*Statements* are parts of programs, and require line numbers. DF, JF, TR, and TL are statements.

*Commands* tell the turtle what to do with programs. Since they are not part of programs, they do not use line numbers. L, N, and R are commands.





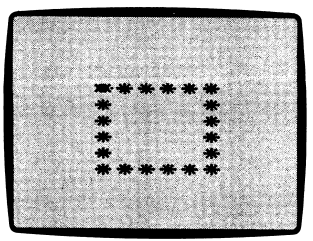
## Challenges

Practice with the JF (Jump Forward) statements, and use minus numbers to jump backward and draw backward. Try the following challenges.

1. Draw a square at the bottom of the screen.
2. Draw one square in the top-right corner of the screen, and another in the bottom-left corner.
3. Draw a slanted line.
4. Draw a capital N.
5. Draw a face. Make up your own, or try this one.

(Suggested Solutions on page 64.)

## Lesson 3: Round and Round in Loops



Return to the blackboard. Type N and press RETURN to clear the turtle's memory. Then type:

```
1 DF5
2 TR
3 GT1
```

(This tells the turtle to Go To line 1.)

Run this program (type R and press RETURN.) You'll see that the turtle draws a square.

Here's how this program works. The turtle carries out the instruction in line 1 and Draws Forward five steps. Then it carries out the instruction in line 2 and Turns Right. In line 3, the turtle is told to Go To line 1. The turtle does so. It carries out the instructions in lines 1 and 2 again, finds itself back at line 3, and is sent to line 1 again.

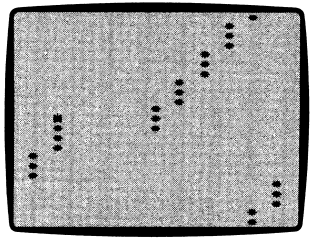
Look at the arrow that connects lines 1 and 3. Do you see why we call this a loop? The turtle loops, or cycles, through these instructions over and over again.

If you look at the screen now, you'll see that the turtle is still tracing the square. The turtle can't escape from this loop. Every time it reaches line 3, it must go right back to line 1 again. Because this type of loop has no end, we call it an *infinite loop.*

### Stopping the Turtle

Stop the turtle by pressing RETURN. Then type N, press RETURN, and type in another program with an infinite loop:

```
1 DF3
2 TR
3 JF2
4 TL
5 GT1
```



Run the program, and watch the turtle for a while. The turtle draws three asterisks, hops over to the right two steps, turns left, draws three asterisks, hops over again, and keeps repeating these steps.

You'll notice that the turtle sometimes wanders off one side of the screen. Watch carefully, and you'll see it reappear on the opposite side. The turtle treats the screen as though opposite sides curled around and actually touched each other.

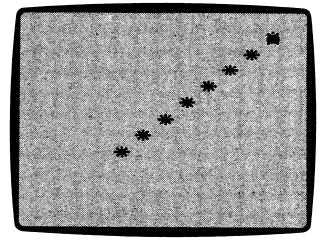
When you're tired of watching this program, return to the blackboard. (Press RETURN.)

## A New Loop

Let's look at a different type of loop. Type N, press RETURN, then type:

```
1 BL    <-+   (This tells the turtle to Begin Loop.)
2 TR      |
3 DF1     |
4 TL      |
5 JF1     |
6 RL7   --+   (This tells the turtle to Repeat Loop seven times.)
```

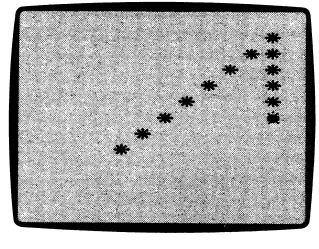
Run this program, and you'll see the turtle draw a slanted line. This type of loop begins with a BL statement (line 1) and ends with an RL statement (line 6).

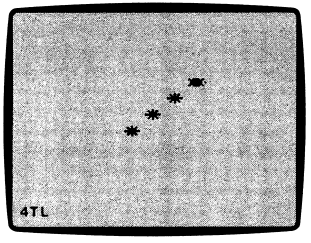


Unlike loops that use GT statements, this type of loop has an end. Here, the end comes after the turtle repeats the loop seven times. The turtle can then go on to new instructions. For example, return to the blackboard and type:

```
7 DF-5
```

Run the program, and you'll see that the turtle draws the same slanted line, and then draws backward five steps, as it is instructed to do in line 7.






## Walk

You can examine loops more closely with a new command. This command is W, for Walk. W tells the turtle to carry out its instructions one at a time. You control the turtle's pace with the SPACE BAR. Try it. Return to the blackboard, type W, and press RETURN. Each time you press the SPACE BAR, a new instruction appears on the screen. Press the SPACE BAR again, and the turtle carries out that instruction, and then a new one appears. Notice that when the turtle reaches line 6, it returns to line 2, the first real instruction in the loop. The turtle will perform the instructions in the loop seven times. Then it leaves the loop and goes on to the next instruction. That instruction, 7 DF-5, tells the turtle to draw backward five steps, and the turtle does so. You may leave a Walk by pressing W again.

## Walking and Running

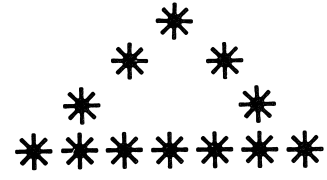
You may switch the turtle back and forth between walking and running while the turtle is carrying out instructions. For example, type W again, press RETURN, and use the SPACE BAR to Walk the turtle through a few instructions. Then press R. Don't press RETURN! The turtle switches to Run. Press W, and the turtle starts walking again. Press R again and the turtle starts running again . . . and so on.

## Debugging

The W command is an excellent debugging (correcting) tool. Even expert programmers find that their programs don't generally work correctly the first time around. They then try to *debug* the program, as though nasty, unwanted bugs were to blame for the problems. Of course, they don't find bugs — but they do eventually find their mistakes. Use the W command and you will, too.

## Challenges

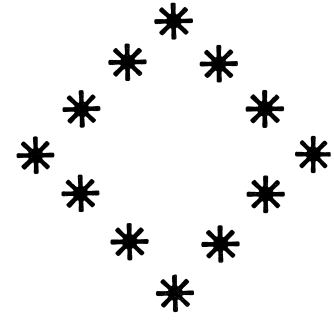
You may find it helpful to use graph paper to design your programs. First, figure out the size of the screen. Do this using DF or JF statements to determine when the turtle leaves each edge of the screen. Then, outline the screen on graph paper. Trace the patterns you want the turtle to draw, then give the turtle the proper instructions.

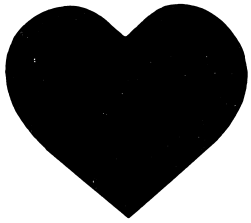


Try some of these challenges using loops:

1. Draw a square within a square.
2. Use loops to draw a triangle.
3. Use loops to draw a diamond.

(Suggested Solutions on page 65.)





# Section II: Picking Up Steam

## Lesson 4: Turtle Tidbits

### Line Numbers

Type N and press RETURN. Then type the following instructions.

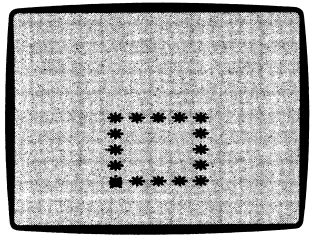
```
1 BL
2 DF4
3 RL4
```

Suppose you wanted the turtle to draw a square when you typed these instructions. But now you Run this program — go ahead and Run it — and you see that it doesn't draw a square.

What went wrong? You List your instructions and notice that you left out a TR or TL instruction between lines 2 and 3.

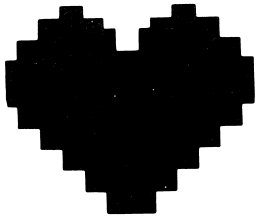
Luckily, you don't have to rewrite the whole program. The turtle understands line numbers that include decimals. So just type:

2.5 TR



Run the program and you'll see that the turtle builds a square. List the program, and you'll see that the new instruction has been inserted in its proper place, between statements 2 and 3. The turtle doesn't care when it receives an instruction. It only cares about line numbers.

Most programming languages don't allow decimal line numbers. So, programmers often number their lines by tens (10, 20, 30, 40, etc.) to leave plenty of room to insert new lines. *Turtle Tracks* allows you to do that, too.



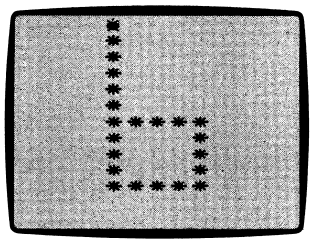
## End

The E statement tells the turtle that a program is over. E stands for End. When the turtle sees an E statement, it stops working, even if more statements follow. For example, add the following two lines to our program:

```
4 E
5 DF10
```

Run the program. As you can see, it looks the same as before. The turtle never reaches line 5.
Now, type:

```
4          (Press RETURN.)
```



Run the program now. Line 4 has been erased, so the turtle carries out line 5 and Draws Forward 10 after it builds a square.

You do not have to use E at the end of a program. So why use it at all? The E statement helps you test a part of a program. Insert it somewhere in a program, and the turtle will carry out all instructions up to that point, and then stop. The E statement, like the W command, can be a helpful debugging tool.

## More About Lists

You may List part of a program instead of the entire program. Here are some ways to do this:

```
L2         (List line 2 only.)
L9-        (List line 9 and all following lines.)
L-20       (List all lines up to and including line 20.)
L3.5-7     (List lines 3.5 through 7.)
```

Try different L commands with the program now in the turtle's memory. Use the commands above, and then try different numbers of your own choice.

### Clear

When your blackboard becomes cluttered with instructions, you may choose to erase it. You can do so by typing C, for Clear, and pressing RETURN. Try it. The C command does not affect the turtle's memory.

### Erasing Groups of Instructions

You may erase an entire group of instructions by simply typing in the lowest and highest line numbers of that group. When you do so, all line numbers in between will be erased. For example, type:

1-3

Press RETURN, and the computer says: DELETED. List your program, and you'll see that only line 5 is left from our program.

You can also erase a group of lines by typing the lowest number followed by a dash. This erases all line numbers higher than, and including, the line number. For example, type:

1-

Press RETURN, and the computer says: DELETED. List your program — and you'll see there's nothing left of it.

### Program Length

You may write programs that have as many as 150 statements. After that, the turtle runs out of memory.

## Loading and Saving

At this point, it's a good idea for you to learn about saving and loading your *Turtle Tracks* programs. Instructions are in Appendix A.

As an added bonus, you'll find some colorful *Turtle Tracks* programs are already on your *Turtle Tracks* disk. You can see a list of them if you type CATALOG, and then press RETURN. Type 1 when the computer asks USE WHICH DISK DRIVE?

After the catalog appears, you may want to load one of these programs into the turtle's memory, and then watch the turtle carry out the instructions. First type N and press RETURN. Then type LOAD followed by the program name. For example, LOAD FLAG. When the computer asks, tell it too use disk drive 1. Wait for the prompt (>) and the cursor to appear. The program is now in the turtle's memory. Treat it like any other program. First Run it—type R and press RETURN. Then if you'd like, List it—type L and press RETURN. You won't recognize many of these instructions yet. Don't worry—you will soon.

Load any other programs from the catalog that you'd like to see. Remember to type N first and press RETURN.

# Lesson 5: The Cast of Characters

Are you tired of watching the turtle draw black asterisks on a white background? If so, cheer up. The turtle knows how to draw many different characters in many different colors. For a quick look at the turtle's abilities, type H and press RETURN.

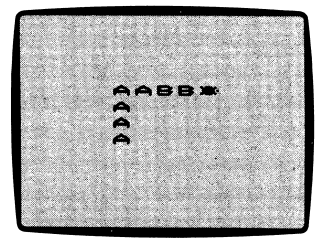
## Help Screens

After the disk drive whirs a bit, you'll see a screenful of characters. These are all the characters the turtle can print. (This chart is reproduced, in easy-to-read style, on page 27.) Press the SPACE BAR, and another screen shows you the color combinations the turtle uses. (This screen is reproduced on page 32.) The next time you press the SPACE BAR, you'll see a full listing of the turtle's vocabulary (the instructions it understands).

These help screens will be more useful to you once you are more familiar with the turtle's language. Then, you can use a help screen to remember a color number, or remind yourself of the proper form of an instruction.

You can call the help screens anytime by typing H and pressing RETURN. You can leave the help screens by pressing any key except RETURN or SPACE BAR.

## Different Characters



Return to the blackboard. Type N, press RETURN, and type:

```
 5 PA
10 DF3
20 TR
30 DF2
40 PB
50 DF2
```

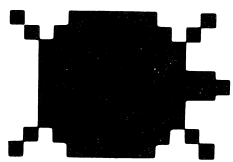
Run the program. The turtle draws with A's, then B's.

Statements 5 and 40 tell the turtle which characters to print. The turtle prints the character following the P. In statement 5, that's an A, so the turtle will print A's when you give it Draw Forward (DF) statements. In line 10, the turtle draws three A's.

In line 30, the turtle draws two more A's. The turtle continues to print A's until you tell it to switch.

Line 40 tells the turtle to switch to printing B's. In line 50, the turtle does so.

Using P statements, the turtle can draw any of the numbers, letters, and symbols that appear on your Apple keyboard. (To print the character on the top of a key, hold down the SHIFT while you press the key.)



## The P# Statement

Return to the blackboard. Change lines 5 and 40 by typing:

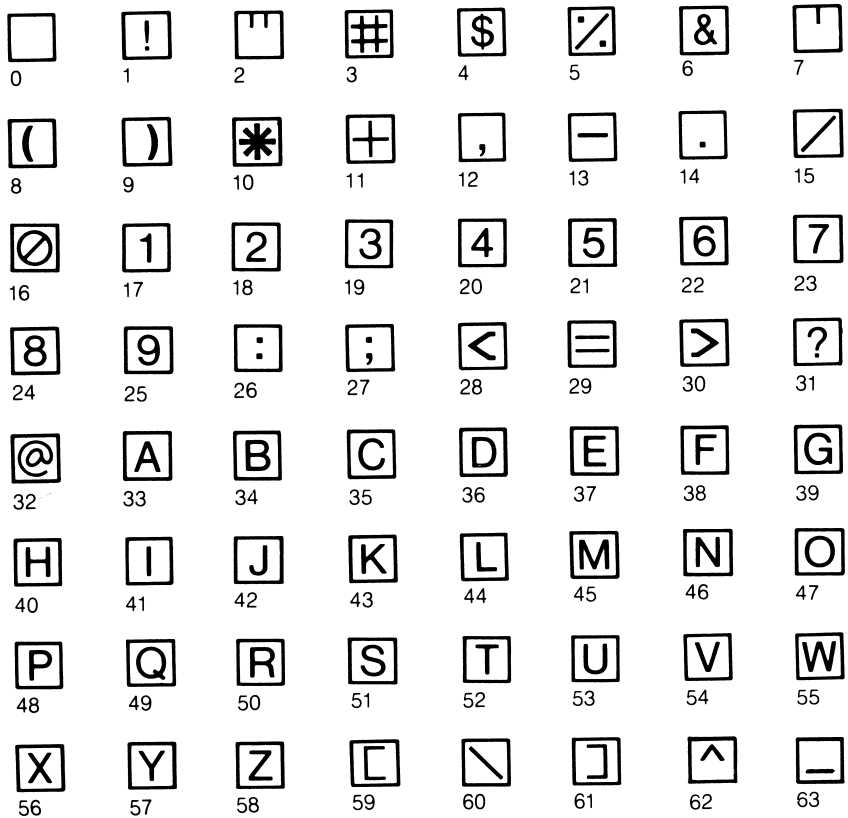
```
 5 P#33
40 P#34
```

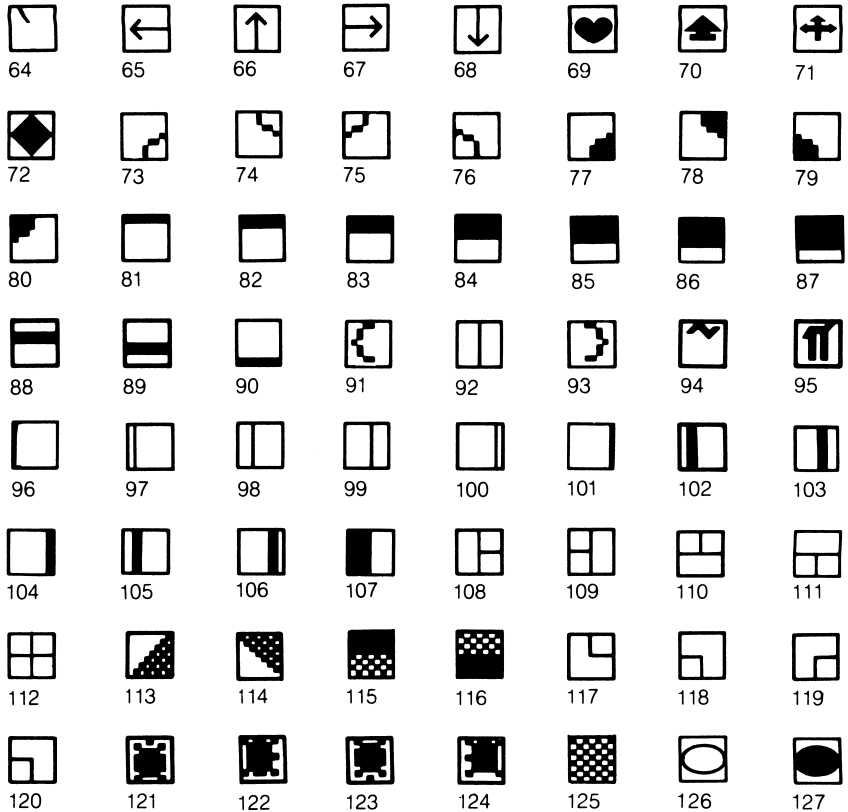
Run this program. It looks familiar, doesn't it? It's the same as the last program, even though the P statements look different. The key here is the # symbol after the P. This symbol means *number.* Now look at the chart on the next page. As you can see, different P# codes stand for different characters. Character #33, for example, is an A, and character #34 is a B. That explains why our new lines 5 and 40 have the same effect as the old ones.

You can also find a character's P# by referring to the first of the help screens. Type H and press RETURN. The disk whirs for a moment, and then the character screen appears.

Here's how you can determine a character's P# from the screen. Find the B. Take the number of the row the B is in — move your finger to the left, and you'll find that's number 3. Then take the number of the column — move your finger up, and you'll find that's number 4. Place the 4 after the 3 — and you should come up with 34. The B is P#34.

# P # Codes

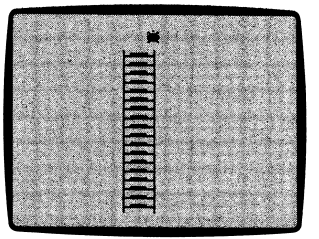

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63


64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127

## Variety

Use the letters, numbers, symbols, and shapes in the character chart, and you'll be able to create more interesting patterns than you could create with asterisks alone. For example, here's a program that builds a simple ladder:

```
 10 JF-10
 20 P#108
 30 DF18
 40 TR
 50 JF1
 60 TR
 70 JF1
 80 P#88
 90 DF18
100 TL
110 JF1
120 TL
130 JF1
140 P#109
150 DF18
```



## Reminders

You may leave yourself reminders inside programs. These reminders will be ignored by the turtle. A reminder begins with an apostrophe (') after the line number.

Here's an example of how you might use a reminder in the above program. Type:

81'#88 line (The apostrophe (') is on the 7 key. Use the SHIFT.)

This reminds you that the character in line 80 draws a line. If you now Run your program, you'll find that the turtle does the same things it did before. List your program, and you'll find that your reminder is right where it should be. The apostrophe tells the turtle to ignore it. A reminder may be nine characters long, including the apostrophe and spaces.

## Challenges

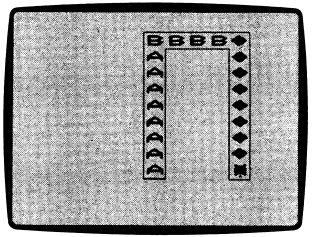
1. Use P statements to tell the turtle to print your name.
2. Use P# statements to tell the turtle to print your name.
3. Draw a small square from four graphics characters.
4. Draw railroad tracks.

(Suggested Solutions on page 66.)

## Lesson 6: Sound and Color!

The P and P# statements you've been using can also tell the turtle to draw in color. The colors the turtle may use are displayed on the second of the three help screens. To see them, type H and press RETURN. When the character screen appears, press the SPACE BAR again.

Notice that 22 combinations of colors are available. Each combination consists of the character color and a background color. Let's see this in action. Type the following program:



```
10 PA;6
20 DF8
30 TR
40 PB
50 DF4
60 TR
70 P#72;14
80 DF8
```

Run this program. The turtle draws eight green A's and four green B's against a black background, and eight black diamonds against a purple background.

Look at statement 10, PA;6. The first part of the statement, PA, looks familiar. It tells the turtle to print A's. The second part of the statement is the number 6 following the semicolon (;). This number tells the turtle which colors to use. These color combinations are listed on the help screen, and again on the next page.

# Color Chart

**(For screen color and character color)**

| Color Number | Color |
|---|---|
| 0 | Black on white |
| 1 | Green on white |
| 2 | Purple on white |
| 3 | Orange on white |
| 4 | Blue on white |
| 5 | White on black |
| 6 | Green on black |
| 7 | Purple on black |
| 8 | Orange on black |
| 9 | Blue on black |
| 10 | White on green |
| 11 | Black on green |
| 12 | Purple on green |
| 13 | White on purple |
| 14 | Black on purple |
| 15 | Green on purple |
| 16 | White on orange |
| 17 | Black on orange |
| 18 | Blue on orange |
| 19 | White on blue |
| 20 | Black on blue |
| 21 | Orange on blue |

Once you tell the turtle to print a certain color, it continues to use it until you tell it to switch. No color number is given in line 40, so the turtle prints B's in the same colors. In line 70, the turtle is told to switch to color 14 — purple letters on a black background — and to print diamonds (P#72). Notice that P statements and P# statements treat color numbers the same way.

### Changing Screen Colors

The following program will change the entire screen to orange in a matter of seconds. Type:

```
10 P#0;16
20 BL
30 DF24
40 TR
50 JF1
60 TL
70 RL20
```

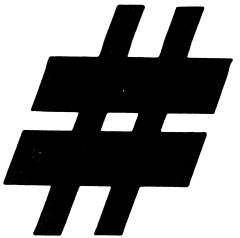
Run the program. Notice that statement 10 tells the turtle to use character #0. For colors 0-4, this character is a blank. For colors 5-21, it appears as a solid block of color.

### Beeps

B statements bring sound and music to *Turtle Tracks.* Type N, press RETURN, and then type:

```
10 B
```

Run this instruction, and you'll hear the turtle beep.



## Pitch

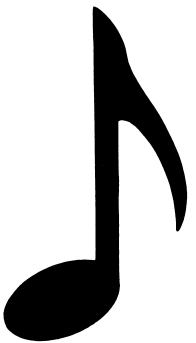
You can tell the turtle to beep at a certain pitch. For example, type:

```
20 B#18
```

Run this program. You'll hear that the second pitch is lower than the first. The second pitch, B#18, is a C note in Octave 1. You can see that on the chart on the next page. The turtle can play notes in a range of more than three octaves — more than the human voice can sing. The chart shows you the notes in this range.

You can use other numbers that aren't on the chart. You'll hear an interesting assortment of sounds.

## Duration



You can tell the turtle how long to hold notes. Type:

```
20 B#20;200
```

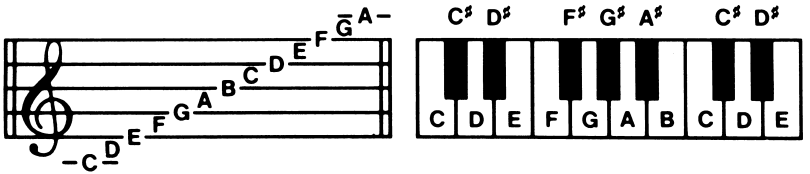
Run this instruction. You'll find that the turtle holds this note for a longer period of time than it held the last note. The number following the semicolon (;) controls the length of time a note is held. You may choose any number between 1 (shortest) and 256 (longest). Actually, you may use other numbers, too. For example, 257 will match back to 1, 258 will match back to 2, and so on.

If you don't use a number, the turtle selects 15. The following two instructions therefore mean the same thing:

```
5 B#10;15          5 B#10
```

# *Turtle Tracks*
# Musical Note Values

| Octave 0 | | Octave 1 | | Octave 2 | | Octave 3 | |
|---|---|---|---|---|---|---|---|
| Note | B# Value | Note | B# Value | Note | B# Value | Note | B# Value |
| G | 1 | G | 13 | G | 25 | G | 37 |
| G# | 2 | G# | 14 | G# | 26 | G# | 38 |
| A | 3 | A | 15 | A | 27 | A | 39 |
| A# | 4 | A# | 16 | A# | 28 | A# | 40 |
| B | 5 | B | 17 | B | 29 | B | 41 |
| C | 6 | C | 18 | C | 30 | C | 42 |
| C# | 7 | C# | 19 | C# | 31 | C# | 43 |
| D | 8 | D | 20 | D | 32 | D | 44 |
| D# | 9 | D# | 21 | D# | 33 | D# | 45 |
| E | 10 | E | 22 | E | 34 | E | 46 |
| F | 11 | F | 23 | F | 35 | F | 47 |
| F# | 12 | F# | 24 | F# | 36 | F# | 48 |



## Music

You can combine B# statements into melodies. Make up your own, or teach your favorite songs to the turtle.

In some cases, you may want to spread out the time between two notes. To do that, you can just put in some statements that won't affect the turtle. For example, type:

```
13 BL
15 RL5
```

Run this program and you'll see that the two new statements serve as a rest.

## Challenges

1. Play the notes in a G scale in Octave 2: G, A, B, C, D, E, F#, G.

2. Play the notes in a C scale starting in Octave 1: C, D, E, F, G, A, B, C.

3. Color one half of the screen green and one half orange.

4. Repeat some earlier challenges, and add color to them.

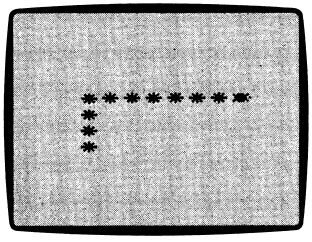
(Suggested Solutions on page 67.)

## Lesson 7: Variables

Return to the blackboard, type N, and press RETURN. Then type the following short program.

```
 5 X=3
10 DFX
```

Run the program. The turtle should Draw Forward three steps. Why would it do that? After all, there aren't any DF3 statements in this program.



Here's what happened. Line 5 told the turtle that X is 3. So, the turtle replaced the X in line 10 with a 3. Now type:

```
15 TR
20 X=7
30 DFX
```

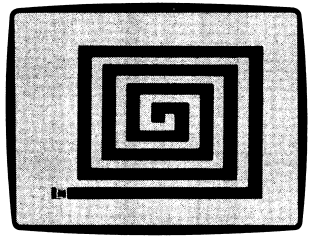
Run the instructions. As before, the turtle Draws Forward three steps. Then it Turns Right and Draws Forward seven steps. Why? Because line 20 told the turtle that X is now 7. So the turtle translated line 30 to read DF7.

This program used DFX twice, and each time it meant something different. That's because the value of X was changed each time. Because you can vary — or change — the value of X throughout a program, X is called a *variable.*

You can use variables to create interesting shapes. Type N, press RETURN, and then type:

```
 5 X=1
10 P#0;19
20 BL
30 DFX
40 TR
50 X=X+1
60 RL16
```

(Tells the turtle to increase X by 1.)



The turtle should create a square spiral shape. Walk through the program. Notice that in line 5, before the loop begins, X is *initialized.* That means it is given an initial, or starting value. In this case, line 5 said that X is 1.

Now look at line 50. It tells the turtle to increase X by 1 each time the turtle passes through the loop.

Try writing this program without variables, and you'll see just how helpful they are.

## Variable Beeps

You may raise the value of variables by numbers other than 1. Here is a program that treats you to a sampler of the sounds the turtle makes. Type N, press RETURN, then type:

```
 5 X=6
10 BL
20 B#X;100      (X controls the pitch of the beep.)
30 X=X+3        (Tells the turtle to increase X by 3.)
40 RL15
```

Run this program.

## Variable Characters and Colors

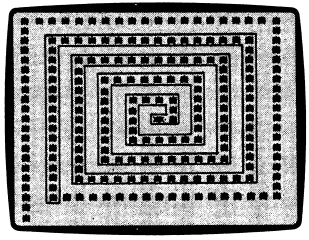
You can use the same variable in several parts of a program. For example, type:

```
 5 X=1
10 BL
20 P#X;X        (X controls the character and its color.)
30 B#X;X        (X controls the length and pitch of the beep.)
40 DFX
50 TR
60 X=X+1        (Increase X by 1.)
70 RL128
```

Run this program and you'll see all the characters and colors the turtle uses, and you'll hear many of the turtle's sounds.

## A Second Variable

*Turtle Tracks* allows you to use two different variables in a program. Variables must be named X or Y. For example, type:



```
  5 JF-10
 10 TR
 20 JF-9
 30 TL
 40 X=20
 50 Y=1
 60 BL
 70 P#121;Y
 80 DFX
 90 TR
100 X=X-1
110 Y=Y+1
120 RL21
```

(Y controls color.)
(X controls the number of steps the turtle draws.)

Run this program. There are two things to learn from it. First, notice in line 100 that you can decrease a variable as well as increase it. Second, notice that we could not have done this program with just one variable. X and Y have different starting values, and X decreases while Y increases.

You may use variables instead of numbers wherever you'd like to in programs. You may use all of the following statements: DFX, JFX, RLX, and even GTX (or DFY, JFY, etc.).

Of course, a line number may not be a variable. The program following at left is okay, but the one at right is not:

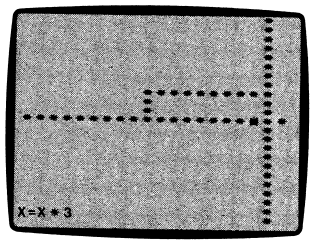
| Okay | Not Okay |
|---|---|
| 5 X=30 | 5 GTX |
| 10 GTX | 10 JF5 |
| 20 JF5 | X DF3 |
| 30 DF3 | |

**Multiplying Variables**

When you multiply variables, they can grow large very fast.

For example, type N, press RETURN, then type:

```
10 X=3
20 BL
30 DFX
40 TR
50 X = X*3
60 RL10
```

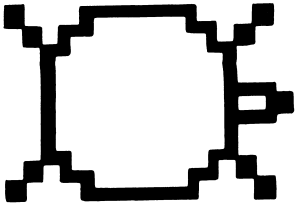



Walk the turtle through this loop a few times. X grows large very quickly, and the turtle draws for long periods at a time. If you press a key to stop the turtle, it will finish whatever statement it's doing before it stops. That can take some time.

The highest number the turtle will accept is 32,767. After that, you'll get a screen message telling you that the number is too large.

You may change the value of a variable in a number of ways. The following are some examples:

```
Y=5  X=-4  X=X+5  Y=Y-7  X=X+5
Y=Y+X  X=Y-X  X=Y*3  Y=Y*5
```



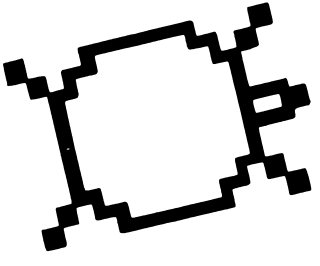
**Challenges**

1. Use a variable to print the first 10 letters of the alphabet across the top row of your screen.

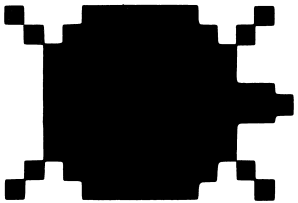
2. Use a variable to print TURTLE four times, each in a different color.

3. Use a variable to fill the screen with diamonds using all the turtle's colors.

(Suggested Solutions on page 68.)



# Section III: Full Speed Ahead



## Lesson 8: Teaching the Turtle New Words

### Miniprograms

You can tell the turtle to remember a group of instructions under a certain name. These groups of instructions are called miniprograms. Anytime you use the miniprogram name, the turtle carries out the entire group of instructions. For example, create a group of instructions to be remembered under the name SQUARE. Type N, press RETURN, then type:

TO SQUARE

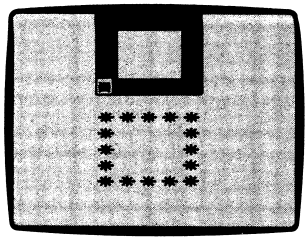
Press RETURN, and the turtle responds:

INVENTING WORD SQUARE

Type these instructions. Remember to use line numbers, as always:

```
10 BL
20 DF4
30 TR
40 RL4
```

Check that the turtle does what you intended. Type R SQUARE and press RETURN. The turtle runs through the miniprogram and draws a square. If you made a mistake in typing, type L SQUARE, press RETURN, then make any necessary changes or additions. Then type R SQUARE again and press RETURN to make sure the miniprogram works correctly.



## Using Miniprograms as Instructions

The following short program uses SQUARE as an instruction. Type:

```
 5 SQUARE
10 JF8
15 P#0;11
20 SQUARE
```

Run this program. Line 5 directs the turtle to carry out the instructions it has remembered under the name SQUARE. The turtle then goes on to line 10, where it Jumps Forward eight steps. In line 15, the turtle is told to print character #0 in color #11. So, when the turtle is told to carry out instructions for SQUARE in line 20, it does so using green blocks.

## Changing Miniprograms

Right now, the miniprogram SQUARE only makes squares that have five characters on a side. Let's make SQUARE more valuable by giving it the ability to draw squares of different sizes. We do this by using a variable in the miniprogram SQUARE.

Tell the turtle you want to work on instructions for SQUARE by typing L SQUARE and pressing RETURN. The turtle lists instructions for SQUARE, and then lets you make additions or changes. Type:

```
20 DFX      (Press RETURN.)
```

There are several ways to tell the turtle that you are finished working on a miniprogram. One is to type D. Do so, and then press RETURN.

### Directory

D stands for Directory, a list of all the miniprograms the turtle knows. All you'll see, of course, is SQUARE.

### Changing the Main Program

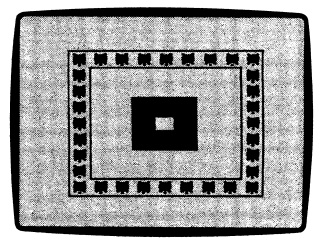
After you type D, instructions you type appear in the main program. Let's take advantage of our new flexibility in SQUARE and write a new main program.

First, we have to erase our old main program. But don't type N, because that erases miniprograms too! Instead, just type:

5-20 (Press RETURN.)

Now type in this program:

```
 10 P#0;11
 20 X=2
 30 SQUARE
 40 JF-3
 50 TL
 60 JF3
 70 TR
 80 P#123;21
 90 X=8
100 SQUARE
```



Run this program. Notice how valuable SQUARE has become! When you supply the proper instructions in the main program, the turtle can make a square of any size, color, and character.

## Miniprograms Within Miniprograms

You may use miniprograms to create new miniprograms. For example, here's a miniprogram that makes three identical squares. Call it TRIPLE. Type TO TRIPLE, press RETURN, and then type in these instructions:

```
 5 P#69;3
10 X=2      (This determines the size of SQUARE.)
20 BL
30 SQUARE
40 JF4
50 RL3
```

Run this program (type R TRIPLE and press RETURN). Type D, and you'll see that TRIPLE is now with SQUARE in the Directory.

## Rules for Using Miniprograms

These six rules will help you steer through the world of miniprograms.

1. Use the same commands for miniprograms that you use with main programs. However, you must add the miniprogram name. For example, R BOX, W SQUARE, L HOUSE.

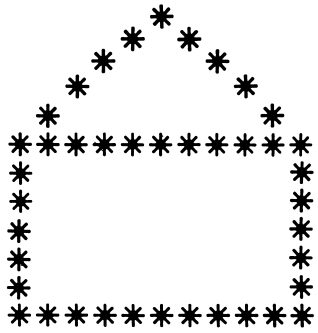
2. To create a miniprogram, first type TO and then the name. For example, TO SQUARE.

3. If you have stopped working on a miniprogram and you later decide to change it, first type L and the miniprogram name. For example, L BOX.

4. If you are working on a miniprogram and decide to erase it, type the lowest and the highest line numbers. For example, 10-50. Then, type D.

5. After you Run or Walk a miniprogram, the turtle thinks you are finished working on it. To make changes or additions to that miniprogram, first type L and then the miniprogram name. For example, L SQUARE.

6. To stop working on a miniprogram, type D or L.

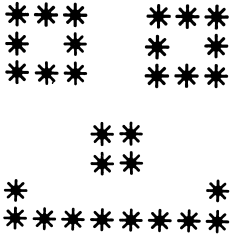


**Challenges**

Use miniprograms to solve these challenges:

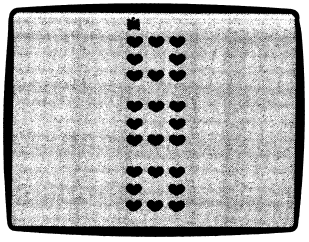
1. Draw a house:

2. Draw a face:



These examples use asterisks to keep things simple. Feel free, of course, to draw with any characters you choose.

(Suggested Solutions on page 69.)



## Lesson 9: Looping the Loops

Type N, press RETURN, and then type the following instructions:

```
 1 P#69;3
 2 BL          (Tells the turtle to Begin Loop.)
 5 BLA         (Tells the turtle to Begin Loop A.)
10 DF2
20 TR
30 RLA4        (Tells the turtle to Repeat Loop A 4 times.)
40 JF4
50 RL3         (Tells the turtle to Repeat Loop 3 times.)
```

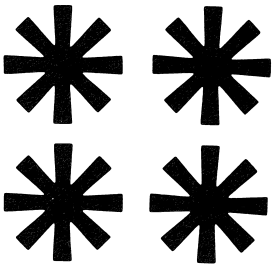
Run this program. The turtle should draw three squares made of orange hearts just like miniprogram TRIPLE in the last lesson. Instead of using miniprograms, however, it uses *nested loops* — loops that are arranged one inside another.

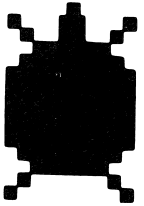
Look more closely at the program by walking through it. Type W, press RETURN, and control the turtle with the SPACE BAR. Notice that the turtle performs the inner loop, lines 5 through 30, four times before continuing to line 40. This inner loop draws a square. The instructions RLA and BLA are used for this loop.

The outer loop, lines 2 through 50, controls the distance between the squares (line 40) and the number of squares the turtle draws (line 50). Instructions RL and BL are used for this loop.

You may find it easier to understand this program if you see that the inner loop can be replaced with a miniprogram SQUARE. The program would then look like this:

```
TO SQUARE
10 BL          1 P#69;3
20 DF2         2 BL
30 TR          3 SQUARE
40 RL4         4 JF4
               5 RL3
```





**Variables and Nested Loops**

Here's a program that uses variables and nested loops together. Type N, press RETURN, and type:

```
 5 BL
10 DF6
20 X=10
30 BLA
40 B#X;100
50 X=X+10
60 RLA5
70 TR
80 RL4
```

The turtle draws a square, and pauses to play five notes after drawing each side. You could also have typed:

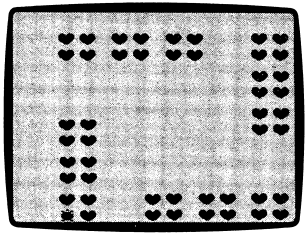
```
TO TUNE
 5 BL
10 B#X;100
20 X=X+10
30 RL5
```

```
 5 BL
10 DF6
20 X=10
30 TUNE
40 TR
50 RL4
```

**A Third Loop**

In *Turtle Tracks,* you may nest loops three deep. That means you may have a loop within a loop that's within another loop. Use instructions RLB and BLB for the third loop.

Here's an example.



```
 10 JF-4
 20 TR
 30 JF-5
 40 TL
 50 P#69;3
 60 BLB
 70 BLA
 80 BL
 90 DF1
100 TR
110 RL4
120 JF3
130 RLA3
140 JF5
150 TR
160 RLB4
```

See screen.

Run this program, and then Walk through it. You may want to think about miniprograms as you do so. Miniprogram TRIPLE on page 46 could be used in place of lines 70 through 130.

In *Turtle Tracks,* any loop may use instructions RLB and BLB, RL and BL, or RLA and BLA. The loop name ("A" or "B" for example) does not determine whether a loop is an inner loop or an outer loop. That's determined strictly by where you place your loops.

More than any part of *Turtle Tracks,* you'll need lots of practice to get used to nesting loops. Expert programmers say it is one of the toughest skills to develop.

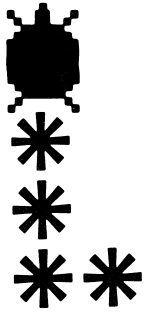
Make sure you understand the examples given in this chapter. Then, when you're ready, tackle some of these challenges.

**Challenges**

Use nested loops in the following challenges.

1. Draw four squares that have the same center.
2. Draw a diamond.
3. Draw two diamonds next to each other. (A tough challenge.)
4. Draw three diamonds, one inside the other. (A super challenger.)

(Suggested Solutions on page 70.)



# Lesson 10: Microturtle

Type N, press RETURN, then type:

10 M#2

Run this instruction, and a square appears. In the bottom-left corner of the square, you'll find the turtle.

Welcome to Micromode — and say hello to Microturtle!

## New Characters

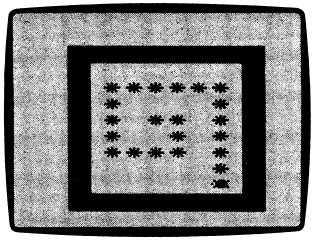
Microturtle helps you create your own characters. Here's an example of how this is done.

First, return to the blackboard and look at line 10, M#2. This statement does two things. It tells the turtle to become Microturtle and to enter Micromode. And it also tells the turtle to design a brand-new P#2 character.

Hit any key to return to the blackboard. Then type these instructions:

```
 20 JF4
 30 TR
 40 JF3
 50 X=1
 60 BL
 70 DFX
 80 TR
 90 X=X+1
100 RL6
```

Now Run these instructions. Microturtle appears, and builds a small spiral in Micromode. When it finishes, return to the blackboard and type one last instruction:



110 EM (Tells the turtle to End Micromode.)

Run the instructions once more. Notice that statement 110 brings Micromode to an end. Now type H, and you'll see that P#2 is now your spiral!

**Using Micromode Characters**

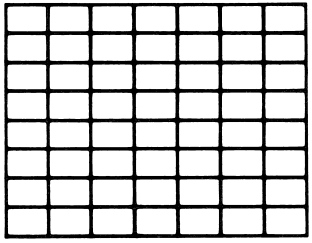
Now that you've created a new P#2, you may use it the same way you'd use any other character. Your new character, P#2, will remain in the computer until you leave *Turtle Tracks.* You can type N, and it will still be there. For example, type N, press RETURN, then type:

```
 5 TR
10 P#2
20 DF10
```

Run these instructions, and you'll see that the turtle draws 10 spirals.

## Designing Your Own Characters

When you design your own characters, you must first decide which of the standard P# characters you want to replace. Choose one that you don't expect to use in any of your programs. (You can only have 128 characters in *Turtle Tracks,* P#0-P#127. If you try and create M#128, you'll find that it takes the place of P#0. M#129 would take the place of P#1, etc.)



Then, draw a square on paper — graph paper is best. Divide the square so that it is seven boxes wide and eight boxes high. These are the dimensions of the Micromode square.

Map out your character by darkening some boxes, and leaving the others as they are. Then, type the *Turtle Tracks* program that tells Microturtle how to draw this character. Use normal *Turtle Tracks* instructions, and tell Microturtle to leave asterisks in boxes you have darkened. Remember that the first line of your program must tell the turtle to enter Micromode, and which character to replace — for example, M#11 would replace the plus sign.

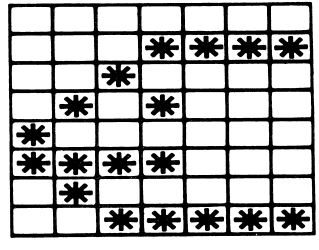
When you finish typing in your program, Run it. If your character doesn't look right, make the necessary changes. Then, add one last line to your program — EM (with a line number, of course.)

Run the program. If you type H, you'll now find that your character is firmly planted in the turtle's memory under the proper P#.

## Saving Your Characters

Once you create a new character, you may continue using it until you leave *Turtle Tracks*. Next time you load *Turtle Tracks*, however, you'll find that the standard P# character has returned, and your character has disappeared.

So, if you create a character that you may want to use again some day, save the program for that character on a disk. Instructions for saving and loading programs are in Appendix A, immediately following.

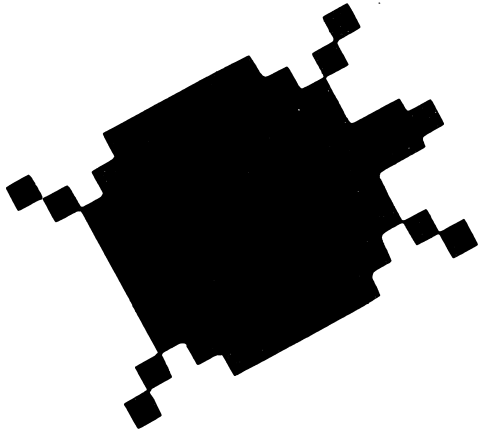


## Challenges

1. Write the program that creates the character at right. In case you can't tell, it's a close-up of the turtle's face.

2. Design more characters. Try designing lower-case letters, for example.

(Suggested Solutions on page 71.)



# Appendix A

## Saving and Loading Your Work

When you type N and press RETURN, or when you turn off the computer, the program in the turtle's memory is erased. That means you must retype all the instructions if you later want to continue working on that program. That's no problem if the program is short. If it's long, however, you won't relish the thought of typing in all those instructions again.

Cheer up — there's a better way to do it. You can use the Apple disk drive to load your instructions into the computer in a fraction of the time it would take you to type them. To do that, of course, you must first save your programs onto a disk before you erase them from the turtle's memory. In this Appendix, you'll find instructions for using disk drives and printers too.

### Preparing a Storage Disk

If you have only one disk drive hooked up to your Apple, you must store your programs on the *Turtle Tracks* disk itself. Skip this section and go on to Saving a Program.

If you have two disk drives, you may choose to use separate program storage disks instead. These are standard blank disks that you must first *initialize,* or prepare. You only do so once for each disk. Place a blank disk in disk drive 2, type INIT, and press RETURN. The disk drive whirs, and the computer soon says: INIT OK. That's all there's to it! You've made a program storage disk.

### Saving a Program

**1** If you are using an initialized storage disk, place it in disk drive 2 and go on to the next step. If you are using the *Turtle Tracks* disk as a storage disk, just go on to the next step.

**2** Type SAVE followed by a program name. (If you don't give your program a name, the computer will say: PROBLEM WITH FILE NAME.)

**3** When the computer asks, tell it which disk drive to use. That will be 1 if you're using the *Turtle Tracks* disk, and 2 if you're using a separate storage disk.

**4** The program will be saved. It is still in the computer, so you may continue working on it.

### Please Erase First

If you try to save a program under the same name as another program already on the disk, the computer says: PLEASE ERASE FIRST. You then have two choices:

**1** Erase the old program. Instructions for erasing a program are in the next section.

**2** Change the name of your new program and simply type SAVE followed by the new name.

### Erasing a Program

To erase a program, type ERASE followed by the program name. Tell the computer which disk drive to use when it asks you. The computer soon says: FILE ERASED.

### Loading a Program

When you load a program into the computer, it does not erase the program already in the turtle's memory. Lines in the new program, however, will replace any lines in the old program that have the same line number.

To avoid confusion, you will often, but not always, decide to type N and press RETURN before you Load in a new program.

1. If you are loading a program from a program storage disk, place that disk in disk drive 2, and go on to the next step. If you are loading a program from the *Turtle Tracks* disk, just go right to the next step.
2. Type LOAD followed by the program name.
3. When the computer asks, tell it which disk drive it should load from.
4. The program quickly loads into the computer. On the screen, you'll see which line numbers are being replaced, and which words are being created. Then, the prompt (>) and the cursor appear. You're ready to work on your new program.

### Catalog of Programs

Type CATALOG and press RETURN, and you'll see a catalog of the programs you've saved on your disk. When the computer asks, tell it which disk drive your disk is in.

## Using a Printer

PRINTING PROGRAM LISTINGS

If you type LPAPER and press RETURN, the printer will print out the list of instructions for the *Turtle Tracks* program you are working on.

PRINTING SCREENS

If you type PAPER and press RETURN, the printer will print out the turtle screen. First, of course, you should Run your program so that the turtle screen shows.

However, the printer will not print graphics characters. Instead, it will print question marks.

# Appendix B

## A Glossary of *Turtle Tracks* Instructions

### Statements

(Statements require line numbers, though none are shown here.)

| | |
|---|---|
| **DF5** | Draw Forward 5 steps. (Lesson 1) |
| **DF-4** | Draw backward 4 steps (Lesson 2) |
| **JF3** | Jump Forward 3 steps. (Lesson 2) |
| **JF-2** | Jump backward 2 steps. (Lesson 2) |
| **TR** | Turn Right. (Lesson 1) |
| **TL** | Turn Left. (Lesson 1) |
| **GT50** | Go To line 50. (Lesson 3) |
| **X=7** | X is 7. (Lesson 7) |
| **Y=Y+1** | Raise the value of Y by 1. (Lesson 7) |
| **PA** | Print A's. (Lesson 5) |
| **PB;3** | Print B's, orange on white. (Lesson 6) |
| **P#34** | Print character number 34. (Lesson 5) |
| **B** | Beep. (Lesson 6) |
| **B#10;100** | Beep at pitch 10; hold for duration 100. (Lesson 6) |
| **E** | End. (Lesson 4) |
| **M#2** | Begin micromode, and create a new character #2. (Lesson 10) |
| **EM** | End micromode. (Lesson 10) |
| **'Reminder** | Message that the turtle skips. (Lesson 5) |
| **BL, BLA, BLB, RL3, RLA6, RLB8** | Begin Loop and Repeat Loop instructions. (Lessons 3 and 9) |

**Commands**

(Commands don't use line numbers.)

| | |
|---|---|
| **L** | List entire program, or series of instructions. (Lesson 1) |
| **R** | Run the main program. (Lesson 1) |
| **W** | Walk through the main program. (Lesson 3) |
| **L WORD,**<br>**R WORD,**<br>**W WORD** | List, Run, or Walk miniprogram BOX. (Lesson 8) |
| **D** | Directory of miniprograms. (Lesson 8) |
| **TO WORD** | Readies computer to create miniprogram WORD. (Lesson 8) |
| **H** | Displays Help screens. (Lesson 5) |
| **C** | Clears screen. (Lesson 4) |
| **N** | New program. Erases old programs and all miniprograms. (Lesson 1) |
| **CATALOG** | Displays names of programs saved on a disk. (Lesson 4) |
| **LOAD** | Enters program from disk into computer. (Lesson 4) |
| **SAVE** | Stores program on disk. (Appendix A) |
| **PAPER** | Prints turtle screen. (Appendix A) |
| **LPAPER** | Prints program listing. (Appendix A) |
| **INIT** | Prepares program storage disk. (For use with disk drive 2 only.) (Appendix A) |

# Appendix C

## Some Common Mistakes

| | |
|---|---|
| **5 TR3** | You may not tell the turtle to turn right three times. Do not follow TR or TL with a number. |
| **DF3** | You forgot to use a line number with statement DF3. All statements require line numbers. |
| **5 R** | Do not use line numbers with commands R, N, W, L, D, SAVE, LOAD, or TO (as in TO SQUARE). |
| **10 BL4** | Do not follow BL (Begin Loop) with a number. The number should be used with the RL (Repeat Loop) statements. |
| **5 DF2.5** | You may only use decimals as line numbers—not in JF or DF statements. |
| **10 P90** | If you want the turtle to draw character number 90, type: 10 P#90. |
| **5 P ♥ ◇** | The turtle will only print one character at a time. When you want it to switch, use a new P statement. |

# Challenge Solutions

## Lesson 1: Suggested Solutions

**1**

```
1 DF5
2 TR
3 DF5
4 TR
5 DF5
6 TR
7 DF5
```

**2**

```
1 DF4
2 TR
3 DF4
4 TR
5 DF4
6 TR
7 DF4
```

(Notice that this program uses DF4 statements to build a square of five balls to a side. The fifth ball is drawn when the turtle begins drawing the next side.)

**3**

```
1 DF2
2 TR
3 DF6
4 TR
5 DF2
6 TR
7 DF6
```

**4**

Capital L

```
1 TL
2 DF4
3 TR
4 DF8
```

Capital T

```
1 DF9
2 TR
3 DF3
4 TL
5 TL
6 DF7
```

## Lesson 2: Suggested Solutions

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 JF-11 | 1 JF12 | 1 TR | 1 DF4 | 1 DF1 |
| 2 DF3 | 2 TL | 2 DF1 | 2 TR | 2 TR |
| 3 TR | 3 JF-10 | 3 TL | 3 DF1 | 3 DF1 |
| 4 DF3 | 4 DF2 | 4 JF1 | 4 TR | 4 TR |
| 5 TR | 5 TL | 5 TR | 5 JF1 | 5 DF1 |
| 6 DF3 | 6 DF2 | 6 DF1 | 6 TL | 6 TR |
| 7 TR | 7 TL | 7 TL | 7 DF1 | 7 DF2 |
| 8 DF3 | 8 DF2 | 8 JF1 | 8 TR | 8 JF2 |
| | 9 TL | 9 TR | 9 JF1 | 9 TL |
| | 10 DF2 | 10 DF1 | 10 TL | 10 JF2 |
| | 11 JF-23 | 11 TL | 11 DF1 | 11 DF1 |
| | 12 TR | 12 JF1 | 12 TR | 12 TL |
| | 13 JF-19 | 13 TR | 13 JF1 | 13 DF7 |
| | 14 DF2 | 14 DF1 | 14 TL | 14 TL |
| | 15 TL | 15 TL | 15 DF1 | 15 DF2 |
| | 16 DF2 | 16 JF1 | 16 TL | 16 JF4 |
| | 17 TL | | 17 JF-1 | 17 DF2 |
| | 18 DF2 | | 18 DF5 | 18 TL |
| | 19 TL | | 19 JF4 | 19 DF2 |
| | 20 DF2 | | | 20 TL |
| | | | | 21 DF2 |
| | | | | 22 TL |
| | | | | 23 DF2 |
| | | | | 24 JF-7 |
| | | | | 25 DF2 |
| | | | | 26 TL |
| | | | | 27 DF2 |
| | | | | 28 TL |
| | | | | 29 DF2 |
| | | | | 30 TL |
| | | | | 31 DF2 |
| | | | | 32 JF-16 |

## Lesson 3: Suggested Solutions

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 1 BL | 1 BL | 1 BL | 15 BL |
| 2 DF3 | 2 DF1 | 2 DF1 | 16 DF1 |
| 3 TR | 3 TR | 3 TR | 17 TR |
| 4 RL4 | 4 JF1 | 4 JF1 | 18 JF1 |
| 5 JF6 | 5 TL | 5 TL | 19 TL |
| 6 TR | 6 RL3 | 6 RL3 | 20 RL3 |
| 7 JF-3 | 7 TR | 7 TR | 21 TR |
| 8 BL | 8 BL | 8 BL | 22 BL |
| 9 DF9 | 9 DF1 | 9 DF1 | 23 DF1 |
| 10 TR | 10 TR | 10 TR | 24 TR |
| 11 RL4 | 11 JF1 | 11 JF1 | 25 JF1 |
| | 12 TL | 12 TL | 26 TL |
| | 13 RL3 | 13 RL3 | 27 RL3 |
| | 14 TR | 14 TR | |
| | 15 TR | | |
| | 16 DF7 | | |

## Lesson 5: Suggested Solutions

### 1

Here's how you would tell the turtle to write TOM SMITH:

```
 1 TR
 2 PT
 3 DF1
 4 PO
 5 DF1
 6 PM
 7 DF1
 8 JF1
 9 PS
10 DF1
11 PM
12 DF1
13 PI
14 DF1
15 PT
16 DF1
17 PH
18 DF1
```

### 2

```
 1 TR
 2 P#52
 3 DF1
 4 P#47
 5 DF1
 6 P#45
 7 DF1
 8 JF1
 9 P#51
10 DF1
11 P#45
12 DF1
13 P#41
14 DF1
15 P#52
16 DF1
17 P#40
18 DF1
```

### 3

```
 1 P#119
 2 TR
 3 DF1
 4 TR
 5 P#120
 6 DF1
 7 TR
 8 P#118
 9 DF1
10 P#117
11 DF1
12 JF3
```

### 4

```
10 TR
20 P#112
30 TL
40 JF1
50 TR
60 DF20
```

## Lesson 6: Suggested Solutions

**1**

```
1 B#25
2 B#27
3 B#29
4 B#30
5 B#32
6 B#34
7 B#36
8 B#37
```

**2**

```
1 B#18
2 B#20
3 B#22
4 B#23
5 B#25
6 B#27
7 B#29
8 B#30
```

**3**

```
 10 P#0;10
 20 BL
 30 DF24
 40 TR
 50 JF1
 60 TL
 70 RL11
 80 P#0;16
 90 BL
100 DF24
110 TR
120 JF1
130 TL
140 RL10
```

## Lesson 7: Suggested Solutions

**1**

```
1 JF12
2 TR
3 JF-9
4 X=33
5 BL
6 P#X
7 DF1
8 X=X+1
9 RL10
```

**2**

```
  5 JF4
 10 TR
 20 JF-3
 30 X=0
 40 BL
 50 PT;X
 60 DF1
 70 PU
 80 DF1
 90 PR
100 DF1
110 PT
120 DF1
130 PL
140 DF1
150 PE
160 DF1
170 JF-6
180 TR
190 JF3
200 TL
210 X=X+1
220 RL4
```

**3**

```
 5 TR
10 JF-8
20 X=0
30 BL
40 PA;X
50 DF1
60 X=X+1
70 RL16
```

**4**

```
10 BL
20 P#72;X
30 DF1
40 X=X+1
50 RL24
60 TR
70 JF1
80 TL
90 GT10
```

## Lesson 8: Suggested Solutions

**1**

```
TO SLANT
 5 BL
10 DF1
20 TR
30 JF1
40 TL
50 RL5
TO SQUARE
 5 BL
10 DF10
20 TR
30 RL4
```

```
 1 TR
 2 JF-5
 3 TL
 5 SLANT
10 TR
20 SLANT
30 TR
40 SQUARE
```

**2**

```
TO SMILE
 5 DF1
15 TL
20 DF7
25 TL
30 DF2
TO SQUARE
 5 BL
10 DFX
20 TR
30 RL4
```

**4**

```
 4 X=2
 5 SQUARE
10 TL
20 JF3
30 SQUARE
40 JF-1
50 TR
60 JF-3
70 X=1
80 SQUARE
90 TL
100 JF3
110 TL
120 JF1
125 SMILE
140 JF-4
```

## Lesson 9: Suggested Solutions

**1**

```
 10 X=1
 20 BLB
 30 BL
 40 DFX
 50 TR
 60 RL4
 70 JF-2
 80 TL
 90 JF2
100 TR
110 X=X+4
120 RLB4
```

**2**

```
1 BL
2 BLA
3 DF1
4 TR
5 JF1
6 TL
7 RLA3
8 TR
9 RL4
```

**3**

```
 10 BLB
 20 BLA
 30 BL
 40 DF1
 50 TR
 60 JF1
 70 TL
 80 RL3
 90 TR
100 RLA4
110 TL
120 JF8
130 TR
140 RLB2
```

**4**

```
  5 X=2
 10 TL
 20 JF2
 30 TR
 40 BLB
 50 BLA
 60 BL
 70 DF1
 80 TR
 90 JF1
100 TL
110 RLX
120 TR
130 RLA4
140 TL
150 JF3
160 TR
170 X=X+3
180 RLB3
```

## Lesson 10: Suggested Solutions

TO SLANT

1 BL
2 DF1
3 TR
4 JF1
5 TL
6 RL3

1 M#2
2 TR
3 JF2
4 DF4
5 TL
6 DF1
7 JF5
8 TL
9 DF3
10 TL

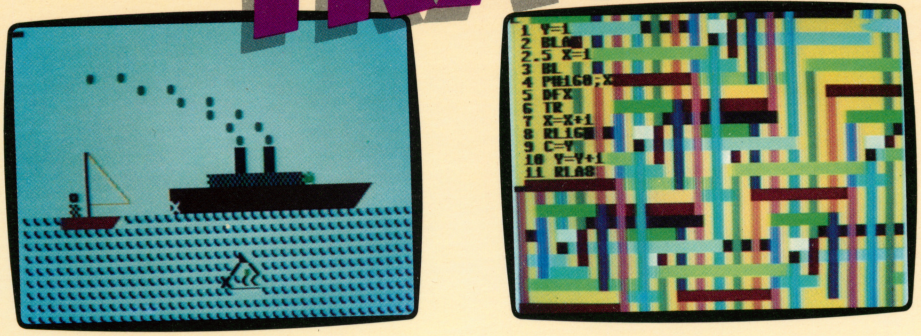
11 SLANT
12 DF1
13 TL
14 SLANT
15 TR
16 JF3
17 DF1
18 JF1
19 TR
20 DF4

# Turtle Tracks Notes

Scholastic Inc.

ISBN: 0-590-95110-6

# TURTLE TRACKS™



## *The LOGO-like graphics language.*

Master the turtle's easy-to-learn language and use it to design and paint intricate computer canvasses. The TURTLE TRACKS handbook will show you how.

The turtle draws more than 100 shapes in a variety of color combinations. You can also teach the turtle to "sing" melodies spanning several octaves.

You can save your finished designs and print them, too.

The TURTLE TRACKS language draws heavily from LOGO and BASIC, two of the most popular microcomputing languages used in schools and homes today.

## *Learning Opportunities*

TURTLE TRACKS helps children explore the graphics capabilities of the computer. It also teaches programming skills, problem solving, and critical thinking.

**Contents:** 1 disk or cassette, 1 handbook

Designed and developed by Thomas R. Smith
(Atari version: with James F. Weider)
Art Direction and Package Design: Sandi Young


***Scholastic Guarantee***

This computer program has been thoroughly tested by ***Scholastic*** among teachers and children and is designed to challenge, stimulate and entertain your child. It reflects Scholastic's more than 60 years of experience in developing young minds.

ISBN 0-590-95105-X

95105

078073039951

*Scholastic Inc.*
*2931 East McCarty St.*
*P.O. Box 7502*
*Jefferson City, MO 65102*

Scholastic

TURTLE TRACKS™