Reston

Sprites, A Turtle, and TI LOGO

JIM CONLAN ▪ DON INMAN

with DYMAX

Sprites,
a Turtle
and TI LOGO

Sprites, a Turtle, and TI LOGO

Jim Conlan
Don Inman
with Dymax

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

Library of Congress Cataloging in Publication Data
Conlan, Jim.
Sprites, a turtle, and TI LOGO.
Includes index.
1. LOGO (Computer program language) I. Inman, Don.
II. Dymax (Firm) III. Title.
QA76.73.L63C66 1983 001.64'24 83-11129
ISBN 0-8359-7036-1

A Prentice-Hall Company
Reston, Virginia 20090

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Contents

1 Getting Started 1
2 Sprites 21
3 Create Your Own Commands 41
4 Meet the Turtle 59
5 "NAME, :NAME, NAME, [NAME] 81
6 Tile Tales 103
7 What Counts 125
8 The Joysticks 147
9 A Word About Lists 165
10 Saving and Recalling 187
Appendix LOGO Summary 209
Index 225

Preface

This book is for anyone who wants a friendly, playful introduction to the TI LOGO computer language. TI LOGO lets you do amazing things quickly and easily. In this book you will meet the 32 flying sprites, the drawing turtle, and the 256 character tiles. You will use the wonderful list processing ability of TI LOGO to make and create computer stories. You will learn how the joysticks can control the expressions on a face.

LOGO is easy to learn, but so powerful that you can create new commands yourself. LOGO grew out of the language LISP, used by researchers in artificial intelligence. LOGO shares the powers of its parent.

For those who have programmed before, Appendix A at the back of the book contains a complete list, with examples, of all the objects, procedures, relations and commands of the TI LOGO language. Play around; you can't hurt anything. Have fun with the world's most useful toy, and tool.

Jim Conlan
Don Inman

Acknowledgments

Two young people gave us special help. Rolly Conlan created many of the sprite shapes in the book. Lisa Albers tried many parts of the book and made good suggestions.

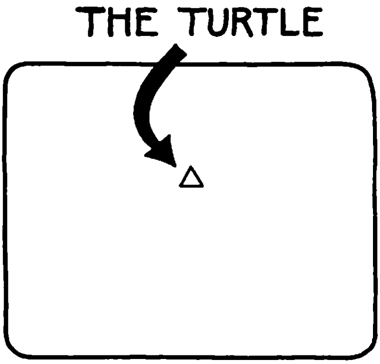
Sprites,
a Turtle
and TI LOGO

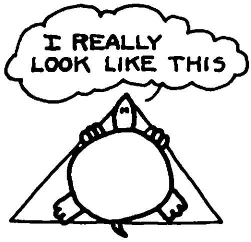
1

Getting Started

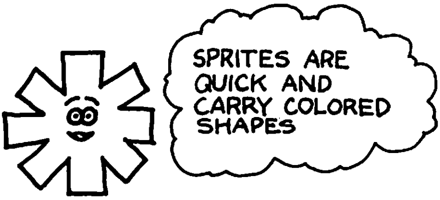
In this book you will learn how to talk to your TI99 computer using the LOGO computer language. The folks who developed LOGO noticed that people of all ages find it easy to talk to objects. You can talk to your cat, your dog or your best friend.

You will talk to objects using the LOGO language. The LOGO objects come in many forms. One of the objects you will talk to is the turtle. The turtle is a small triangle that you can move around the video screen. The turtle will draw for you.





LOGO also has objects called sprites. Sprites are quick and industrious, but invisible. You can tell the sprites to carry familiar shapes that you can see, such as a ball, a truck or a plane. You can even design your own shapes for the sprite to carry. You'll learn how to talk to LOGO objects in no time at all.



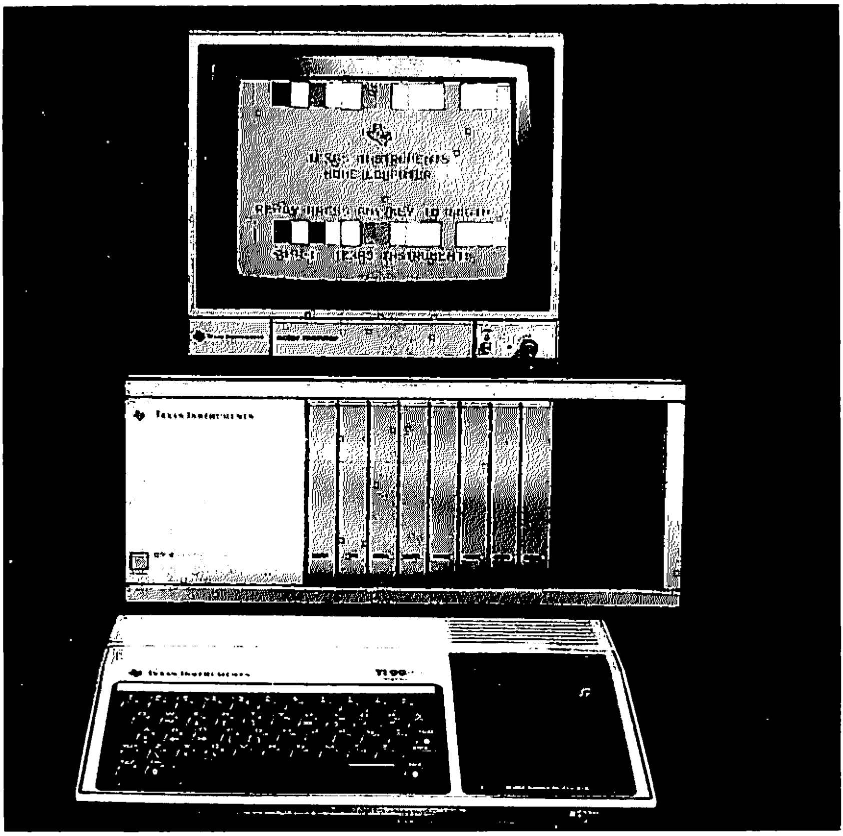
It's time to get started. Here is the minimum equipment you will need to start.

- TI99/4A or TI99/4 computer
- 32K memory expansion module
- TV set plus a video modulator

  or

  a video monitor
- a TI LOGO command module

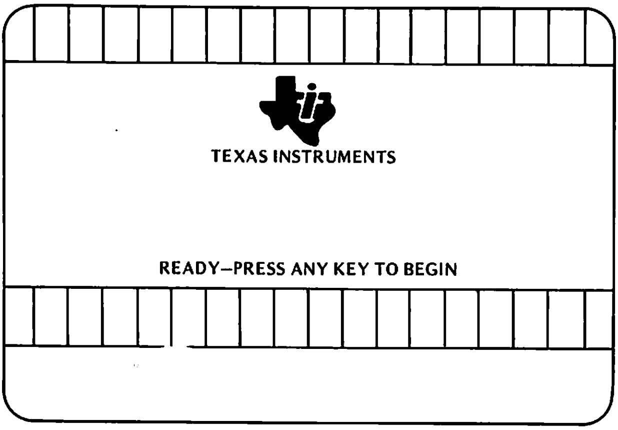
If you want to get a little fancier, Texas Instruments has developed a Peripheral Expansion Unit which will allow you to easily add extra power to your TI99 computer. The Peripheral Expansion Unit has 8 slots where extra boards may be inserted. These boards may contain extra memory or control disk drives and other devices.



To get started:

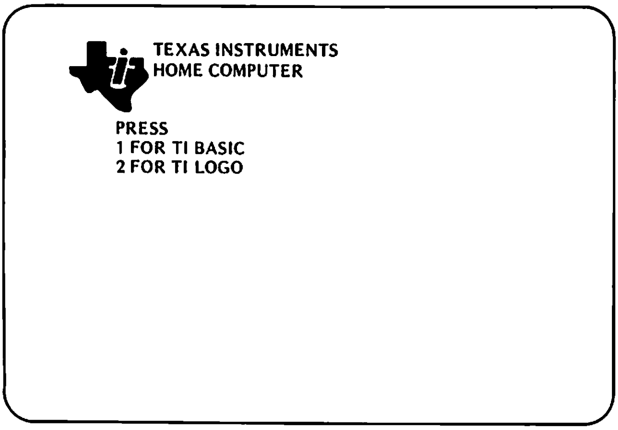
- Attach any accessories such as cassette tape recorders, disk drives or printers
- Slide the TI LOGO module into its slot on the right side of the keyboard
- If you are using a disk drive, turn the controller and disk drive on first
- Next turn on the memory expansion unit and any other devices
- Now turn on the computer console.

Here is what you see:



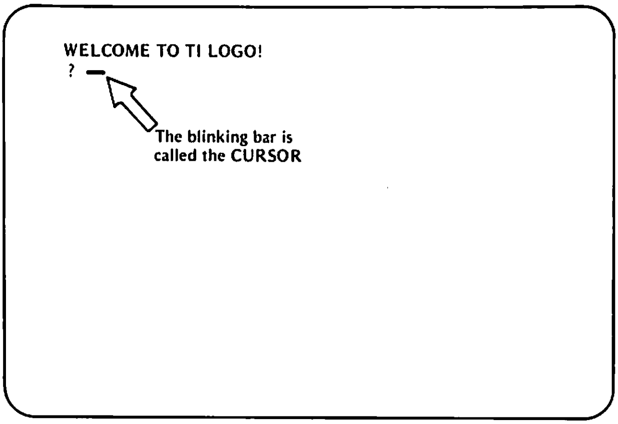
```
TEXAS INSTRUMENTS

READY—PRESS ANY KEY TO BEGIN
```

Press any key. Now you see this:



```
TEXAS INSTRUMENTS
HOME COMPUTER

PRESS
1 FOR TI BASIC
2 FOR TI LOGO
```

You are going to use the LOGO language. Type the number 2. You see this:



```
WELCOME TO TI LOGO!
? _
```

The blinking bar is called the CURSOR

You are ready to talk to your TI99 computer in LOGO. The computer is now in the command mode. The computer will do any legal command that you give it. Not every command is legal. Type this:

HI (ENTER) ☜ then press the (ENTER) key

In the future: (ENTER) means press the (ENTER) key.

You see this:

```
TELL ME HOW TO HI
?_
```

The computer doesn't understand the word HI. HI is not a legal command for LOGO. When LOGO doesn't understand something, it tells you. Let's try a legal command that the computer does understand. Type this:

BEEP (ENTER) ☜ that's the enter key

You will hear a steady, high-pitched tone, or beep. Adjust the volume on your TV so you can hear the beep. Lucky for you, the computer also understands the command NOBEEP. When you are tired of the beep, type this:

NOBEEP (ENTER)

The computer stops its beeping. BEEP and NOBEEP are both legal commands that LOGO understands.

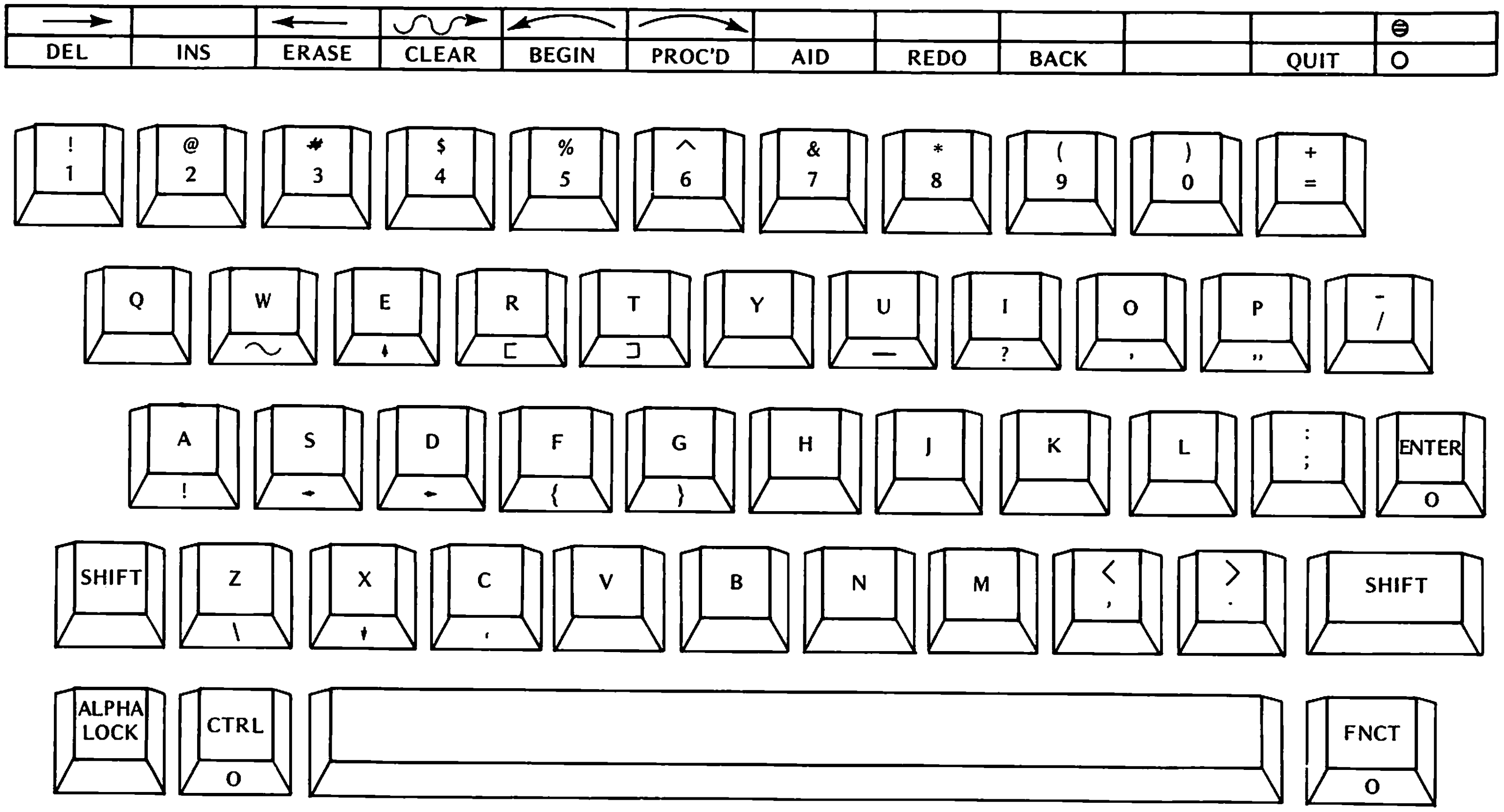
The Keyboard

Page 6 shows the TI99/4A keyboard.

▷ Put a reminder strip in its slot. It will help. Make your own out of paper or use the strip provided.

All those keys will soon be dear friends. They help you give commands to the computer. There are some special keys that you will use often.

If you have a TI99/4 computer: Your keyboard has the same keys, but they work differently than the TI99/4A keys. Check your *User's Manual* to find the keys and how to use them. To keep this book simple we will only talk about the TI99/4A keyboard.


DEL
INS
ERASE
CLEAR
BEGIN
PROC'D
AID
REDO
BACK
QUIT
⊜
O
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
+ =
Q
W ~
E ↑
R [
T]
Y
U _
I ?
O '
P "
- /
A !
S ←
D →
F {
G }
H
J
K
L
: ;
ENTER O
SHIFT
Z \
X ↓
C `
V
B
N
M
< ,
> .
SHIFT
ALPHA LOCK
CTRL O
FNCT O

The FUNCTION Key (FNCT)

Find the key labeled (FNCT) at the bottom, right side of your keyboard. That's the FUNCTION key. It makes some of the other keys act in special ways. The FUNCTION key (FNCT) is always used with another key. The (FNCT) key does nothing by itself. It will make the (3) key act like an eraser.

The ERASE Key (FNCT)(3)

The (ERASE) key is the most useful key on the keyboard. Whenever you make a mistake you can cure it with the (ERASE) key. Type this:

```
IT WORKS
```

Now hold down the (FNCT) key and type the (3) key. You see the cursor hop left, erasing as it goes. (The cursor is the blinking line that shows where you are about to type.) Keep erasing until the screen looks like this:

```
IT
```

Now type PLAYS. The screen should look like this:

```
IT PLAYS
```

Press the (ENTER) key. This sends the line you typed to the computer. You will see this:

```
TELL ME HOW TO IT
```

The computer doesn't understand the word IT. IT is not a legal command.

Legal Characters

You can see that all the letters of the alphabet are on the keyboard. TI LOGO uses only the capital letters.

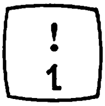
- Type a row of eight A's.
  Now erase 4 of the A's and replace them with B's.
  Now erase 2 of the B's and replace them with C's.
  Now you are an expert eraser.
- Notice the symbols written on the front side of the keys facing you. Hold down the (FNCT) key and type these characters:

  ? " ' [ ]

  These five characters are all used in the LOGO language.

The SHIFT Key

The (SHIFT) key is so important that it appears twice near the bottom of the keyboard. When you hold down the (SHIFT) key, the other keys print the upper symbol on the top of the key. Here is a picture of the (1) key:



Type the (1) key. You see the number 1 on the screen. Now hold down the (SHIFT) key while you type the (1) key. Now you see the exclamation mark !.

- Type these 8 characters:

  * ( ) + - : < >

  These eight characters are all used in the LOGO language.

How to Clear the Screen

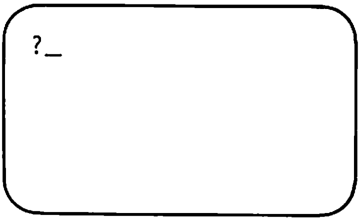
Is the screen beginning to look a bit messy? The CLEARSCREEN command is just what you need. Type this:

CS (ENTER) Clear screen

or this

CLEARSCREEN (ENTER)

The screen is cleared. You see:



```
?_
```

a blank screen

CS stands for CLEARSCREEN. Both the long and the short versions are LOGO commands that the computer understands. Use whichever you prefer.

There are long forms and short forms for many of the commands of the LOGO language. The short form is easy to type. The long form is easy to understand. If you write LOGO using the long form your LOGO commands will be easy for others to read and understand. If you use the short form, typing will go much faster. Take your pick.

- Type some junk, press the (ENTER) key. Now clear the screen using CS.

A LOGO Surprise

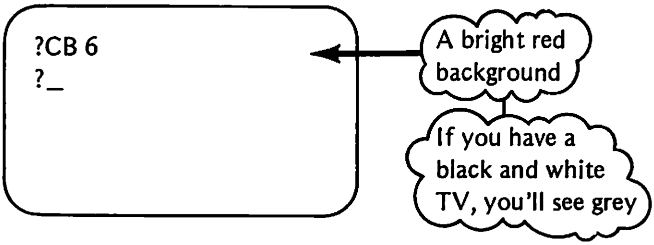
Here is a short LOGO command to try. Type this:

be sure to leave a space here

CB 6 (ENTER)

or

COLORBACKGROUND 6 (ENTER)

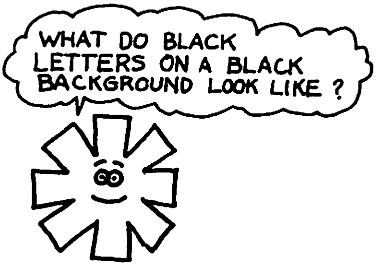


```
?CB 6
?_
```

The command CB is short for COLORBACKGROUND. Color number 6 is RED. You see a whole screen full of red.

- Type in each of the commands CS 2, CS 3, CS 4, etc., and match the color numbers with the color names:

CLEAR__
BLACK__
GREEN__
LIME__
BLUE__
SKY__
RED__
CYAN__
RUST__
ORANGE__
YELLOW__
LEMON__
OLIVE__
PURPLE__
GRAY__
WHITE__



An Easy Way to Remember Colors

You don't need to remember the numbers for the colors. The names of the colors also work. Try this:

TO GET (:) TYPE (SHIFT)(:)

```
CB :RED (ENTER)
```

The computer knows that :RED stands for 6.

- Try this:

```
CB :WHITE ☜ :WHITE STANDS FOR 15
```

- Now let's get back to the normal background color. The normal background color is a light blue called CYAN. CYAN is color number 7. Type this:

```
CB 7
```

or

```
CB :CYAN
```

The Famous Colon Mark (:)

The colon mark (:) has a very special use. Names of numbers, words and lists start with a colon mark (:). The colon mark tells the computer that you are referring to something by its name. For example, :RED is the name of the number 6. The number 6 is called :RED in TI LOGO.

The PRINT Command

You can tell the computer to print. Here is an example to try:

```
PRINT "HI (ENTER)
```

You see this:

(FNCT)(")

```
PRINT "HI
HI
?_
```

The computer just did what it was told. It printed HI. The quote mark (") has a special use in LOGO. It tells the computer that you are using the word itself. When the computer sees the quote mark ("), it knows that you are not giving a command, or referring to a thing by a name. It is the word itself that you want printed. The computer types until it comes to a space. The computer stops printing when it finds the first space.

- Command the computer to print your first name. Remember to use the quote mark in front of your name.

Here's how we did it:

```
PRINT "BELMONDER (ENTER)
```

Here's what we saw:

```
PRINT "BELMONDER
BELMONDER
?_
```

- Try this. What happens?

```
PRINT "BELMONDER W. FUZWUMZEL
```

You see this:

```
PRINT "BELMONDER W. FUZWUMZEL
BELMONDER
TELL ME HOW TO W.
?_
```

The computer prints everything from " to the first space. It stops printing and tries to understand what comes next. It doesn't understand W.

How to Print Lists -- tag

You can tell the computer to print more than one word. The square brackets [and] are what you need. Do you see them on the front of the R and T keys? Hold down the (FNCT) key to get [and]. Type this:

```
PRINT [THIS IS A LIST]
```

You see this:

```
PRINT [THIS IS A LIST]
THIS IS A LIST
?_
```

The computer types everything between [and].

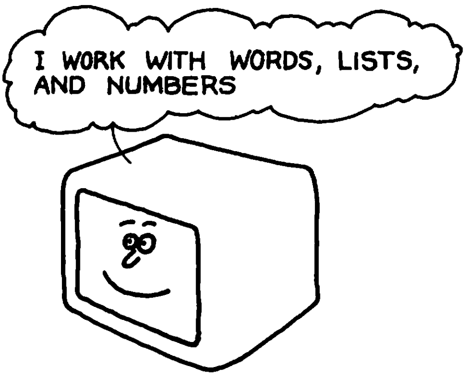
Printing Named Objects

- Give this a try:

```
PRINT :RED (ENTER)
```

You see this:

```
PRINT :RED
6
?_
```



What is happening? The colon mark (:) tells the computer that :RED stands for something else. In fact, :RED stands for the number 6 in TI LOGO. The computer prints the number 6.

- Try this:

```
PRINT :ALL (ENTER)
```

You see this:

```
PRINT :ALL
0 1 2 3 4 5 6 7 8 9 10 11 12 1
3 14 15 16 17 18 19 20 21 22 2
3 24 25 26 27 28 29 30 31
?_
```

The list of 32 numbers from 0 to 31 is called :ALL in TI LOGO. Later you will use :ALL to get the attention of all the sprites.

- Try this:

```
PRINT :BALL
```

You see the number 4 appear. Why do you suppose someone took the trouble to call the number 4 :BALL? Do you suppose there is a ball lurking somewhere?

How to Print all LOGO Names

- Clear the screen and type this:

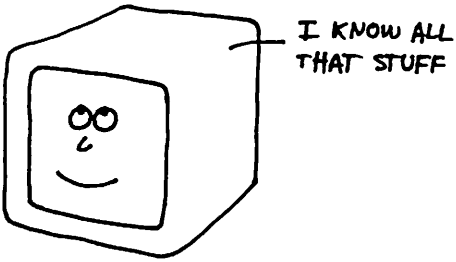
PN (ENTER) ☜ PN means PRINT NAMES

You see this:

```
?PN
"BOX IS 5
"BALL IS 4
"ROCKET IS 3
"TRUCK IS 2
"PLANE IS 1
"WEST IS 270
"SOUTH IS 180
"EAST IS 90
"NORTH IS 0
"WHITE IS 15
"GREY IS 14
"PURPLE IS 13
"OLIVE IS 12
"LEMON IS 11
"YELLOW IS 10
"ORANGE IS 9
"RUST IS 8
"CYAN IS 7
"RED IS 6
"SKY IS 5
"BLUE IS 4
"LIME IS 3
PRESS ENTER TO GO ON
```

Press the (ENTER) key and you see this:

```
"GREEN IS 2
"BLACK IS 1
"CLEAR IS 0
"ALL IS [0 1 2 3 4 5 6 7 8 9 1
0 11 12 13 14 15 16 17 18 19 2
0 21 22 23 24 25 26 27 28 29 3
0 31 ]
?_
```



These are the names the computer already knows. You will learn how to make names for numbers, words and lists later in this book.

- Command the computer to print out :NORTH. Do you get 0? Try :EAST, :SOUTH and :WEST.
- Have the computer print :RED + :BLUE. What is the answer?
- Have the computer print :WEST/:EAST.
- You can do ordinary arithmetic calculations. Try this calculation

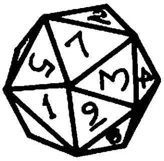
```
PRINT 5*2 + 1
```

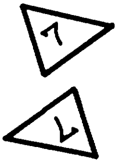
ANSWERS:

:EAST = 90, :SOUTH = 180, :WEST = 270
:RED + :BLUE = 6 + 4 = 10
:WEST/:EAST = 270/90 = 3
5*2 + 1 = 11

Some Numbers at Random

Pick a number from 0 to 9. You could use a die with 10 numbers on it. Here is a picture of a die with the ten digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 on it. In fact each number occurs exactly twice: once on the top and once on the bottom.





Roll a Ten-Sided Die

Your TI computer can roll a ten-sided die and print out the result.

- Type this:

```
PRINT RANDOM (ENTER)
```

You see some number like this:

```
5
```

The number you see may be any of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Try it again. Type this:

```
PRINT RANDOM (ENTER)
```

We got 0, you probably got something else.

A 100-Sided Die

Here is a command that makes a 100-sided die. Try this:

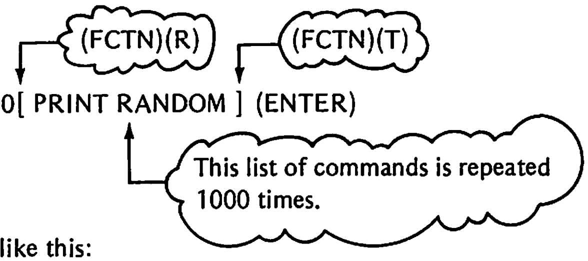
```
PRINT (RANDOM * 10 + RANDOM)
```

The computer multiplies a random digit by 10, then adds a different random digit. The result will be a random number between 00 and 99 like this:

```
62
```

The REPEAT Command

Your computer can do something more than once. Here is how you tell the computer to repeat a command 1000 times. This example prints 1000 random digits:



```
REPEAT 1000[ PRINT RANDOM ] (ENTER)
```

This list of commands is repeated 1000 times.

You see something like this:

```
5
3
0
7
2
5
. . .
```

The numbers flash by so fast you can hardly read them.

Here is a command that changes the background color 1000 times:

```
REPEAT 1000[ CB RANDOM ] (ENTER)
```

This gives you a very flashy display. When the display is finished your screen may not be a color you like. You can use the CB command to get a more pleasant color, or press (FNCT)(QUIT) to start the computer over again with the original color CYAN 7.

The WAIT Command

Did all that color changing go by too fast? Let's slow it down a bit. You will need to tell the computer to WAIT for a jiffy or two. Give this a try:

```
REPEAT 100[ CB RANDOM WAIT 60 ] (ENTER)
```

The computer colors the background with a random color, then waits for 60 jiffies, then repeats the process 100 times. One second is 60 jiffies.

- How long does WAIT 30 take?
- How long does WAIT 120 take?
- Write a command that will make the computer type HO HO 20 times.
- Write a command that will make the computer type HO HO once every second, for 20 seconds.
- Write a command that beeps for 1/2 a second, then is quiet for 1/2 second, then repeats the process 100 times.

ANSWERS:

```
1/2 second; 2 seconds; REPEAT 20[ PRINT [ HO HO ] ];
REPEAT 20[ PRINT [ HO HO ] WAIT 60 ]
REPEAT 100[ BEEP WAIT 30 NOBEEP WAIT 30 ]
```

Summary of Chapter 1

In this chapter you learned:

- How to hook up and turn on your TI99 computer
- That the computer only understands some commands
- That LOGO is the name of the language that the computer understands
- That the turtle is an object that appears as a small triangle on your screen and can be commanded to move and draw
- That the sprites are invisible objects that can carry visible objects about the screen
- That the (ENTER) key tells the computer when you are done typing a line

- That the cursor is the blinking line that shows where you are about to type
- That the computer types TELL ME HOW TO . . . when it doesn't understand a command
- That the command BEEP causes the computer to turn on a continuous tone
- That the command NOBEEP turns off the tone
- That the (FNCT)(ERASE) key erases mistakes
- That the command CS clears the screen
- That the command CB :RED colors the background RED
- That the command CB 6 colors the background RED
- That :RED is another name for the number 6
- That the colon (:) tells the computer the next word is the name for something
- That there are 16 colors, numbered 0 to 15
- That the command PRINT "HI causes the computer to print the word HI
- That the quote marks (") tell the computer to use what comes next
- That the command PRINT :RED causes the computer to print the number 6
- That the command PRINT [THIS IS A LIST] prints THIS IS A LIST
- That PN causes the computer to print the names of all named objects
- That RANDOM is the name for a number which is different each time you use it. RANDOM returns a digit between 0 and 9.
- That the command REPEAT 10[ ] causes any commands inside the parentheses to be repeated 10 times
- That the command WAIT 60 causes the computer to wait 60 jiffies (1 second)
- That certain colors, directions and shapes have numbers associated with them.

Predefined Variables

:ALL IS [0 1 2 . . . 31]

:CLEAR IS 0
:PLANE IS 1

:TRUCK IS 2
:ROCKET IS 3
:BALL IS 4
:BOX IS 5

:CLEAR IS 0
:BLACK IS 1
:GREEN IS 2
:LIME IS 3
:BLUE IS 4
:SKY IS 5
:RED IS 6
:CYAN IS 7
:RUST IS 8
:ORANGE IS 9
:YELLOW IS 10
:LEMON IS 11
:OLIVE IS 12
:PURPLE IS 13
:GRAY IS 14
:WHITE IS 15

:NORTH IS 0
:EAST IS 90
:SOUTH IS 180
:WEST IS 270

Self-Test—Chapter 1

1. Which of these words are *not* LOGO commands?

HI
BEEP
NOBEEP
PUNT
CS
CB 6
CB :RED
PRINT
UNDO
REPEAT

2. What number corresponds to the name :CLEAR?
3. How many different colors are available?
4. What punctuation mark tells the computer the next letters are the name of some object?
5. What punctuation mark tells the computer that the next letters themselves are being talked about?
6. Write a program that prints out the message "THIS WORKS" exactly 10 times.
7. What command tells the computer to color the background red?
8. Write a command that will turn on a tone, wait 60 jiffies, then stop.
9. What will the command PRINT :ALL do?
10. What command tells the computer to print a random digit from 0 to 9?

Answers

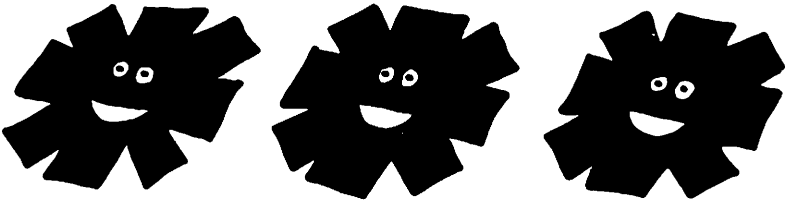
1. The computer understands all the commands except HI, PUNT and UNDO.
2. :CLEAR stands for the number 0.
3. Sixteen colors are available. They are numbered from 0 to 15.
4. The colon (:) tells the computer a name is about to start. Some typical names are :RED, :BOX and :TRUCK.
5. The quote mark (") tells the computer that you are talking about the letters themselves. For example, if you wanted to print the message HI, you would give the command PRINT "HI
6. REPEAT 10[PRINT "THIS WORKS]
7. CB :RED, or CB :6
8. BEEP WAIT 60 NO BEEP
9. The command PRINT :ALL will print the list of numbers 0 to 31.
10. PRINT :RANDOM

2

Sprites

In this chapter you will meet the sprites. The sprites are invisible computer creatures that can carry colored shapes and move on the screen. You can tell a sprite to set its color to a new value, or to carry a different shape, or to move faster or slower. In this chapter you will learn:

- How to TELL things to the sprites
- How to tell a sprite to CARRY a shape
- How to SET the COLOR of the sprite's shape
- How to send the sprite HOME to the center of the screen
- How to design your own sprite shapes
- How to SET the sprite's SPEED
- How to tell the sprites to go FORWARD, BACKWARD, LEFT and RIGHT
- How to SET the sprites HEADING to any direction
- How to talk to many sprites at once
- How to make beautiful, moving patterns of sprite shapes



How to Tell a Sprite

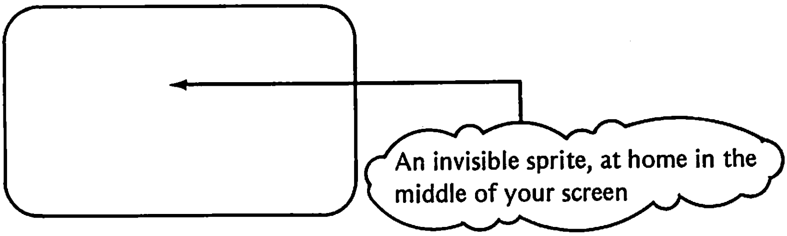
Sprites, you should know, are invisible. You can't see a sprite, but you can tell a sprite to do things. First you must get the sprite's attention. There are 32 sprites and each sprite has a number. The sprites are numbered from 0 to 31. Give a sprite's number to the computer. Type this:

```
TELL 0
```

You now have the attention of sprite number 0. Once you get sprite 0's attention, you can tell it all sorts of things. You can tell the sprite to go HOME to the middle of the screen. Type this:

```
HOME
```

The invisible sprite is now at home, right in the middle of your screen.



An invisible sprite, at home in the middle of your screen

A sprite needs a shape and a color so that you can see where it is. Here is how you tell the current sprite to carry a ball. Type this:

```
CARRY :BALL
```

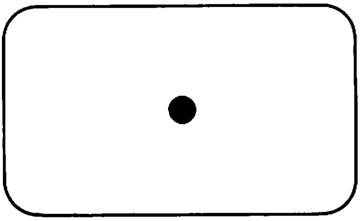
You can't see anything yet because the shape has no color. It is clear, like glass. Let's set the color of the shape to black. Type this:

```
SC :BLACK
```

or

```
SETCOLOR :BLACK
```

You see a black ball in the middle of your screen.



SC stands for SETCOLOR. If you have a color TV monitor, you can set other colors. Try this:

```
SC :RED
```

You see a red ball in the middle of your screen. You can use the color numbers, if you like. Try this:

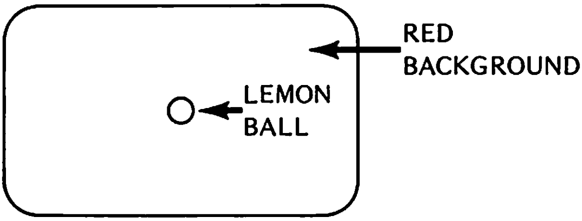
```
SC 11
```

You see a lemon-colored ball. A lemon ball might be more interesting on a red background. You can use the CB or COLORBACKGROUND command. Type this:

```
CB :RED
```

or

```
COLORBACKGROUND :RED
```



You see a lemon-green ball on a red background. If you are working with a black and white TV you will see a bright ball on a light grey background.

Color the Background Cyan 7

The usual color for the background is a nice light blue called CYAN. Cyan sounds like Siam. It is a lucky thing that cyan is the lucky number 7. Let's color the background cyan again. Type this:

```
CB :CYAN
```

or

```
CB 7
```

There, now you have your nice cyan blue background back.

- Try some SC and CB on your own. There are a lot of beautiful combinations to try.

Flasher

Here is a command that flashes the color of the ball from black to white.

```
HOME
SC 1
CB 7
REPEAT 100 [ WAIT 30 SC 1 WAIT 30 SC 15 ]
```

The sprite color changes between white and black every 30 jiffies.

The (FNCT)(BACK) Key

Sometimes you will want to stop the computer in the middle of its job. The (FNCT)(BACK) key does the job. Your computer doesn't have this key labeled unless you put in the long key name strip that comes with your computer. See page *6* to make your own strip. (FNCT)(9) is this key's other name. We'll call it the (FNCT)(BACK) key from now on. The (FNCT)(BACK) key puts you back in control of the computer.

You can make the last program more exciting by adding colors and flashing the background like this:

```
REPEAT 30 [ WAIT 20 CB 15 SC 1 WAIT 20 CB 1 SC 15 ]
```

Now the computer exchanges the color of the background with the sprite color every 20 jiffies.

Play around. Make some changes of your own. There are hundreds of possibilities here.

You can make the shape appear and disappear at random. Try this:

```
CB 7 SC 1
REPEAT 300 [ WAIT RANDOM SC 0 WAIT RANDOM SC 1 ]
```

More Shapes

There are other shapes available besides the ball. Type this:

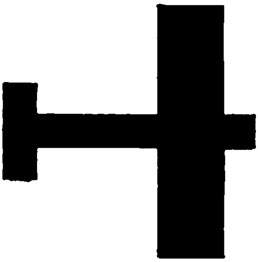
```
CARRY :BOX
```

You see a black box in the middle of a cyan background.

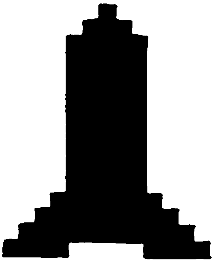
There are other shapes besides the BALL and BOX that the sprites can carry. Have the sprite carry these shapes:

```
CARRY :TRUCK
```

CARRY :PLANE



CARRY :ROCKET



The sprite shapes also have numbers. Try these and fill in the blanks with the names of the shapes you see:

CARRY 1 _____ CARRY 2 _____ CARRY 3 _____
CARRY 4 _____ CARRY 5 _____ CARRY 0 _____

How to Make Sprite Shapes

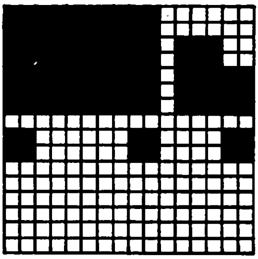
You have already seen the five shapes that are always available when you use the LOGO language. The truck, the plane, the rocket, the ball and the box are there when you turn on your computer with the LOGO cartridge in place. You can change these shapes and make new shapes of your own. Here is how you make sprite shapes. Type this to make sprite shape 1:

```
MS 1
```

or

```
MAKESHAPE 1
```

A large grid appears. Since shape number 1 is the truck, the grid contains the shape of the truck.



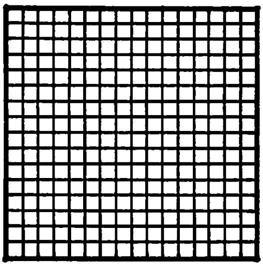
Arrow and (FCTN) Arrow

You can change sprite shape 1. The (FNCT) key and the arrows do the job. Here is the rule:

ARROW KEY CLEARS AND JUMPS
FUNCTION ARROW MARKS AND JUMPS

- Press the right arrow key (→). The cursor clears the square and jumps right.
- Hold down the (FCTN) key and press the right arrow key (→). The cursor marks the square and jumps right.
- Press the left arrow key (←). The cursor clears the square and jumps left.

Now it's your turn. Play around and draw a new design for sprite shape number 1.



When your shape is ready, hold down the (FNCT) key and press the (BACK) key [that's the (FCTN)(9) key]. The computer goes back to LOGO command mode. But now there is a new design stored for shape number 1. Shape 1 is no longer a truck, but the design you made. Take a look; type this:

```
CARRY 1 (ENTER)
```

The 26 Sprite Shapes

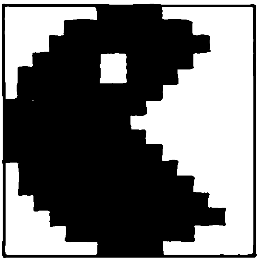
There are 26 sprite shapes possible. The shapes are numbered from 0 to 25. Only shapes 1, 2, 3, 4, and 5 have a design right now. The others are blank, waiting for you to give them a shape.

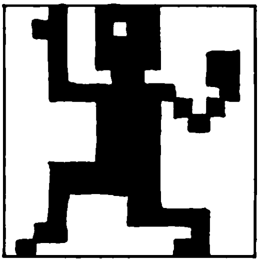
- Make a design for shape 6.

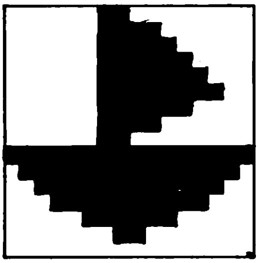
```
MS 6
```

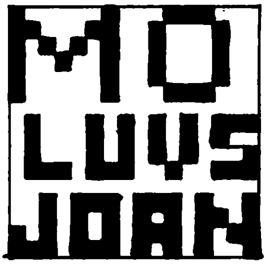
When you are done, type the (FNCT)(BACK) key. Now tell the sprite to carry your shape:

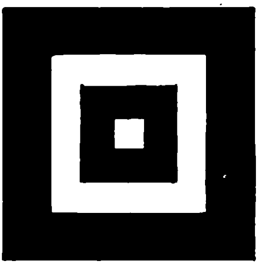
CARRY 6

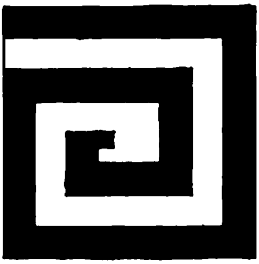












Forward and Backward

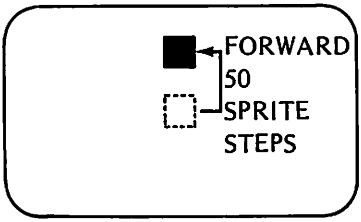
The sprite is quick. It can move quick as a wink. The FORWARD command makes the sprite move forward. Send the sprite HOME, and

CARRY :BOX

Then, try this:

```
FD 50
```

You see:



The sprite moved forward 50 sprite steps. One sprite step is 2 TV lines. The command FD 50 sends the sprite FORWARD 100 TV lines.

- Move the sprite so that its upper edge just touches the top of the screen.
- Move the sprite beyond the top edge of the screen until it reappears at the bottom of the screen.
- Send the sprite HOME. Then move the sprite FORWARD 8 steps at a time until the top of the sprite just touches the top of the screen. How many steps does it take?

When we measured the distance from HOME to the top of the screen we got 96 sprite steps.

The sprite can go BACK. Send the sprite home, then try this:

```
BK 50
```

or

```
BACK 50
```

The sprite moves backwards 50 steps.

Sprite Dance

- Here is a short command that makes the sprite dance up and down:

```
REPEAT 30 [ WAIT 20 FD 50 WAIT 20 BK 50 ]
```

- This command shows a rocket that may not make it to the moon:

```
CARRY :ROCKET
REPEAT 100 [ WAIT RANDOM FD RANDOM WAIT RANDOM BK
RANDOM ]
```

- Here is a command that moves the rocket ahead 8 steps at a time and beeps as it goes:

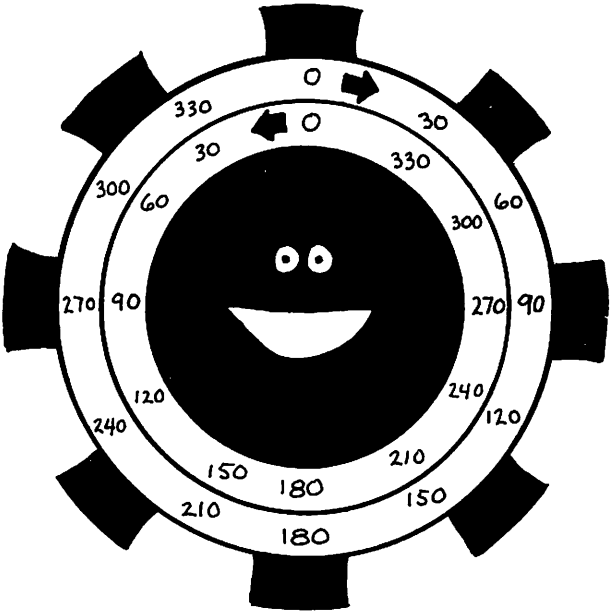
```
REPEAT 100 [ WAIT 5 FD 8 BEEP WAIT 5 NOBEEP ]
```

- This command sends the sprite 1000 steps at a time. But it comes back from the bottom. You can't tell where it will land.

```
REPEAT 20 [ WAIT 20 FD 1000 ]
```

Sprite Turn

Your sprite can do more than just jump forward and back. It can also turn to the RIGHT and turn to the LEFT. Here is a picture of the compass that the sprite uses:

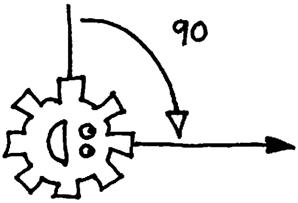


The sprite measures the angle of turn in degrees. Let's turn the sprite to the right by 90 degrees. First, send the sprite HOME and CLEARSCREEN, then try this:

```
RT 90
```

or

```
RIGHT 90
```



What do you see? Not much. The sprite turned 90 degrees, but since the sprite is invisible you don't see much. The shape stays in the same position. When the sprite moves you will see it move in a new direction. Type this:

```
FD 50
```

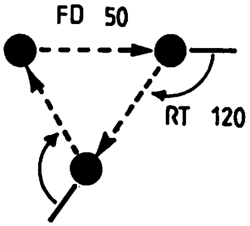
The sprite moves 50 steps toward the right side of the screen.

Square Deal

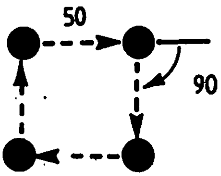
Here is a command that sends the sprite hopping around the corners of a triangle:

```
CARRY :BALL
REPEAT 30 [ WAIT 20 FD 50 RT 120 ]
```

You see this:



- Change the last command so that the sprite hops around the corners of a square like this:



Random Sprite

Here is a command that makes the sprite hop in a random spiral around the screen:

```
REPEAT [ FD RANDOM RT RANDOM ]
```

Set Heading

In which direction is the sprite headed? It is easy to lose track. There is a way to reset the sprite so that it is pointing up towards the top of the screen again. Type this:

```
SH 0
```

or

```
SETHEADING 0
```

The sprite immediately turns toward the top. Move the sprite to see that it really did turn toward the top:

```
FD 40
```

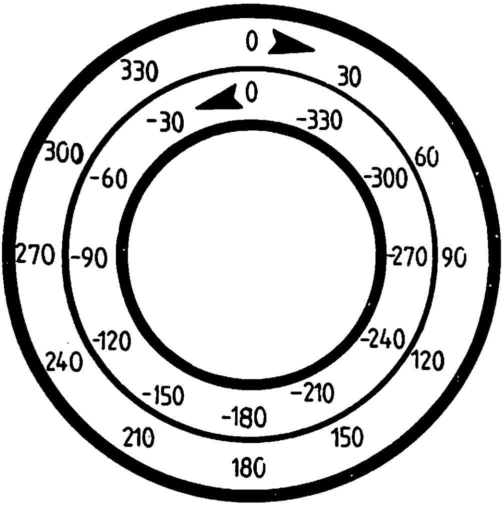
You can set other headings for the sprite. Type this:

```
SH 90
```

The sprite immediately heads toward the right side of the screen. Move the sprite forward to check that it really did turn:

```
FD 40
```

The sprite has a compass that tells which way is up. It looks like the compass that moves with the sprite, but this one doesn't turn. SH 0 always turns the sprite towards the top.

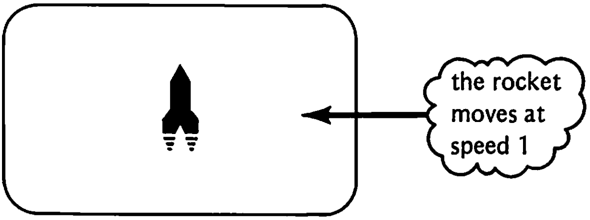


Sprite Speed

Your sprite does more than hop about. Sprites are smooth movers. You can set the speed of the sprite with the SETSPEED command. Try this:

```
SH 0
CARRY :ROCKET
SS 1 ☜ set speed 1
```

The sprite moves slowly upward at speed 1. Speed 1 is about 4 TV lines per second. You see:

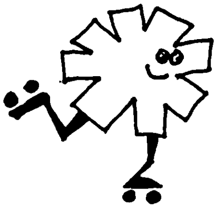


Would you like to see the sprite go faster? Type this:

```
SS 10 ☜ set speed 10
```

Zip! Now the sprite is moving faster. Notice that the sprite leaves the top of the screen and then reappears from the bottom of the screen. Would you like to see the sprite at its fastest speed? Of course you would. Type this:

```
SS 127 ☜ the fastest speed
```



Speed Backwards

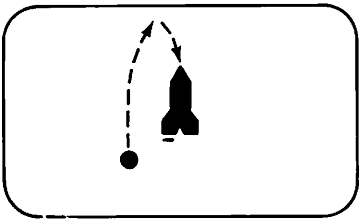
Send your sprite HOME, carrying a :ROCKET, with its color set :BLACK. Now, try this:

```
SS -10
```

The sprite now moves down, towards the bottom of the screen. The command SS -10 means backwards with speed 10. The number -10 is called "minus ten" or sometimes "negative ten." Take your pick. Try this:

```
SS 0
HOME
CS
REPEAT 20 [ WAIT 30 SS 30 WAIT 30 SS -30 ]
```

The sprite goes at speed 10 for 30 jiffies, then goes at speed minus 10 for 30 jiffies. You see the rocket bounce up and down the screen. When the REPEAT command is finished, the rocket keeps flying on its own.



Let's speed things up a bit. SETSPEED 0, send the sprite HOME and CLEARSCREEN. Now try this:

```
REPEAT 30 [ WAIT 20 SS 127 BEEP WAIT 20 SS -127 NOBEEP ]
```

Another Sprite

You have gotten to know sprite number 0 quite well. It is time to meet another member of the famous sprite family. First you need to get its attention. Type this:

```
TELL 1
```

Now you have the attention of sprite number 1. It has no shape or color yet and needs to be sent to HOME.

```
CARRY :BALL
SC 1
HOME
```

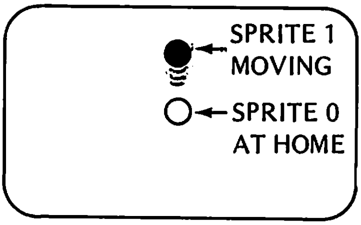
Sprite 1 is now in the center of your screen, facing up towards the top of the screen, carrying a black ball. Send it flying:

```
SS 20
```

Sprite 1 is now travelling upwards at sprite speed 20. You can use sprite 0 at the same time as sprite 1. Type this:

```
TELL 0
SC :WHITE
CARRY :BALL
HOME
```

This is what you see:



Sprite 0 is a white ball that sits at home in the middle of the screen. Sprite 1 is a black ball that glides up the screen. You still have sprite 0's attention; give it some speed.

```
SS 30
```

Now both the sprites are sailing up the screen. But sprite 0 is traveling faster.

Sprite Precedence

Did you notice that sprite 0 passes in front of sprite 1. The low-numbered sprites are in front of the high-numbered sprites. Sprite 1 can hide behind sprite 0. Try this:

```
SS 0
HOME
```

Sprite 0 is now stopped in the center.

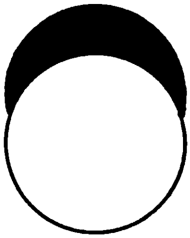
```
TELL 1
SS 0
HOME
```

Now sprite 1 is hiding behind sprite 0 at the home position.

Give sprite 1 a little nudge:

```
FD 8
```

Now you can see it peek out from behind sprite 0.



Talking to Two Sprites at Once

You can get the attention of both sprites 0 and 1 at the same time. Type this:

```
TELL [ 0 1]
```

You now have both sprites' attention. You can give them both a command. Try this:

```
FD 50
```

Both sprites move 50 steps.

```
SS 30
```

Both sprites begin to move at speed 30.

```
CARRY :PLANE
```

Both sprites carry the plane shape.

```
SC 15
```

Both sprites set their color number to 15 (WHITE).

```
SS 0
HOME
```

Both sprites set their speed to 0 and go home.

Bounce

Here are some commands that make sprites 0 and 1 appear to bounce. Let's set everything up from the beginning:

```
TELL [ 0 1 ]
HOME
SC 1
CARRY :BALL
```

Now give the balls movement in opposite directions:

```
SS 30
TELL 0
SS -30
```

Now start both balls together at the center:

```
HOME
```

You see two balls that appear to collide and bounce.

- Use the SS command to make the balls bounce faster. Try this:

```
RT 90
```

Now the balls have each turned and are moving in the horizontal direction. Start them both at the center:

```
HOME
```

Now the balls bounce in the horizontal direction.

Now it is your turn. Here are some things to try:

- Get more sprites flying. Fill the screen with sprites.
- Give the sprites different colors and shapes.
- Make your own special shapes and send them flying.
- Clever things from your own creative imagination.

The Big Finish

Here is something to try. These simple commands do some surprising things. The commands EACH and YN tell EACH of the current sprites that is listening to use its own number to do something.

We'll talk more about EACH and YOURNUMBER in the next chapter. Give these commands a try:

```
TELL :ALL ☜ ALL is [ 0 1 2 . . . 31 ]
CARRY :BALL
SC 15
HOME
```

The sprites are now ALL at home carrying a white ball.

EACH [SH 15*YN] ☜ SET HEADING to 15 times YOUR NUMBER

SS 30

We really can't describe the lovely moving patterns you will see. You'll just have to try it.

EACH [SC YN] ☜ SET COLOR to YOUR NUMBER

- Change the speed to 127. WOW!
- Send the sprites back to HOME.
- Play around with the commands. Change 15 to 60.

Summary of Chapter 2

In this chapter you learned:

- That sprites are invisible creatures that carry colored shapes and can move on the screen
- That TELL 0 opens communication with sprite 0
- That the command HOME sends the current sprite to center screen
- That the command CARRY :BALL causes the sprite to carry a ball
- That the SETCOLOR command SC :BLACK makes the sprite's color black
- That there are 32 sprites numbered 0, 1, 2, 3, up to 31
- That there are five numbered shapes 1 :PLANE, 2 :TRUCK, 3 :ROCKET, 4 :BALL, and 5 :BOX
- That shapes can be referred to by name or by number
- That the (FNCT)(9) alias (FNCT)(BACK) key stops the computer in the middle of a job and gives you back control
- That you can make your own shapes with the MAKESHAPE command MS
- That there are 26 shapes numbered from 0 to 25
- That the FORWARD command FD 20 causes the sprite to move forward 20 sprite steps
- That the BACK command BK 20 causes the sprite to move backwards 20 sprite steps
- That the RIGHT command RT 90 causes the sprite to make a right turn by 90 degrees

- That the LEFT command LT 90 causes the sprite to make a left turn by 90 degrees
- That the SETHEADING command SH 0 heads the sprite towards the top
- That the SETSPEED command SS 20 gives the sprite speed 20
- That you can set many sprites in motion at the same time
- That the command TELL [0 1] opens communication with sprites 0 and 1 at the same time
- That sprite 0 moves in front of sprite 1

Self-Test—Chapter 2

1. Which of these is the proper way to get a sprite's attention?

   HI SPRITE, LISTEN YOU, TELL 1, HELLO 5

2. Write the command that sends the sprite home to the middle of the screen: ____________________

3. Write the command that makes a sprite pick up shape 4.

   ____

4. Write the names of the five shapes that are initially available:

   __________ __________ __________ __________ __________

5. Write the command that sets the sprite's color green:

   ____

6. Write a command that makes the sprite carry a random shape:

   ____

7. Write a command that makes the sprite begin to move with speed 50:

   ____

8. Write a command that makes the sprite move forward 20 sprite steps:

   ____

9. Write a command that makes the sprite move backwards 20 steps:

   ____

10. Write a command that makes the sprite turn 90 degrees to right:

    ____

11. Write the command you use to make a new shape number 7:

    ____

12. What keys do you use to stop a program in the middle of a job, and also to get back from the make-shape mode?

____

Answers

1. To get the attention of sprite number 1, type TELL 1.
2. The command HOME sends the sprite home to the middle of the screen.
3. The command CARRY 3 makes a sprite pick up shape 3, the rocket shape, and carry it.
4. The names of the five shapes that the sprite knows how to carry are: 1 :PLANE, 2 :TRUCK, 3 :ROCKET, 4 :BALL, 5 :BOX.
5. The command SC :GREEN sets the color of the sprite green.
6. The command CARRY :RANDOM makes the sprite carry a random shape.
7. The command SS 50 makes the sprite move with speed 50.
8. The command FD 20 makes the sprite move forward 20 sprite steps.
9. The command BK 20 makes the sprite move backwards 20 sprite steps.
10. The command RT 90 causes the sprite to make a right turn of 90 degrees.
11. The command MS 7 or MAKESHAPE 7 causes the computer to enter the make-shape mode for shape 7.
12. The (FNCT)(9) key, alias (FNCT)(BACK), brings the computer back to the sprite mode where you can give commands.

3

Create Your Own Commands

In the last chapter you met the sprites. The sprites are a family of 32 invisible computer creatures that carry colored shapes and flit about the screen at your command.

Let's review the commands for getting a sprite to appear on the screen.

- TELL 1 ☜ get a sprite's attention
- CARRY :BALL ☜ command the sprite to CARRY a shape
- SC :BLACK ☜ SET the sprite's COLOR to :BLACK
- HOME ☜ send the sprite HOME to the center of the screen
- SS 20 ☜ SET the sprites SPEED to 20

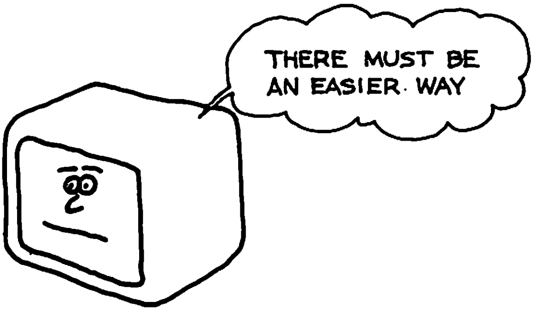
In this chapter you will learn how to make the computer save lists of commands for you. When you want the commands performed, you can just mention the name of the list and the computer will do all the commands in the list. In this chapter you will learn:

- That a list of commands is called a PROCEDURE
- That the computer has a special mode that makes it easy for you to make procedures
- That procedures allow you to create your own LOGO command words
- How to make brilliant and colorful patterns by typing a single letter

Waste Not, Want Not

It seems wasteful to keep typing the same list of commands again and again. You have typed this list of commands to start a sprite at least 10 times so far in this book:

```
CARRY :BALL
SC 1
HOME
```



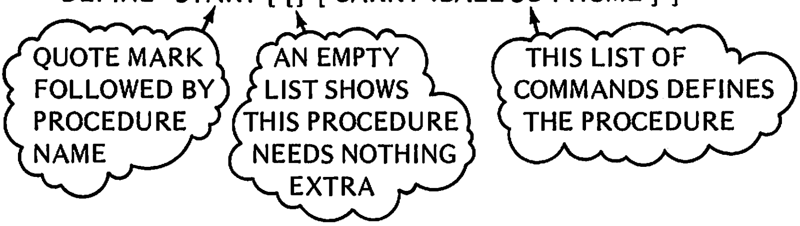
Rejoice! You can create a simple command that will accomplish this useful procedure. You can define your own commands in LOGO. The power to define new commands that perform useful procedures is what makes the LOGO language powerful.

Let's make a new command called START that will start a sprite. There are two ways to define a new command.

Define A Procedure

Here is an example that defines a new command called START. Type this:

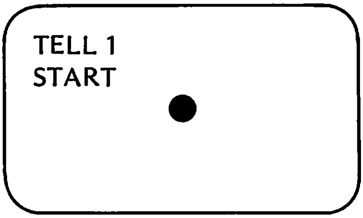
```
DEFINE "START [ [] [ CARRY :BALL SC 1 HOME ] ]
```


QUOTE MARK FOLLOWED BY PROCEDURE NAME
AN EMPTY LIST SHOWS THIS PROCEDURE NEEDS NOTHING EXTRA
THIS LIST OF COMMANDS DEFINES THE PROCEDURE

Now you have a new command to use. The computer understands the command START. Try your new command:

```
TELL 1
START
```

You see:



```
TELL 1
START
```

When you type START, the computer gets the list of commands that goes with the word START and performs them. A black ball appears at home in the middle of the screen.

Are you wondering about that empty list in the DEFINE statement?

```
DEFINE "START [ [ ] [ TELL 1 CARRY :BALL SC 1 HOME ] ]
```

THE LIST OF OBJECTS THAT THE PROCEDURE NEEDS TO DO ITS WORK. IN THIS CASE, NONE.

The definition of START is a list of lists. The first list [ ] of the definition is the empty list. The procedure START doesn't need any numbers or other objects from you. However, you must include the brackets [ ] to tell the computer that the first list is empty. Later you will see procedures that need some numbers or other objects to do their job.

Save Your Procedure

The procedure START will remain in your computer's memory until you turn off your computer. The procedure is lost when you turn the computer off. If you want to build a library of new procedures then you will need to save the definition on your cassette recorder or disk drive. Chapter 10 tells you how to save your new procedures.

- What will this procedure do when you type PUNT?

```
DEFINE "PUNT [ [] [ PRINT "GOOD ] ]
```

- What's wrong with this next definition?

```
DEFINE BADDY [ [] [ PRINT "BADNAME ] ]
```

- What's wrong with this next definition?

```
DEFINE "WRONGONE [ [ PRINT "OBJECTS? ] ]
```

ANSWERS:

The first procedure will print the word GOOD whenever the command PUNT appears.

The second definition is missing a quote mark (") in front of the procedure name BADDY.

The third definition is missing the list of objects used by the procedure. It is an empty list in this case.

- Try this procedure:

```
DEFINE"HI [ [] [ PRINT [ HELLO, HOW ARE YOU TODAY? ] ]
```

Now your computer is well mannered. Whenever you type HI, your computer will respond with HELLO, HOW ARE YOU TODAY?

Procedure Procedure

Here is the definition of a procedure that uses itself. What do you suppose will happen? Try it and see.

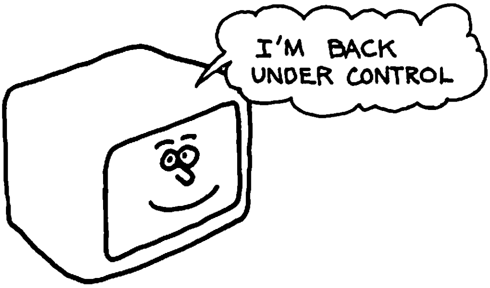
```
DEFINE "DOALOT [ [] [ PRINT "*** DOALOT ]
DOALOT
```

When you type DOALOT, you see this:

```
***
***
***
***
***
etc.
```

A procedure can use itself as the last command in the procedure list. When you type DOALOT, the computer performs the command PRINT "***, then it performs the command DOALOT. What does DOALOT do? It first performs the command PRINT "***, then it performs the command DOALOT. What does DOALOT do? Round and round it goes, doing a lot. The computer is caught in an infinite loop. It will never stop until you turn off the computer, or you regain control by pressing the (FNCT)(BACK) key, alias (FNCT)(9).

(FNCT)(BACK) GETS BACK CONTROL



- Write the definition for a procedure called ME that prints out your name over and over again.

To Be or Not To Be

There is a second way to define a new procedure. A special editor is available to help you. Type this:

```
TO ROLL (ENTER)
```

The whole screen changes. You see this:

```
TO ROLL_              ☜ The cursor
END
```

The computer is now in the procedure editing mode. The cursor is blinking at the end of the top line. Is the line alright? Yes. Press the (ENTER) key. Now you see this:

```
TO ROLL
_                     ☜ space for your line
END
```

The computer has made space for a new line. The cursor is resting at the beginning of the new line. You can enter commands. Add a command so your screen looks like this:

```
TO ROLL
PRINT RANDOM_         ☜ leave the cursor here
END
```

Now, get back to the regular command mode. Press the (FNCT)(BACK) key. The screen returns to its previous state. You see something like this:

```
TO ROLL               ☜ you typed this earlier
?_
```

The computer now knows a new procedure called ROLL. Give it a try.

```
ROLL (ENTER)          ☜ call the new procedure
```

You see a random number appear on the screen:

```
ROLL
7                     ☜ ROLL did its job
?_
```

Each time you type ROLL the computer prints a random digit.

You have just defined a procedure using the procedure editor. The procedure editor allows you to easily make changes to your procedures. Type this:

```
TO ROLL (ENTER)
```

The whole screen changes. You see this:

```
TO ROLL_
PRINT RANDOM
END
```

The cursor is resting at the end of the first line. You can use the (FNCT) arrows to move the cursor within the procedure lines.

The (FNCT) Arrows

When the computer is in the procedure editing mode, you can move the cursor from line to line and word to word using the (FNCT) key with the arrow keys. Hold down the (FNCT) key and use the arrows to move the cursor to the last line of the procedure like this:

```
TO ROLL
PRINT RANDOM
END                     ☜ the (ENTER) key will make a space
```

You can make a space for a new line like this: with the cursor at the beginning of a line, press the (ENTER) key. The cursor and the line it is on are moved down leaving a blank line. You see this:

```
TO ROLL
PRINT RANDOM
                        ☜ the (ENTER) key made space
END
```

You can now use the (FNCT) arrows to move the cursor up to the blank line and type a new command. Add a line so the procedure looks like this:

```
TO ROLL
PRINT RANDOM
BEEP WAIT 20 NOBEEP_ ☜ a new line
END
```

Press (FNCT)(BACK) to get back to command mode.

Try the new version of ROLL. Type:

```
ROLL
```

Now the program prints a random digit and also gives a short beep.

Some Editing Tools

The procedure editor has some valuable tools that make it easy for you to correct mistakes and change your procedure definitions. Get into the procedure editing mode again. Type:

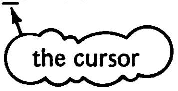
```
TO ROLL
```

The procedure definition appears:

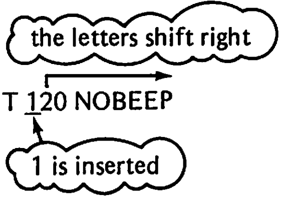
```
TO ROLL_
PRINT RANDOM
BEEP WAIT 20 NOBEEP
END
```

Use the (FNCT) arrows to move the cursor under the 2 like this:

```
BEEP WAIT 20 NOBEEP
```



Anything you type now will be inserted in the line at the cursor position. Type the (1) key. You see this:



```
BEEP WAIT 120 NOBEEP
```

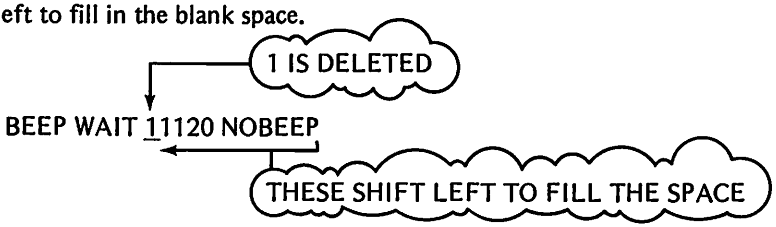
The letter under the cursor scooted to the right and the number (1) is inserted in the line. Press the (1) key a few more times. More 1's are inserted. Next you will see how to remove letters from a line.

Deleting

Your screen should look like this:

```
TO ROLL
PRINT RANDOM
BEEP WAIT 1111120 NOBEEP
END
```

You can delete whatever is under the cursor. Hold down the (FNCT) key and press the (DEL) key. The (DEL) key is the same as the (1) key on the TI99/4A. As you press (FNCT)(DEL) the extra 1's are removed and the other letters shift left to fill in the blank space.



```
BEEP WAIT 11120 NOBEEP
```

- Play around. Insert and delete letters until you feel like an expert.
- Put the cursor at the beginning of the line and delete the whole line by holding down the (FCTN) key and pressing the (DEL) key until the whole line is gone.

Notice what happens when you use the delete key while the cursor is on a blank line. The blank line is deleted and the remaining lines move up to fill the gap. This is useful for removing unwanted blank lines from your procedure.

```
TO ROLL
PRINT RANDOM
                (FNCT)(DEL) will delete the blank line.
END
```

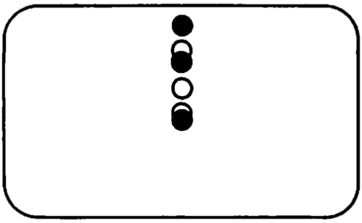
To Fly

Here is a procedure that sends sprites off at random, carrying random colors.

```
TO FLY
TELL RANDOM
CARRY :BALL
SC RANDOM
SS 30
HOME
FLY
END                 no space allowed here
```

When you are done typing the procedure, press (FNCT)(BACK). Then type:

```
FLY
```



You see multicolored balls created at the center and moving upwards. Press (FNCT)(BACK) to stop the procedure when you are ready. Notice that the balls continue moving on their own. If you add one extra command to this procedure it will act quite differently.

Add this new command to the procedure. Type:

```
TO FLY
```

Now move the cursor down to the beginning of line 3 like this:

```
   TO FLY             ☜ This is line 0
   TELL RANDOM
   CARRY :BALL
→  SC RANDOM          ☜ This is line 3
   SS 30
   HOME
   FLY
   END
```

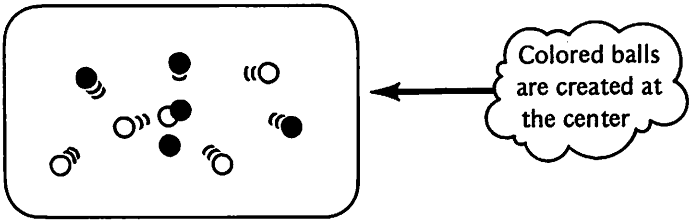
Press the (ENTER) key to make a space for a new line. Add this new line:

```
RT 10*RANDOM
```

Your procedure should look like this:

```
TO FLY
TELL RANDOM
CARRY :BALL
RT 10*RANDOM_
SC RANDOM
SS 30
HOME
FLY
END
```

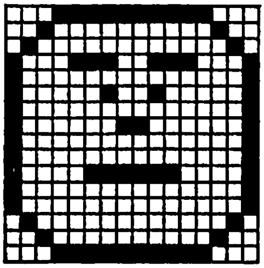
Type (FNCT)(BACK) and give the FLY command a try. You see this:



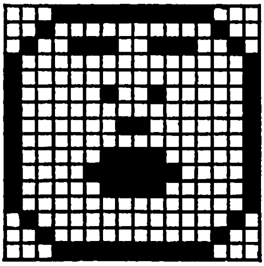
Colored balls are created at the center

Talking Face

This next procedure shows some of the graphics power of your TI99 computer and the LOGO language. This procedure makes a face that talks. The talking face will really be two face shapes that the sprite will carry. Here is the first sprite-shape for you to make. Type MS 6 and make this shape. Remember that (FNCT) arrow prints black, and the arrow alone prints clear.



When you are done making sprite-shape 6, type the (FNCT)(BACK) key to get back to command mode. Now type MS 7 and make this sprite shape:



When you are done making sprite-shape 7, press the (FNCT)(BACK) key. Now you are ready to make a procedure that switches the sprite back and forth between these shapes. Type TO TALK to get into the procedure editing mode. Type this procedure:

```
TO TALK
TELL 1
CARRY 6
NOBEEP
WAIT RANDOM
CARRY 7
BEEP
WAIT RANDOM
TALK
END
```

☜ no space allowed here

Notice that the procedure TALK calls itself into action again in the line just before the END. Be sure that there is no blank line left between TALK and END. TALK must recur at the tail to work properly in this procedure. We will see other possibilities later.

Everything is ready. Type

```
TALK
```

In the middle of the screen appears a beeping animated face with a moving mouth and eyebrows.

You can stop the procedure with the (FNCT)(BACK) key. If the procedure stops with the beep on, you can stop the beep with the NOBEEP command.

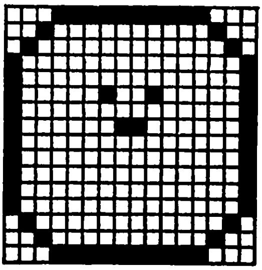
- Add a clearscreen command CS to the TALK procedure.
- Add a setspeed command SS 20 to the TALK procedure.
- Play around with sprite shapes 6 and 7. Make your own faces.
- Change the two WAIT RANDOM commands to this:

  ```
  WAIT 100/(RANDOM * RANDOM+1)
  ```

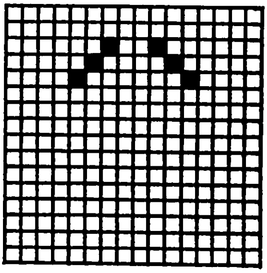
  Do you think this sounds more like someone talking?
- Make a third face on sprite-shape 8. Add a line to the procedure to make the sprite carry the new face for a while.
- Set the sprite's color :ORANGE and color the background :BLACK. You won't be able to see what you type until you color the background to CB 7.
- Create, create, create.

A Change of Face

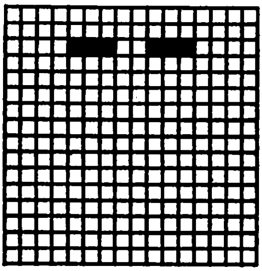
People understand how to read faces. You can tell a happy face from a sad one, an angry face from a straight face. This next procedure makes a face that can change its expressions. Different sprites will carry different parts of the face. Sprite 1 will carry the outline of the head and the eyes and nose. Sprite 2 will carry the eyebrows. Sprite 3 will carry the mouth. The first thing that you must do is make the following 7 simple shapes:



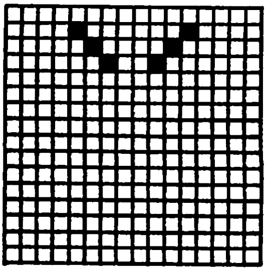
MS 10



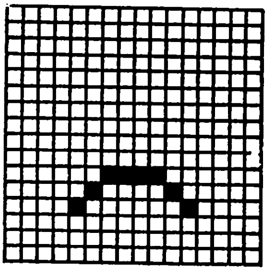
MS 11



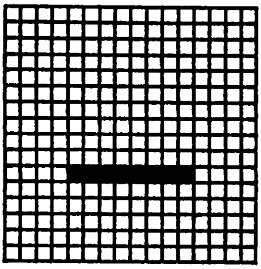
MS 12



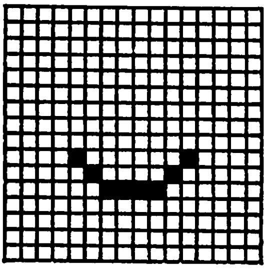
MS 13



MS 14



MS 15



MS 16

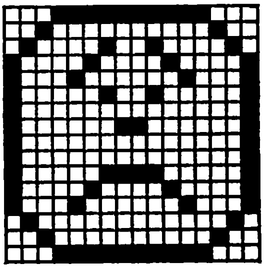
- You can try out faces once you have the shapes. Try this:

```
TELL [ 1 2 3 ]
SC 1
HOME
```

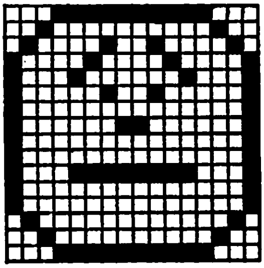
Now tell them to carry a face part:

```
TELL 1 CARRY 10
TELL 2 CARRY 11
TELL 3 CARRY 14
```

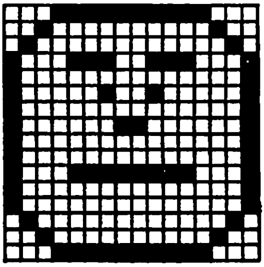
You see this face:



- You still have the attention of sprite 3. Tell it to carry shape 15.



- Tell sprite 2 to carry shape 12.



Here is a procedure that will make the face change its expression by itself:

```
TO EMOTE
TELL 1 CARRY 10
TELL 2 CARRY ( ( RANDOM + 1)/4 + 11) ☜ 11, 12, or 13
TELL 3 CARRY ( ( RANDOM + 1)/4 + 14) ☜ 14, 15, or 16
WAIT 120
EMOTE
END
```

When you tell computer to EMOTE it displays a face that changes its expression every two seconds.

Print Procedure Names

Do you remember the names of all the procedures that you have defined in this chapter? Let's see. There was START and PUNT and some others. The computer can print procedure names. Type this:

```
PP
```

If you have not turned your computer off, or pressed the (QUIT) key, then you will see something like this:

```
EMOTE
TALK
FLY
ROLL
ME
DOALOT
HI
START
```

The print procedures command PP helps you remember what procedures are available.

Print Out A Procedure

You can print out any procedure anytime you like. Just type this:

```
PO START
```

If the procedure START is still in memory it will be printed out like this:

```
TO START
CARRY :BALL SC 1 HOME
END
```

Remember that you typed in the procedure START using the DEFINE command like this:

```
DEFINE "START [ [ ] [ CARRY :BALL SC 1 HOME ]
```

The computer has arranged your definition in the form used by the procedure editor mode. Both definition methods yield the same final result.

Erasing Procedures

You can erase any procedure from the computer's memory. To erase the procedure ME, you type this:

```
ERASE ME
```

Now the procedure is gone. If you print the procedure names with the command PP, you will see that ME is gone. If you try to print out the procedure ME using the command PO, then nothing will print.

Summary of Chapter 3

This chapter was about defining procedures and about the editing tools available to you. In this chapter you learned:

- That a named list of commands is called a procedure
- That you can use the DEFINE command to define a procedure
- That typing the procedure's name causes the computer to perform the procedure
- That a procedure can use itself
- That the (FNCT)(BACK) key alias (FNCT)(9) gets back control of the computer, and gets back from edit mode
- That the word TO tells the computer to enter the procedure editing mode
- That the (FNCT) arrows can be used in edit mode to move the cursor anywhere in the procedure
- That letters may be inserted and deleted anywhere in a procedure while in edit mode
- That blank lines may be inserted using the (ENTER) key while in edit mode
- That blank lines may be deleted using (FNCT)(DEL) while in edit mode
- How to make animated faces using the MAKESHAPE command MS and the procedure editor
- How to print procedure names using the PP command
- How to print out a procedure using the PO command
- How to erase a procedure from memory using the ERASE command

Self-Test—Chapter 3

1. Fix all the mistakes in this definition:

```
DEFINE BURP   PRINT "GHAGH!
```

2. Tell what this procedure will do when BANG is typed:

```
DEFINE "BANG [ [ ] [ PRINT [ YOU GOT ME ] BANG ]
```

3. What key do you press to get back control when the computer is running a procedure?

4. What do you type to get into the procedure editing mode?

5. What happens if you are in procedure editing mode and the cursor is on the first letter of a line and you press the (ENTER) key?

6. How do you get out of the edit mode, into regular command mode?

7. If you are in the edit mode, with the cursor in the middle of a line, what happens when you type something?

8. What does the (FNCT)(DEL) key, alias (FNCT)(1), do in edit mode?

9. How do you delete a blank line when editing a procedure?

10. What is wrong with this procedure?

```
TO RECURSE
PRINT [ AGAIN AND ]
RECURSE

END
```

11. Write a procedure called BARK that will print GRUMP over and over again. Do it using both methods of definition.

Method 1: ____________________

Method 2: ____________________
____
____
____

12. What command causes the names of all the procedures to be printed?

____

Answers

1. DEFINE "BURP [[ ] [PRINT "GHAGH]]
2. When you type BANG, the computer finds the procedure called BANG and begins to perform the list of commands. First it prints BANG, then it goes to find the procedure BANG. It prints BANG and goes to find the procedure BANG, it prints and BANGs, again and again.
3. When you wish to get back control of the computer, you type the (FNCT)(BACK) key, alias (FNCT)(9).
4. The word TO tells the computer to enter the procedure definition mode.
5. In the edit mode, with the cursor on the first letter, the (ENTER) key causes the line to move down, leaving a blank line.
6. Press (FNCT)(BACK) to get back from edit mode.
7. If you are in edit mode and the cursor is in the middle of the line, then whatever you type is inserted in the line. The remaining part of the line shifts right to make space.
8. In edit mode the (FNCT)(DEL) key, alias (FNCT)(1), deletes whatever is under the cursor. The remaining letters of the line shift left to fill the space created.
9. To delete a blank line while editing a procedure, put the cursor at the beginning of the line, then press the (FNCT)(DEL) key.
10. RECURSE can use itself, but the call to use RECURSE must occur right before the END. The blank line must be deleted.
11. Method 1: DEFINE "BARK [[ ] [PRINT "GRUMP BARK]]
    Method 2:

```
TO BARK
PRINT "GRUMP
BARK
END
```

12. The print procedures command PP causes the computer to print out the names of all the procedures you have defined during your session.

4

Meet the Turtle

In the last two chapters you met the 32 sprites. In this chapter you will meet the turtle. There is only one turtle. The turtle is a computer creature that moves and draws for you. In this chapter you will learn:

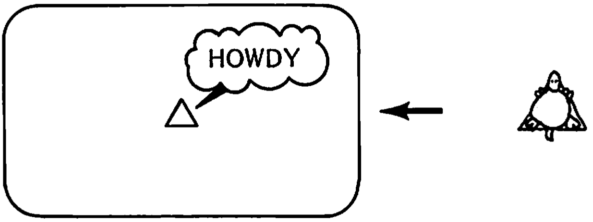
- How to get the turtle's attention
- That the turtle can move FORWARD and BACK
- That the turtle can turn RIGHT and LEFT
- That the turtle has a pen that can be put up and down
- How to set the color of the turtle's pen to any color of the rainbow
- How to make the turtle draw triangles, squares, stars, circles and other beautiful shapes
- How the turtle uses a screen map to get around the screen
- How to send the turtle to any point on the screen

Tell Turtle

You talk to the turtle just like you talk to sprites. You must get the turtle's attention. Type this:

```
TELL TURTLE          ☜ THIS GETS THE TURTLE
```

You see the turtle appear at the center of the screen:



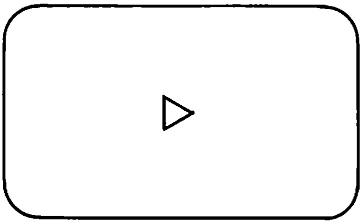
You still have the turtle's attention. The turtle pays attention to your commands until you tell something to some other computer creature like a sprite.

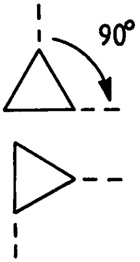
Turtle Turn

The turtle accepts the same commands as the sprites, except that you cannot set its speed. Give the turtle a quick turn to the right:

RT 90 ☜ RIGHT 90 DEGREES

The turtle turns 90 degrees to the right. Now the screen looks like this:





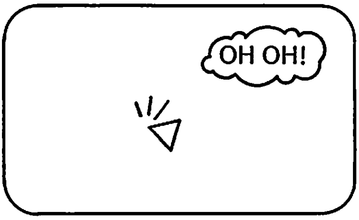
- Give the turtle some more right turns. Use your own turn numbers.
- Try a left turn, like this

LT 90 ☜ LEFT 90 DEGREES

Here is a program that spins the turtle:

```
REPEAT 100 [ RT 10 ]
```

You see the turtle spin like a top:



- What does the command RT 180 do?
- What does the command RT 360 do?
- What does the command RT -90 do?

ANSWERS:

RT 180 turns the turtle to face in the opposite direction.
RT 360 turns the turtle all the way around to face in the same direction as before.
RT -90 turns the turtle LEFT by 90 degrees. The minus sign (-) tells the turtle to turn counterclockwise.

- Turn the turtle so that its pointy face is headed towards the bottom of the screen.

Set Heading

Your turtle understands the SETHEADING command SH just like the sprites. You can see the turtle change the direction it is headed. Remember that SH 0 will always turn the sprite, or the turtle, to head towards the top of the screen. Try it.

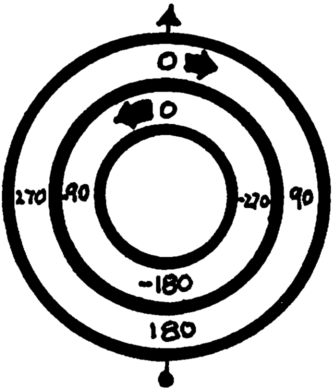
SH 0 ☜ SETHEADING 0

- What direction will the turtle head if you type the command SH 90?
- What direction will the turtle head if you type the command SH 180?
- What direction will the turtle head if you type the command SH -90?
- What direction will the turtle be headed if you type the command SH 360?

ANSWERS:

SH 90 heads the turtle towards the right side of the screen.
SH 180 heads the turtle towards the bottom of the screen.
SH -90 heads the turtle to the left side of the screen. SH 270 will have the same effect.
SH 360 heads the turtle toward the top of the screen. SH 360 has the same effect as SH 0.

Here is a picture of the compass used to set the headings of the sprites and the turtle:

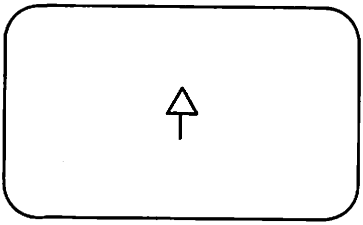


The heading compass always points up

Forward March

The clever turtle can go FORWARD and BACK just like the sprites. But, the turtle does more. The turtle has a pen and draws as it goes. Try this:

```
SH 0        SETHEADING 0
FD 20       FORWARD 20
```

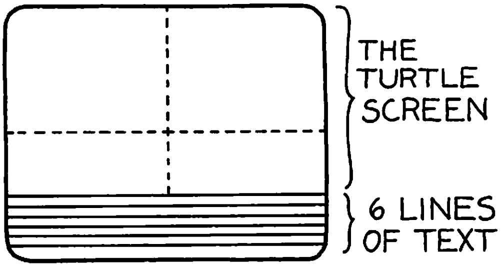


The turtle moves ahead 20 turtle steps, drawing in black as it goes. One turtle step is one TV dot.

- Move the turtle forward 20 more steps. Keep going. Make the turtle leave across the top edge of the screen and come back from the bottom. Did you notice anything strange? You can't see the turtle at the bottom lines of the screen.

The Turtle Screen

When the turtle is active, the screen acts in a special way. Six text lines are reserved at the bottom of the screen to display the commands that you type. The turtle screen starts just above the six text lines.



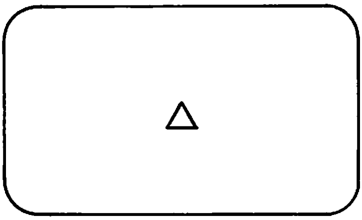
- Use the RT 90 command and the FD 20 command to draw a square.
- Fool around and experiment.

Clearscreen

The CLEARSCREEN command CS erases the turtle's drawing and sends the turtle home to the middle of the screen, pointing up. Give it a try:

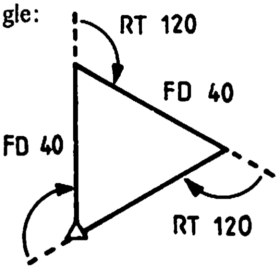
```
CS              ☜ CLEARSCREEN
```

You see a clear screen with the turtle at home in the middle:

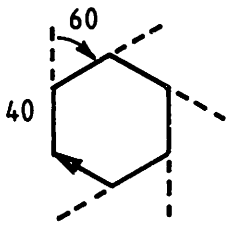


- Here is a command that draws a quick triangle:

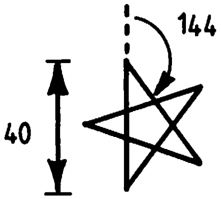
  REPEAT 3 [FD 40 RT 120]



- Make the turtle draw a hexagon like this:



- Make the turtle draw a star like this:



ANSWERS: There are many ways to make the turtle draw the hexagon and the star. Here is one way:

The hexagon: REPEAT 6 [FD 40 RT 60]
The star: REPEAT 6 [FD 40 RT 144]

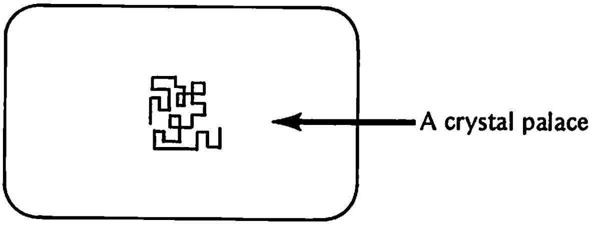
- Here is a command that makes the turtle draw a "circle":

```
CS
REPEAT 100 [ FD 8 RT 10 ]
```

- Here are some commands that make the turtle scurry around the screen building a beautiful crystal palace:

```
CS
REPEAT 500 [ FD 2*RANDOM RT 90*RANDOM ]
```

You see something like this:



Did your turtle run out of ink? The turtle cannot draw forever. It runs out of ink. The turtle draws on little square tiles the size of letters. The turtle has available exactly 192 square tiles to draw on. The turtle prints OUT OF INK if the drawing takes more than 192 tiles. The CLEARSCREEN command CS clears all the tiles for your next drawing.

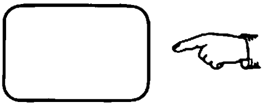
Hideturtle, Showturtle

The turtle can make itself invisible. Try this:

```
HT
```

or

```
HIDETURTLE
```

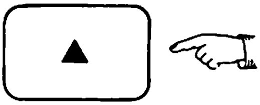


The turtle vanishes from the screen. To make the turtle visible again type this:

```
ST
```

or

```
SHOWTURTLE
```



The turtle reappears.

- Make the turtle appear and disappear like this:

```
REPEAT 20 [ HT WAIT 20 ST WAIT 20 ]
```

Pen Colors

The turtle can change its pen color. You still have the turtle's attention. Type the SETCOLOR command SC like this:

```
SC :WHITE
```

Now the turtle is using a white pen.

```
RT 90
FD 20
```

All of the 16 LOGO colors are available as pen colors.

```
SC :RED
FD 20
SC :YELLOW
FD 20
```

Here are some commands that make the turtle draw a line of many colors:

```
CS
RT 90
REPEAT 20 [ SC RANDOM FD 10]
```

The turtle cannot draw very far if it uses many colors. The turtle quickly runs out of ink.

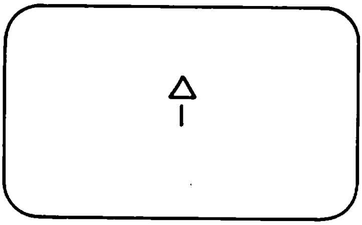
Some TV monitors display colors better than others. Thin lines of color do not always turn out the way you plan. Many TVs will display lines well in the horizontal direction, but the color will be "washed out" when drawn in the vertical direction. You will need to play around and experiment to see which colored lines display well on your TV monitor.

Pen Down, Pen Up

The turtle can pick its pen up and stop drawing. The PEN UP command is PU. Here is an example to try:

```
CS
FD 5
PU ☜ PEN UP
FD 5
```

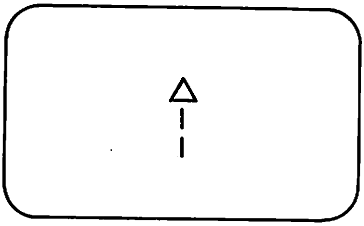
You see:



The turtle still moves when its pen is up, but it doesn't draw as it goes. When you want the turtle to draw again, you give the PEN DOWN command PD. Give this a try:

```
CS
PD FD 5 PU FD 5 PD FD 5
```

You see a totted line like this:



Here is a genuine dotted line maker to try:

```
CS
RT 60
REPEAT 300 [ PD FD 5 PU FD 5 ]
```

Pen Erase

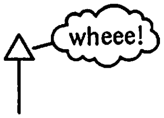
Every pen should have an eraser. The turtle's pen can act like an eraser. The PEN ERASE command is PE. Watch this one work:

```
PD
FD 50
PE      ☜ PEN ERASE
BK 50   ☜ Back over the previous line
```

You see the turtle draw a line, then back up and erase it.

Here is another example. In this example the turtle repeatedly draws a line, then erases the line.

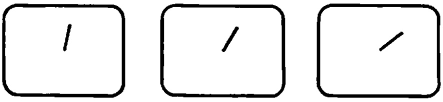
```
REPEAT 50 [ PD FD 60 PE BK 60 ]
```



It looks like a turtle practicing jumping.

This next example makes the turtle draw a clock with a moving hand:

```
CS
REPEAT 200 [ RT 12 PD FD 30 PE BK 30 ]
```



. . . etc.

Pen Reverse

The turtle has one more type of pen. This pen is very special. It reverses light and dark wherever it draws. The REVERSE pen draws dark over light areas and draws light over dark areas. The next example uses the PEN REVERSE command PR to make the turtle draw and then undraw a square:

```
PR  ☜ PEN REVERSE
REPEAT 80 [ FD 50 RT 90 ]
```

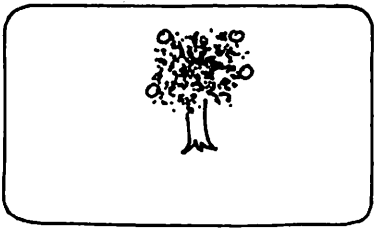
The square is drawn, then redrawn with the reversing pen. But that erases it.

Here is a command that draws at random using the reverse pen:

```
PR
REPEAT 200 [ FD 50 RT RANDOM + 180 ]
```

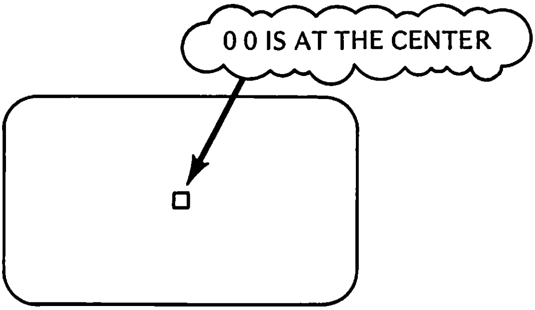
Notice that the pen draws dark over light areas and prints light over dark areas. This last drawing looks a lot like the leaves of a tree or a bush.

- Use the SC :GREEN command, then use the last command to redraw the tree. Add a trunk and some red apples to the tree.



The Screen Map

The turtle and the sprites know how to move to any place you tell them on the screen. Each dot on the screen has an address. The address consists of two numbers. The address of the dot right in the center of the screen is 0 0.



Dots

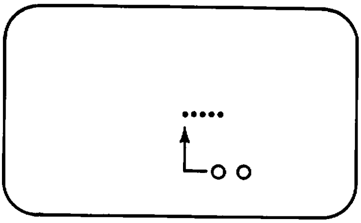
The turtle can make a dot at any place on the screen. Let's hide the turtle and make some dots:

HT ☜ HIDE TURTLE

Be sure to put spaces here

```
DOT 0 0
DOT 1 0
DOT 2 0
DOT 3 0
DOT 4 0
```

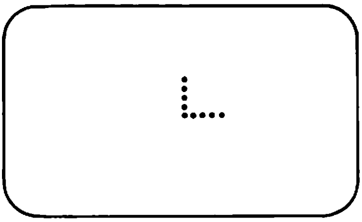
Here is what you see:



Now try this:

```
DOT 0 1
DOT 0 2
DOT 0 3
DOT 0 4
```

You see this:



- Where will this dot go?

```
DOT 4 4
```

04				
03				
02	12	22		
01	11	21		
00	10	20	30	40

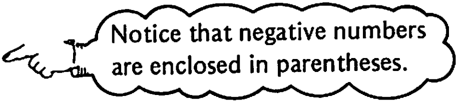
- Find how far out you can go to the right. Try DOT 100 0. Can you make a dot farther out?
- Find how far up you can go towards the top. Try DOT 0 50. Can you make a dot farther out?

ANSWERS:
DOT 0 119 is the furthest to the right
DOT 96 0 is the furthest up.

Minus Dots

You can make dots to the left and towards the bottom. Here is a dot near the left side of the screen:

```
DOT (-119) 0
```

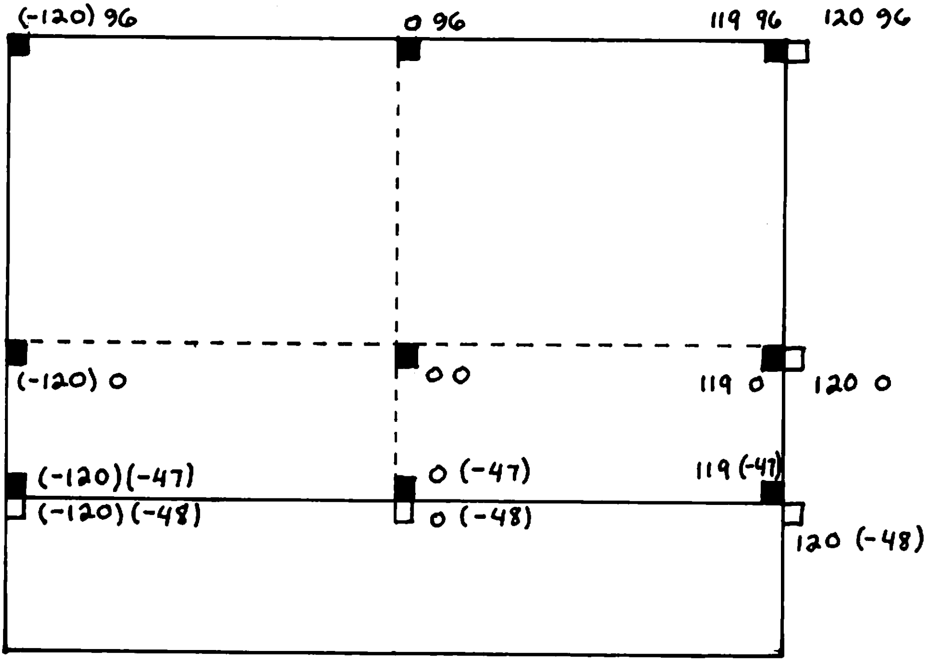


Notice that negative numbers are enclosed in parentheses.

The minus sign (-) tells the turtle to go to the left.

Here is the dot at the bottom of the turtle screen:

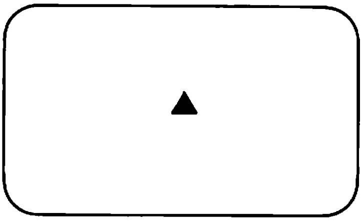
```
DOT 0 (-47)
```



There is an important thing you should know about turtle dots. The turtle doesn't move to make the dots. You can see this for yourself. Get the turtle's attention and try this:

```
HOME
ST      show the turtle
```

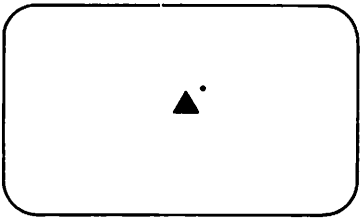
You see the turtle appear in the middle of the screen.



Now tell the turtle to make a dot like this:

```
DOT 10 10
```

You see that the turtle stays in the center of the screen while the dot is made at 10 10.

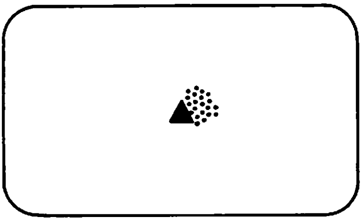


Lots A Dots

Here is a short procedure that makes a lot of dots:

```
TO DOTS
TELL TURTLE
PEN REVERSE
DOT RANDOM RANDOM
DOTS
END
```

Press the (FCTN)(BACK) key, alias (FCTN)(9), to get back to LOGO command mode. Now type DOTS. You see a constantly changing pattern of dots in a small square near the center.



Press (FCTN)(BACK) when you wish to go on.

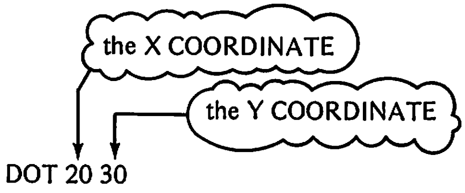
- Change line 3 of the procedure DOTS to this:

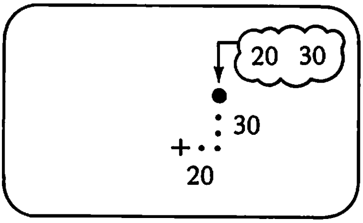
```
DOT 10*RANDOM 10*RANDOM
```

Now DOTS makes a big array of dots that flicker on and off.

X Direction Y Direction

There is a long and venerable tradition for naming the pair of position numbers used by the turtle. The first number is called the X COORDINATE. The second number is called the Y COORDINATE.

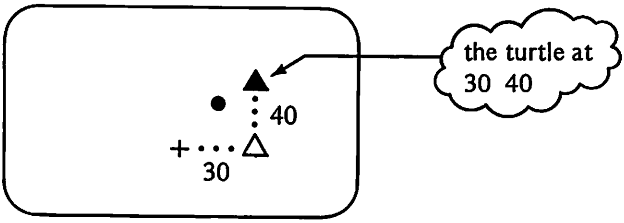




You can set the X coordinate of the turtle with the SET X command SX. You can set the Y coordinate of the turtle with the SET Y command SY. Here is how they work. Give this a try:

ST	show turtle
HOME	set the turtle's position at 0 0.
SX 30	set the turtle's X coordinate to 30
SY 40	set the turtle's Y coordinate to 40

First you see the turtle move to the right to the position 30 0. Next you see the turtle move up to the position 30 40.

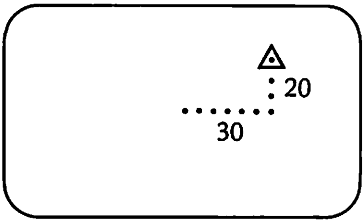


Set the XY Coordinates

You can send the turtle immediately to any position on the screen using the SET XY command SXY. Try this:

```
ST ☜ SHOW TURTLE
SXY 30 20 ☜ SET THE XY COORDINATES TO 30 and 20
```

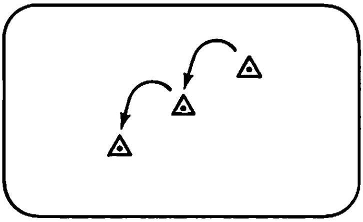
You see the turtle move immediately to the position 30 20



- Give this a try:

```
SXY 20 20
SXY 0 0
SXY (-20) (-20)
```

You should see something like this:



Nervous Turtle

You can use the SETXY command SXY to make the turtle move at random. Give this procedure a try:

```
TO FLIT
TELL TURTLE
REPEAT 100 [ SXY RANDOM RANDOM ]
END
```

Get back to LOGO command mode and type FLIT. You see the turtle flit nervously in a square region near the center of the screen.

- Give the turtle a bit more freedom. Change the second line of the procedure to this:

```
REPEAT 100 [ SXY 10*RANDOM 10*RANDOM ]
```

- Make the turtle do some work. Change the second line to this:

```
REPEAT 100 [ SXY 10*RANDOM 10*RANDOM FD 1 ]
```

- Here is more work for the turtle:

```
REPEAT 100 [ SXY 10*RANDOM 10*RANDOM RT RANDOM
FD 16 ]
```

XCOR and YCOR

The computer always knows the present address of the turtle. You can find out the address numbers whenever you want them. The computer stores the X coordinate under the name XCOR. The computer stores the Y coordinate under the name YCOR. Let's set the X and Y coordinates and see if XCOR and YCOR work as advertised. Get the turtle's attention and type this:

```
SXY 40 50 ☜ set the X and Y coordinates of the turtle
PRINT XCOR ☜ print the X coordinate number
```

You see

```
40 ☜ the X coordinate of the turtle
```

Now type

```
PRINT YCOR ☜ print the Y coordinate number
```

You see

```
50 ☜ the Y coordinate of the turtle
```

Hide and Go Seek

Here is a game. We will hide the turtle and then find it again and bring it home. To hide the turtle, type this:

```
HT ☜ hide turtle
PU ☜ pen up
REPEAT 20 [ FD RANDOM RT RANDOM ] ☜ lose the turtle
```

Now it's your turn. Get the turtle back to 0 0. You can only use the commands PRINT, XCOR, YCOR, SXY, SX and SY.

Turtle Speed

Here is a procedure that uses YCOR to set a new Y coordinate.

```
TO MOVET
CS ☜ clear the screen
                        set the new Y coordinate to YCOR + 10
REPEAT 100 [ SY YCOR+10 WAIT 10 ]
END
```

Get back to LOGO command mode and type MOVET. The screen clears and you see the turtle move up the screen. The computer changes the Y coordinate to its present value YCOR plus 10. The turtle moves up the screen 10 steps.

- Change MOVET to make the turtle move in the X coordinate direction.
- Change MOVET so that both the X and the Y coordinates change.
- Change MOVET so that the turtle moves in the X coordinate direction, and the heading depends on XCOR.

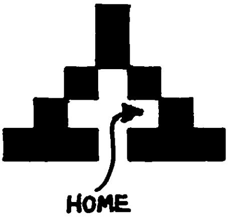
Here are the changes that we made. Yours could be different:

```
REPEAT 100 [ SX XCOR + 10 ]
REPEAT 100 [ SX XCOR + 10 SY YCOR + 20 ]
REPEAT 100 [ SX XCOR + 10 SH XCOR ]
```

The Turtle and the Sprites

The home position for the turtle, and for the sprites is at 0 0, the center of the screen. You need to know what part of the turtle is over the center, and what part of the sprite is over the center. You can easily see. Do these commands:

```
TELL TURTLE
HOME
TYPE XCOR TYPE YCOR
```



You see

```
00
```

The home of the turtle is position 0 0. Now put a dot at 0 0:

```
PD          pen down
HT          hide turtle
DOT 0 0
```

The dot shows the turtle's home at 0 0. Now show the turtle:

```
ST
```

You see this:

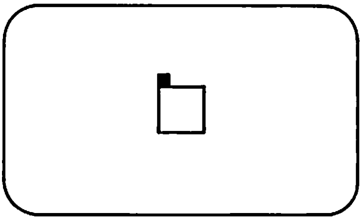
- Flash the turtle on and off with this command:

```
REPEAT 50 [ ST WAIT 20 HT WAIT 20 ]
```

Now hide the turtle for the moment. Put a box at home position:

```
HT
TELL 1
CARRY :BOX
SC :WHITE
HOME
```

You see a white box appear like this:



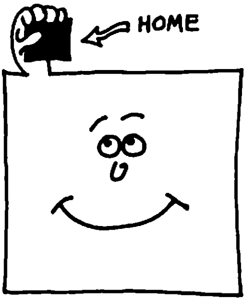
Check if the box is at 0 0:

```
TYPE XCOR TYPE YCOR
```

You see

```
00?_
```

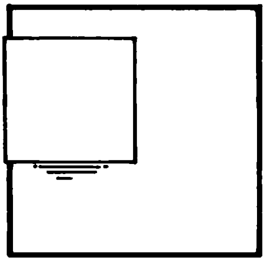
The upper left corner of the sprite's box is just below the dot at 0 0. The sprite rests its upper left corner against the dot at 0 0.



- Here is a procedure that makes the turtle draw a square, then makes the sprite travel around the square:

```
TO FOLLOW
CS
TELL TURTLE
HOME
REPEAT 4 [ FD 32 RT 90 ]
TELL 1
CARRY :BOX
SC :WHITE
HOME
REPEAT 100 [ FD 32 RT 90 WAIT 20 ]
END
```

Press the (FNCT)(BACK) key and then type FOLLOW. You see the turtle draw a square, then the white sprite box move around the corners of the square.



- Change the procedure so that the sprite moves inside the turtle's square.

Summary of Chapter 4

In this chapter you learned:

- That the command TELL TURTLE gets the turtle's attention
- That the right turn command RT 90 turns the turtle right by 90 degrees
- That the left turn command LT 90 turns the turtle left by 90 degrees
- That the set heading command SH 0 sets the turtle's heading north, toward the top of the screen
- That the forward command FD 20 moves the turtle ahead 20 steps
- That one turtle step is one TV dot

- That the back command BK 20 moves the turtle backward 20 steps
- That the pen up command PU raises the turtle's pen; the pen down command PD lowers the turtle's pen
- That the clearscreen command CS clears the screen and sends the turtle to the center, headed up
- How to draw triangles, squares, hexagons, stars and circles
- That the turtle draws on 192 letter sized tiles
- That the hideturtle command HT makes the turtle invisible, the showturtle command ST makes the turtle visible again
- That the setcolor command SC :RED sets the turtle's pen color red
- That the pen erase command PE changes the turtle's pen to an eraser, the pen reverse command PR changes the turtle's pen to a color reversor
- That each point on the screen has a two-number address
- That the two screen address numbers are called the X coordinate and the Y coordinate
- That the turtle's position is stored under the pair of names XCOR and YCOR
- That the dot command DOT 20 30 puts a dot on the screen at the point with X coordinate 20 and Y coordinate 30
- That the set XY command SXY 20 30 sends the turtle to the point on the screen with X coordinate 20 and Y coordinate 30

Self-Test—Chapter 4

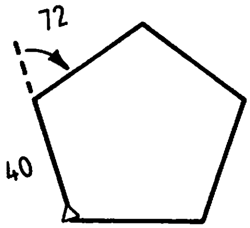
1. Write the command that gets the turtle's attention.

   ____

2. Write the command that turns the turtle around to face in the opposite direction. ______________________________
3. Write the command that turns the turtle towards the top of the screen.

   ____

4. Write a command to draw this pentagon:



   ____

5. Write the commands to draw a red line 50 units long, then erase it.

   ____

6. Write a command that puts a single dot at the upper right corner of the screen: ______________________________

7. What will these commands print?

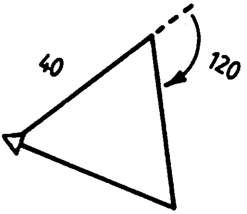
```
SH 0
SXY 30 50
FD 5
PRINT XCOR
PRINT YCOR
```

8. Write a procedure called TRI that draws a triangle with sides of length 40, at the turtle's position.

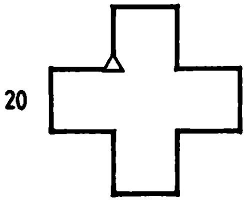
   ____

   ____

   ____



9. Write a procedure called CROSS that causes the turtle to draw this figure:



   ____

   ____

   ____

10. Write a procedure called BORDER that draws a black border around the edge of the screen.

   ____

   ____

   ____

   ____

   ____

Answers

1. The command TELL TURTLE gets the turtle's attention.
2. Any of these commands will turn the turtle in the opposite direction:

```
RT 180 or RT (-180) or LT 180 or LT (-180)
```

3. The command SH 0 will head the turtle towards the top of the screen.
4. Here is one way to draw a pentagon. There are many other ways that are ok.

```
REPEAT 5 [ FD 40 RT 72 ]
```

5. These commands draw a red line and then erase it:

```
TELL TURTLE
SC :RED PD FD 50 PE BK 50
```

6. The command DOT 119 96 puts a dot at the upper right hand corner of the screen.
7. The command SH 0 sets the turtle heading towards the top. SXY 30 50 sets the turtle's X and Y coordinates to be 30 and 50 respectively. FD 5 moves the turtle forward, up towards the top. This changes the turtle's Y coordinate from 50 to 55. The computer prints the numbers 30 and 55.
8. Here is just one of many possible ways to make a triangle:

```
TO TRI
REPEAT 3 [ FD 40 RT 120 ]
END
```

9. Here is one possible way to draw the cross shape:

```
TO CROSS
REPEAT 4 [ FD 20 RT 90 RT 90 LT 90 ]
END
```

10. Here is one possible way to draw a border around the screen:

```
TO BORDER
TELL TURTLE
SC :BLACK PD
SXY 119 96 SH (-90)
REPEAT 2 [ FD 240 LT 90 FD 144 LT 90 ]
END
```

5

"Name, :Name, Name, [Name]

Your faithful computer can accept what you type, check it, name it and store it. In this chapter you will learn:

- The difference between the word "RED, the number :RED, the procedure RED and the one word list [RED]
- How to MAKE names for words and lists
- How to use the READCHARACTER command RC to read one character
- How the IF . . THEN command is used to make decisions
- How to use the READLINE command RL to read a whole line
- How to make the computer act smart and call you by name
- How the computer edits lists into a standard form
- How to use the computer to write quick letters
- How to write a computer story that lets the reader make decisions
- How RC? becomes TRUE when a key is pressed

"Red Is A Word

You know all about the color red. But, do you know about "RED? "RED has three letters. "RED is a three-letter word. The quote mark (") is used in LOGO to remind you that the word itself is meant. Type this:

```
PRINT "RED
```

The computer prints

```
RED
```

☜ the computer took off the " mark

:Red Is A Thing

You may remember from Chapter 1 that :RED stands for the number 6. When you put in the LOGO cartridge, your computer makes the word :RED stand for the number 6. :RED is 6. The colon mark (:) reminds you that the object (number 6) is meant rather than the word itself.

Type this:

```
PRINT :RED
```

The computer prints the object for which :RED stands:

```
6
```

Red Is A Procedure

You can DEFINE a procedure called RED. Type this:

```
DEFINE "RED [ [] [PRINT "READY ] ]
```

You could also use the procedure editor to define the procedure RED. Now, type

```
RED
```

The computer gets the procedure RED and runs it. You see

```
READY
```

[Red] Is A List

By putting square brackets around RED you can make a one-word list:

```
[RED ]
```

Type this:

```
PRINT [RED ]
```

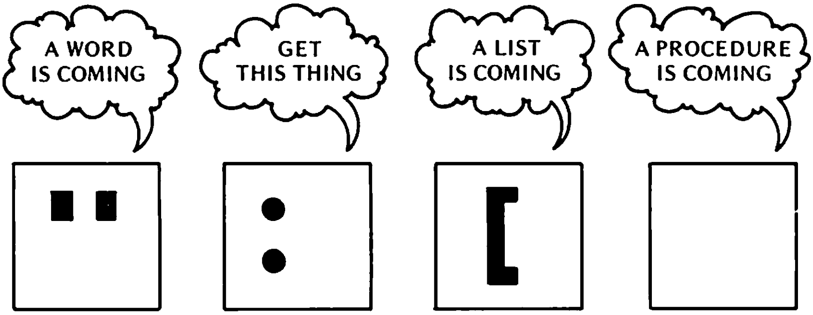
Your computer prints the contents of the list:

```
RED
```

''Red, :Red, Red, (Red)

Now we have four different reds:

"RED	THE WORD
:RED	THE NUMBER 6
RED	THE PROCEDURE COMMAND RED
[RED]	THE ONE-WORD LIST



A Short Quiz

Put "RED, :RED, RED, or [RED] in the proper space:

- _____ is the number 6.
- _____ has three letters.
- _____ is a list.
- _____ is a command.

ANSWERS: :RED, "RED, [RED], RED

Making Names for Things

You can make names for things in LOGO. You already know that :RED stands for the number 6. Here is how you make a word the name for another word. We'll make the word "SLOWEST the name of the word "GRESNELDA. Type this:

```
MAKE "SLOWEST "GRESNELDA
```

☜ "SLOWEST stands for "GRESNELDA

LOGO has now stored the word "GRESNELDA under the name "SLOWEST. To see what the name "SLOWEST stands for type this:

```
PRINT :SLOWEST
```

the colon mark (:) tells the computer to use what "SLOWEST names

The computer knows that the word "SLOWEST is the name for the word "GRESNELDA. The computer prints

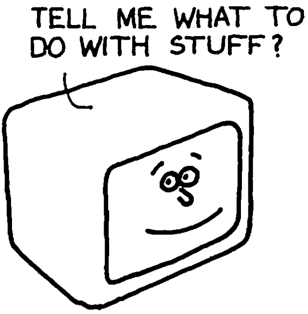
```
GRESNELDA
```

Here is an example in which the name "STUFF is given to the list [BACON AND EGGS]. Type this:

```
MAKE "STUFF [BACON AND EGGS ]
```

The computer has stored the list of words [BACON AND EGGS] under the name "STUFF. To check that the computer really did store the list, type this:

```
"STUFF
```



The computer finds the list [BACON AND EGGS], but doesn't know what to do with it. You see this message:

```
TELL ME WHAT TO DO WITH [BACON AND EGGS ]
```

Type this:

```
PRINT :STUFF
```

Now the computer finds the list [BACON AND EGGS] and prints it like this:

```
BACON AND EGGS
```

the words in the list without brackets

Notice that the PRINT command takes off the first and last brackets before it prints a list.

Try printing some other variations of STUFF. Type this:

```
PRINT "STUFF ☜ the word
```

The quote mark (") tells the computer that the word STUFF itself is meant. You see

```
STUFF ☜ the letters of the word "STUFF
```

Notice that the PRINT command takes off the quote mark (") and just prints the word itself.

Try this:

```
PRINT STUFF ☜ a procedure call
```

Since there is no quote mark (") or colon mark (:) or bracket ([), the computer tries to find a procedure called STUFF. It can't find anything. It prints this message:

```
TELL ME HOW TO STUFF
```

You can make a procedure called STUFF like this:

```
TO STUFF
REPEAT 100 [TYPE "STUFF ]
END
```

Now when you get back to command mode and type STUFF you see the screen fill with

```
STUFFSTUFFSTUFFSTUFF ...
```

Here is one last one to try:

```
PRINT [STUFF ]
```

The computer finds the bracket ([) and knows that the list itself is to be printed. The computer prints until it finds the other bracket (]) that goes with the first bracket. You see

```
STUFF
```

The computer removes the first and the last bracket before printing.

- Tell what the PRINT commands will print out:

```
MAKE "X "XTRA
MAKE "Y [WHY NOT ]
DEFINE "Z [ [] [ PRINT "ZEBRA ] ]
PRINT :X ____________________
PRINT :Y ____________________
PRINT [X ] __________________
PRINT "X ____________________
Z ___________________________
PRINT [X Y Z ] ______________
```

ANSWERS: XTRA, WHY NOT, X, X, ZEBRA, X Y Z.

Make and Call

There are two very similar commands in LOGO that you can use to assign names to things. You have already learned about the MAKE command. You can also use the CALL command for the same purpose. The MAKE and the CALL commands do the same thing, but in a different order. Here is an example. This example assigns the name "STUFF to the list [BACON AND EGGS] using MAKE and CALL to ~~accomplish the same~~ thing:

```
MAKE "STUFF [EGGS AND BACON ] ☜ NAME FIRST,
                                THING SECOND
CALL [EGGS AND BACON ] "STUFF ☜ THING FIRST,
                                NAME SECOND
```

You can MAKE a name for a thing, or you can CALL a thing by a name. You can use either MAKE or CALL to name things. Use the one you like best. We will stick with MAKE in this book, just to keep things simple.

Read a Character

An alert computer should pay attention when you touch its keys. The READCHARACTER command RC tells the computer to read the next letter or other character that you type. Give this a try:

```
RC (ENTER)
```

The computer stops and waits for you to press some key. Press the (C) key. You see

```
TELL ME WHAT TO DO WITH C
```

The command RC tells the computer to READ a CHARACTER from the keyboard. RC is the command, and also the name of the character.

Give this a try:

```
PRINT RC
```

the computer reads the next key you press and puts it here

The computer stops and waits for you to press a key. Press C. You see

```
C
```

The computer prints the character that it reads from the keyboard. Try this:

```
REPEAT 5 [ PRINT RC ]  ☜ print what is read
```

The computer waits for you to type some characters. Type G H I J K. You see

```
G
H
I
J
K
```

the computer reads and prints the five characters

The computer repeatedly prints what RC brings back from the keyboard.

- Here is a command that types each letter you press, but in an odd way:

```
REPEAT 50 [ TYPE "( TYPE RC TYPE ") ]
```

Every letter you type is now in parentheses.

Do A Lot

Here is a procedure that does a lot. This procedure reads a character from the keyboard, then types it 1000 times. Give it a try:

```
TO DOALOT
MAKE "KEY RC ☜ "KEY is the name of the key that is read
REPEAT 1000 [TYPE :KEY ]  ☜ type the key 1000 times
DOALOT  ☜ do it again
END
```

Get back to LOGO command mode and type DOALOT. The computer finds that command MAKE "KEY RC. The computer waits for you to type a key. Type some key, say an (A) key. The computer fills the screen with the letter A. Each time you press a key the computer fills the screen with 1000 copies of that key.

- Change the REPEAT line in the last procedure so that it will PRINT the key 100 times.

If . . . Then . . .

You can change DOALOT so that it only responds when you type a particular key. This requires a new and important command called the IF . . THEN command. Get into procedure editing mode by typing TO DOALOT. Now change DOALOT so that it looks like this:

```
TO DOALOT
MAKE "KEY RC
IF ( :KEY = "B ) THEN PRINT :KEY ☜ if the key is B, print it
DOALOT
END
```

This procedure gives the name "KEY to the character read from the keyboard. If the key is B, then the key is printed. Use the (FNCT)(BACK) key to stop this procedure.

- Change DOALOT so that it looks like this:

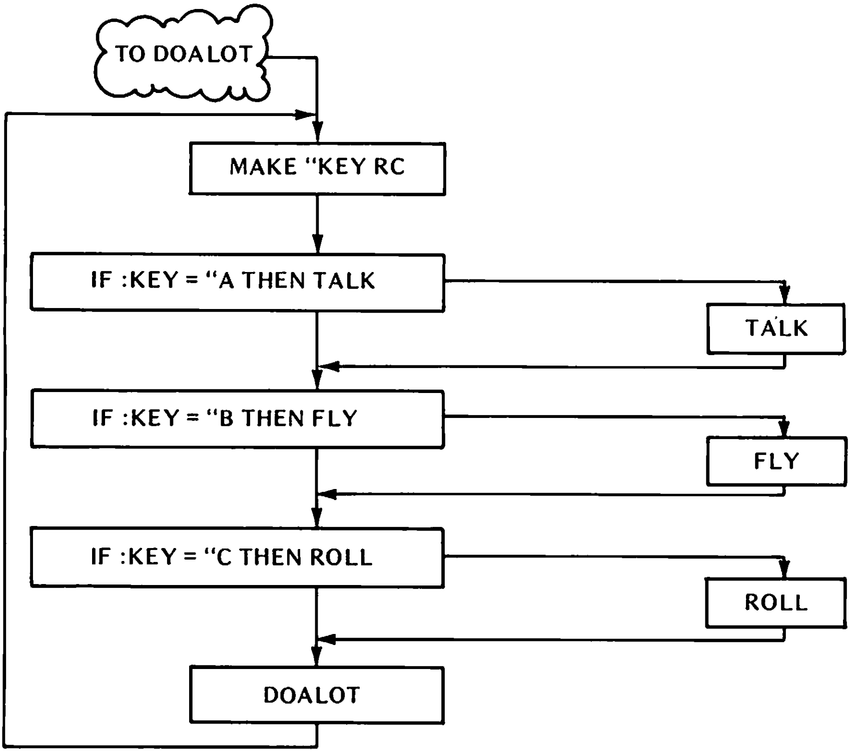
```
TO DOALOT
MAKE "KEY RC
IF :KEY = "B THEN PRINT :KEY
IF :KEY = "T THEN PRINT [ I LIKE TEA ]
DOALOT
END
```

- Make some changes of your own. You can make each key do something different. One key can change the screen color, another can beep 20 times, another can start a sprite. Page 89 shows a flow chart of one possibility that uses other procedures.

Of course, you must create the procedures TALK, FLY and ROLL. You can use the procedures from Chapter 3, if you like.

Read A Line

Your TI99 computer can do more than read a single character from the keyboard. It can read a whole line. The READLINE command RL tells the computer to read a whole line from the keyboard. The computer reads characters until you press the (ENTER) key. Give this a try:



```
RL
```

You see this:

```
>_
```

Notice how the computer tells you it is expecting a whole line. It prompts you with (>) instead of (?). Type some keys and press (ENTER). We typed THIS AND THAT. Here is what we saw:

```
>THIS AND THAT
TELL ME WHAT TO DO WITH [THIS AND THAT ]
?_
```

Notice that the computer stored the line of words you typed as a list with brackets at either end.

Try this:

```
PRINT RL
```

You see

```
>_
```

The computer is waiting for your line. Type THIS AND THAT (ENTER). You see this:

```
>THIS AND THAT  ☜ you
THIS AND THAT  ☜  the computer
?_
```

100 Lines

Here is a short procedure that takes the line you type and prints it 100 times:

```
TO REP
MAKE "INPUT RL  ☜ INPUT is the name of the line
REPEAT 100 [PRINT :INPUT ]
END
```

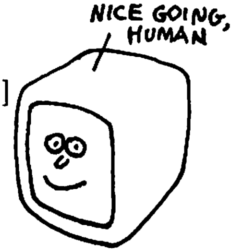
Now get back to command mode and call the procedure by typing

```
REP
```

The computer prompts with a (>). Type your name and press (ENTER). One hundred copies of your name appear on the screen.

- Change the REPEAT command in REP to this:

```
REPEAT 100 [TYPE [NICE GOING ] PRINT :NAME ]
```



A Well-Mannered Machine

A well-mannered computer should at least ask your name and say hello when you say HI. Here is a procedure called HI that does the job:

```
TO HI
PRINT [HI, WHAT'S YOUR NAME? ]
MAKE "NAME RL
TYPE [PLEASED TO MEET YOU ]
PRINT :NAME
END
```

Get back to command mode and type

```
HI
```

The computer responds

```
HI, WHAT'S YOUR NAME?  ☜ the computer
>_
```

We typed

```
BERTHA BLOT
```

The computer typed

```
PLEASED TO MEET YOUBERTHA BLOT  ☜ the computer
```

How the Computer Edits Lists

Did you notice that the computer put YOU and BERTHA together? The computer always tidies up the lists you give it. You need to know what the computer does to your lists. Here is a list with many spaces:

```
[______ABC______DEFG_________HIJK______________]
```

The computer edits your list and stores this:

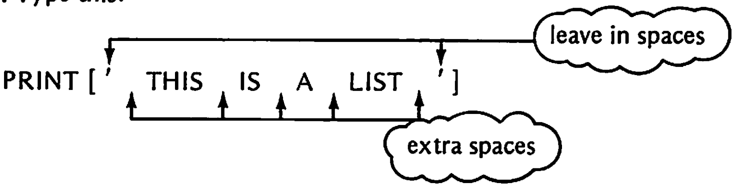
```
[ABC DEFG HIJK ]
```

- exactly one space here, but never printed
- extra spaces out, one left
- no space here in front

The computer removes all beginning spaces. If it finds two or more spaces it removes all but one. The computer puts exactly one space at the end of the list. That extra space at the end won't be printed.

Space Case

You can tell the computer not to edit a list. The apostrophe mark (') signals the computer to stop its editing and leave in extra spaces. The next apostrophe mark tells the computer to start editing again. Here is an example to try. Type this:



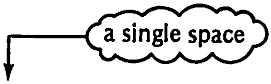
```
PRINT [' THIS  IS  A  LIST  ']
```

You see

```
____THIS____IS____A____LIST____
```

spaces are left in

Here is a short procedure called SP that lets you make spaces easily:



```
TO SP
TYPE [ ' ' ]
END
```

☜ (') turns the editor on and off

The apostrophe mark (') tells the computer not to take out spaces. While the procedure SP is in your computer's memory you can get a space with the command SP.

- Change the procedure HI so the third line looks like this:

```
TYPE [PLEASED TO MEET YOU ] SP
```

☜ the space maker

Now HI spaces the words properly.

```
PLEASED TO MEET YOU BERTHA BLOT
```

- Try these to see how the apostrophe keeps spaces in words and lists

```
PRINT "THIS AND THAT
PRINT " 'THIS AND THAT'
PRINT [   ' THIS AND THAT ' AND     MORE ]
```

Here is what you see:

THIS	the words AND THAT are ignored
THIS_AND_THAT	the (') keeps the spaces
_THIS_AND___THAT__AND_MORE	some space in, some out

Form Letter

Have you sent your thank-you letters yet for all those nifty gifts you got last Christmas? This next program will help you a lot. It is a letter maker that helps you write letters quickly. This procedure will use the space procedure SP. Make sure SP is still stored.

```
TO LETTER
```

```
PRINT [NAME? ]
MAKE "NAME RL
PRINT [GIFT? ]
MAKE "GIFT RL
PRINT [ADJECTIVE? ]
MAKE "ADJECTIVE RL

PRINT [ ] ☜ a blank line
TYPE [DEAR ] SP PRINT :NAME
PRINT [ ]
TYPE [THANKS FOR THE ] SP TYPE :ADJECTIVE SP PRINT :GIFT
PRINT [I ENJOY YOUR GIFT A LOT.]
SP SP SP SP PRINT [YOURS TRULY,]
SP SP SP SP PRINT [THURSTON ] ☜ your name here
END
```

Now get back to command mode and type LETTER to call the procedure. The computer will ask you for the name of the person who gets the letter, the name of the gift they sent, and an adjective to describe the gift. Here is what we typed and the letter that the computer printed:

We typed

```
LETTER
```

The computer typed

```
NAME?
>
```

We typed AUNT POOTY

The computer typed

```
GIFT?
>
```

We typed PARSNIP PEELER

The computer typed

```
ADJECTIVE?
>
```

We typed CUNNING

The computer typed out this letter:

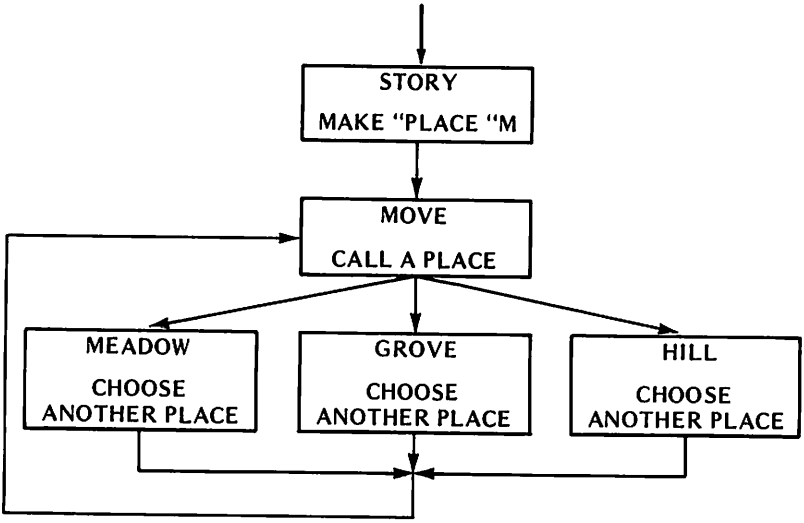
```
DEAR AUNT POOTY

THANKS FOR THE CUNNING PARSNIP PEELER
I ENJOY YOUR GIFT A LOT.
               YOURS,
               THURSTON
```

You can add your own lines to LETTER. There are lots of possibilities here: love letters, angry letters, funny letters, crazy letters.

An Interactive Story

This next program is a special kind of story that lets the reader make decisions. The program consists of five procedures that work together. The first procedure is STORY. STORY starts everything off by asking you where you wish to go. STORY passes the information on to MOVE. MOVE takes you where you want to go. The places are described and controlled by MEADOW, GROVE and HILL. At first the places will only ask you where you want to go and then pass you back to MOVE. Later you can add lines to make the places more interesting.

Here is a flow chart of the program:



```
TO STORY
PRINT [WHERE DO YOU WANT TO GO?]
PRINT [(M)EADOW, (G)ROVE, (H)ILL ]
MAKE "PLACE RC
MOVE
END

TO MOVE
IF :PLACE = "M THEN MEADOW
IF :PLACE = "G THEN GROVE
IF :PLACE = "H THEN HILL
MOVE
END

TO MEADOW
PRINT [YOU ARE IN A MEADOW ]
PRINT [(G)ROVE OR (H)ILL? ]
MAKE "PLACE RC
END

TO GROVE
PRINT [YOU ARE IN A GROVE]
PRINT [(M)EADOW OR (H)ILL? ]
MAKE "PLACE RC
END

TO HILL
PRINT [YOU ARE ON A HILL ]
PRINT [(M)EADOW OR (G)ROVE? ]
MAKE "PLACE RC
END
```

To start the story, type STORY. Here is what you see:

```
WHERE DO YOU WANT TO GO?
(M)EADOW, (G)ROVE, (H)ILL
```

We typed H and saw

```
YOU ARE ON A HILL
(M)EADOW OR (G)ROVE?
```

We typed G and saw

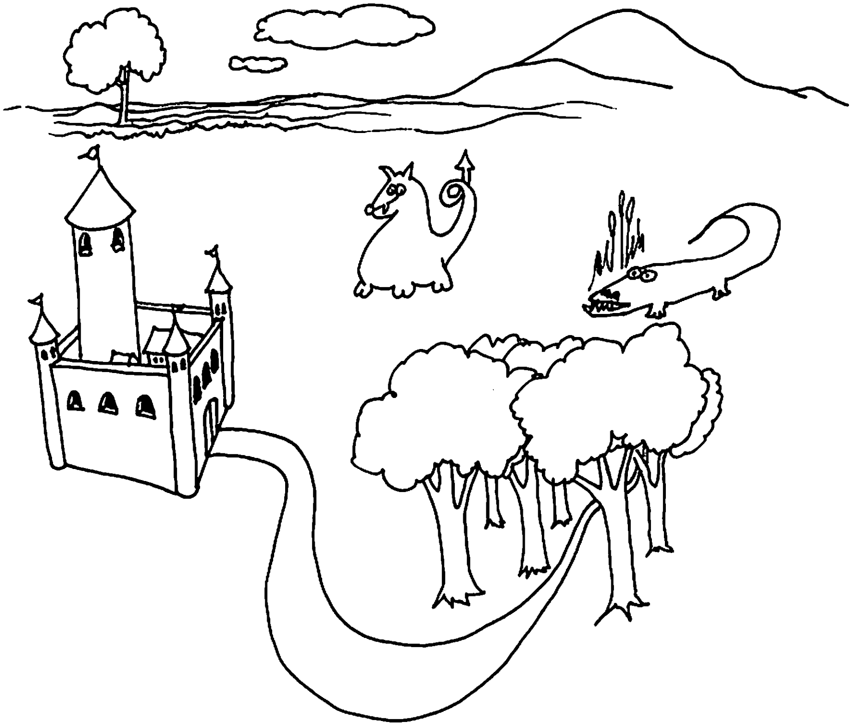
```
YOU ARE IN A GROVE
(M)EADOW OR (H)ILL?
```

This is not the most exciting story ever written. We kept it as simple as possible to show how it's done. It would be a more interesting story if something unexpected or surprising happened. We changed MEADOW so that it printed this:

```
YOU ARE IN A GRASSY, SUNLIT MEADOW. YOU SEE A SHADY
GROVE TO THE NORTH AND A LOW, ROUNDED HILL TO THE
SOUTH. WHICH WAY WOULD YOU LIKE TO GO? (G)ROVE OR
(H)ILL?
```

You can make up a more exciting story than ours.

- Change MEADOW so that it prints something more exciting.
- Change HILL so that a dragon appears and asks your name.
- Change MOVE to make some new places. How about a castle with a dungeon and a dragon or a troll with a treasure? Make a creature that asks your name and then uses it in the conversation.



The Key Flag RC?

There will be times when you want to see if a new key has been pressed, but don't care which key. TRUE or FALSE is enough. The RC? flag is just what you need. RC? is the name of a standard LOGO procedure that causes the computer to check if any key has been pressed. If no new key has been pressed, then RC? reports the word "FALSE. If some new key has been pressed, then the computer returns the word "TRUE. RC? stays "TRUE until RC is used in some way. As soon as the character in RC is used, RC? changes back to "FALSE. Here is an example:

Type this:

```
RC?
```

The computer finds no key pressed. RC? returns the value "FALSE. You see

```
TELL ME WHAT TO DO WITH FALSE
```

Type this:

```
REPEAT 1000 [PRINT RC? ]
```

Now, while the computer is busy checking, press some key. You see

```
FALSE
FALSE
FALSE
TRUE    ☜ key pressed here
TRUE
TRUE
```

RC? stays TRUE until RC is used. The next example will use RC by typing it.

Key Crash

This next procedure makes the keyboard act strangely. Asterisks (**) are typed before and after every letter you type.

```
TO KEY
IF RC? THEN TYPE "** TYPE RC TYPE"**
KEY
END
```

This procedure does nothing until you press a key. Then RC? changes to TRUE. Then the procedure types "**, then types the RC character, then types "**. We typed H I H O and saw

```
**H****I****H****O**
```

- Change KEY to this:

```
TO KEY
IF RC? THEN MAKE "X RC REPEAT 30 [TYPE :X ]
PRINT [WAITING ]
KEY
END
```

Now when you call KEY, you see

```
WAITING
WAITING
WAITING
WAITING
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
WAITING
WAITING
```

we pressed the A key here

Press (FCTN)(BACK) to stop this procedure.

- Change KEY so that the computer prints STOP THAT whenever the Z key is pressed.
- Change KEY so that the computer clears the screen whenever the 4 key is pressed.

ANSWERS:

```
IF RC? THEN IF RC = "Z THEN PRINT [STOP THAT! ]
IF RC? THEN IF RC = "4 THEN CS
```

Summary of Chapter 5

This chapter was about the kinds of objects used in LOGO. In this chapter you learned:

- That "RED is the word RED
- That :RED is the object whose name is "RED
- That RED is the procedure whose name is "RED
- That [RED] is the list containing the word "RED
- That you can MAKE a name stand for an object
- That you can CALL an object by a name
- That RC reads one character from the keyboard
- That the IF . . THEN command is used to test and branch
- That RL reads a line that ends with (ENTER)

- That the apostrophe mark (') tells LOGO to leave in spaces
- How to write a program that helps you write letters
- How to write an interactive story
- That RC? returns TRUE if a new key is pressed

Self-Test—Chapter 5

1. Say whether each of these is a word, a list, a procedure or some other LOGO object.

   a. :BLOT __________
   b. "BLOB __________
   c. [BLAT] __________
   d. BLIMP __________

2. Give the command that assigns the name MOP to the list [HEAD HANDLE]. ______________________
3. What will make these commands print?

```
MAKE "DONUT "BAGEL
PRINT :DONUT
```

   ____

4. Write a command that will print the next keyboard character that you type. ______________________
5. Write a command that will store the next character that you type under the name "NEXT. ______________________
6. Write a procedure called CONTRARY that lets the user type a word. If the word is YES, the computer types NO. If the word is NO, the computer types YES.

   ____
   ____
   ____
   ____
   ____
   ____

7. Write a procedure called SP2 that will type 2 blank spaces whenever it is called.

8. Write a procedure called BUSY that will repeatedly type the word BUSY, but will clear the screen and start over whenever you press any key.

9. Write a procedure called DRAGON that asks the user's name, stores the name as "NAME, then says PLEASE COME FOR DINNER ________. The name goes in the blank.

10. Write a procedure called TOUCH that types OUCH anytime a G key is pressed, but does nothing otherwise.

Answers

1. a. some logo object named "BLOT
   b. a word
   c. a list
   d. a procedure

```
2.  MAKE "MOP [HEAD HANDLE ]
3.  BAGEL
4.  PRINT RC
5.  MAKE "NEXT RC
6.  TO CONTRARY
      MAKE "ANSWER RL
      IF :ANSWER = [YES ] THEN PRINT "NO
      IF :ANSWER = [NO ] THEN PRINT "YES
      CONTRARY
      END
7.  TO SP2
      TYPE [ '' ]  ☜ two spaces
      END
8.  TO BUSY
      TYPE "BUSY
      IF RC? THEN CS
      BUSY
      END
9.  TO DRAGON
      PRINT [HI, WHAT'S YOUR NAME? ]
      MAKE "NAME RL
      TYPE [PLEASE COME FOR DINNER ] TYPE [ '' ] TYPE :NAME
      END
10. TO OUCH
      IF RC? THEN IF RC = "G THEN PRINT "OUCH
      OUCH
      END
```

6

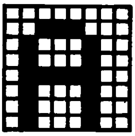
Tile Tales

You have already met the 32 sprites and the graphic turtle. In this chapter you will meet the character tiles. They are the shapes of all the letters and the numbers. In this chapter you will learn:

- That there are 256 numbered character tiles
- How to make new character tiles
- That the blank screen is really filled with character tile 32
- How to fill the screen with pattern by changing tile 32
- How to get the attention of a tile
- That character tiles come in families of eight
- How to change the color of a family of characters
- How to put a face on the cursor
- How to print a character at any screen address
- The secret of the mysterious Texas jigsaw characters
- How to make a procedure that codes and decodes secret messages
- How to make the computer count
- How to make a program that draws on the screen with colored blocks

Character Tiles

In this chapter you will meet the 256 character tiles. The character tiles hold the shapes that you see whenever you type a letter on the keyboard. Character tile 65 looks like this:



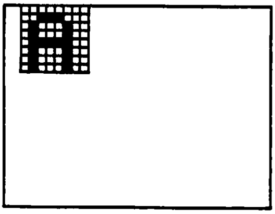
Character tile 65 is just the letter A. Would you prefer a fancier A? You can make a new character shape with the MAKECHAR command MC. Type this:

```
MC 65
```

or

```
MAKECHAR 65
```

You see the screen change to look like this:

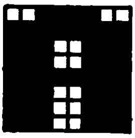


The MAKECHAR procedure is now at work to help you make a new character tile 65. The (FNCT) arrow keys print and move just like they do when making sprite shapes.

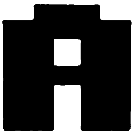
Here is how the (FCTN) arrow keys work:

(FCTN) ARROW PRINTS BLACK AND JUMPS
ARROW PRINTS CLEAR AND JUMPS

We changed A to look like this:



You can make any shape you like. When you are done, type the (FCTN) (BACK) key to get back to command mode. Now type the A key. You see the new version of character tile 65. We saw this:



The Blank Character 32

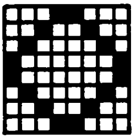
You can't see it, but the screen is filled with character tile 32. Character tile 32 is blank. The blank is what the space bar at the bottom of the keyboard types. Here is a surprise to try:

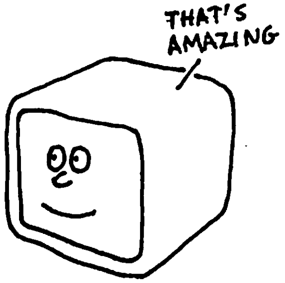
MC 32

Now make some changes using the (FCTN) arrow keys. The whole screen changes as you draw.

The screen is filled with the blank character 32. When you change character tile 32, the whole screen changes.

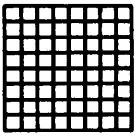
- Try this version of character 32:



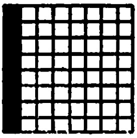


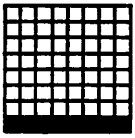
The (FCTN)(CLEAR) Key

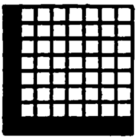
You can clear any character tile quickly and easily with the (FCTN) (CLEAR) key. While you are in the MAKECHAR procedure, type the (FCTN) (CLEAR) key and the character grid will be immediately wiped clear.



- Try these versions of character tile 32:

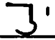
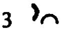
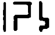
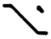
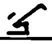




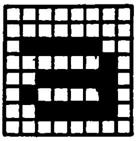
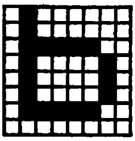
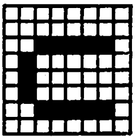
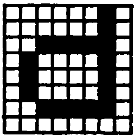
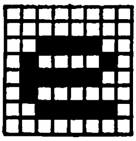
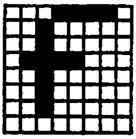
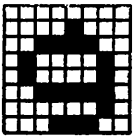
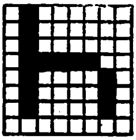
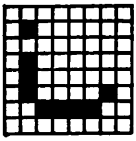
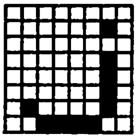
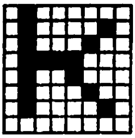
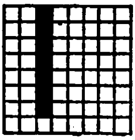
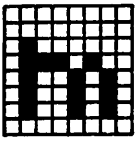
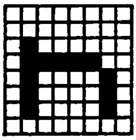
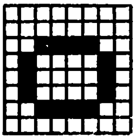
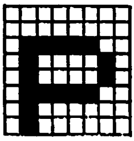
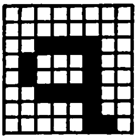
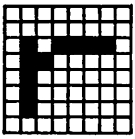
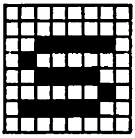
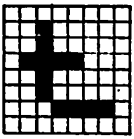
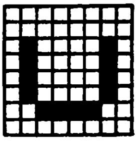
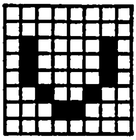
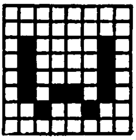
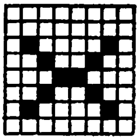
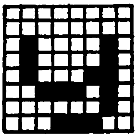
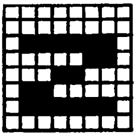
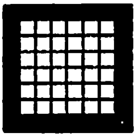


The 256 Character Tiles

Every keyboard character has a number. The number is called its ASCII number (pronounced "askee"). The ASCII number of A is 65. The ASCII number of B is 66, and so on. Here is a table that shows the numbers of all the keyboard characters, and some others besides:

0 □	1 █	2 	3 	4	5	6 	7
8 	9 	10 ©	11	12	13 (CR)	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 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7
56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G
72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	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
128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151
152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167
168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183
184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215
216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231
232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247
248	249	250	251	252	253	254	255

Many of the character tiles are blank, just waiting for your creative touch. Would you like a fancier alphabet? You can do it. Here is an example of a lower-case alphabet. You could store these on character tiles. If you put these shapes on tiles 97 to 122, then the lower case characters will be exactly 32 larger than the upper case letters.

Cheap Joke

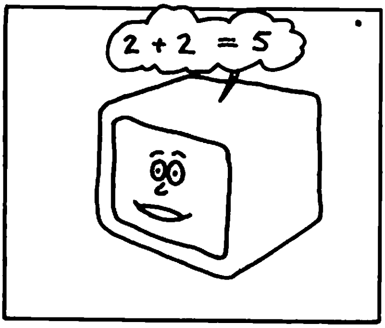
You might like to try your skill at making new character tiles and play a little joke on a friend at the same time. What do you suppose will happen if you change the C key so that it types D; the A key so that it types O, and the T key so that it types G. Have a friend try to type CAT and see that happens. Character tile 67 is C, character tile 65 is A, character tile 84 is T. Go do it.

- Some people think computers are smart and never make mistakes. Character tile 52 is the digit 4. Change character tile 52 to the digit 5. Now try this:

```
PRINT 2 + 2
```

Your poor confused computer prints

```
5
```

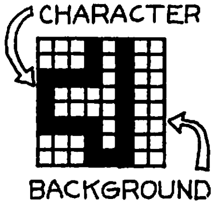


The (FCTN)(QUIT) Key

The letters and numbers are becoming a bit confused in your computer. You can start fresh. Press the (FCTN)(QUIT) key. This has the same effect as turning your computer off and then on again. Everything is just like new again.

Set Color

Each character tile has two colors, the color of the character itself and the color of the background. The background is usually clear. You will see in a moment how to change both the character color and the background color.



To change the color of a character tile you must first get its attention with the TELL command. Type this:

```
TELL TILE 66
SC :WHITE
```

GET THE TILE'S ATTENTION.
TILE 66 is B.

Now type some B's. The B's are now typed in white. Some other letters also got changed to white.

Eight Letter Color Family

The letter B belongs to a family of eight letters that all change color together. Type these letters:

```
@ABCDEFG
```

ONE FOR ALL
ALL FOR ONE

these are all white

You can see the color families in the ASCII table.

Here is part of the ASCII table. Each eight-letter line forms a color family. When you change the color of one member of the family, you change the color of all the members of the family.

32	33 !	34 "	35 #	36 $	37 %	38 &	39 '
40 (	41)	42 *	43 +	44 ,	45 –	46 •	47 /
48 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7
56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G
72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	91 [	92 \	93]	94 ^	95 —

Type these letters:

```
HIJKLMNO
```

They are all in the same family and are still colored black. Let's change the color of K and see what happens:

```
TELL TILE 75
SC :BLUE
```

Tile 75 is K

You see all the letters HIJKLMNO immediately change to blue.

- Type SC 0. What happens?
- Try some more colors. Find which colors are clearest on your TV.

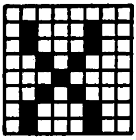
Flasher

Here is a command that makes the current letters flash on and off:

```
REPEAT 100 [ SC 0 WAIT 20 SC 1 WAIT 20 ]
                ↖           ↖
              clear        black
```

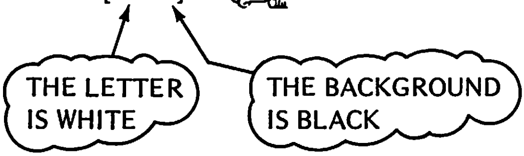
Figure and Ground

Every tile starts out black on a clear background.

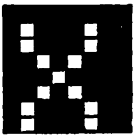


You can change both colors at the same time. Type a row of X's. The letter X is character tile 88. Now type this:

```
TELL TILE 88
SC [ 15 1 ]
```
☜ a list of two numbers



Now your X's look like this:

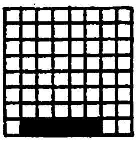


Notice that the cursor and the brackets also changed. The cursor and the brackets are in the same color family as X.

X Y Z [\ ] ∧ _ ☜ all in the family

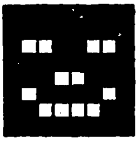
The Cursor

Character tile 95 is our old friend the cursor. Character tile 95 is also the underline key (FCTN)(U). Type MC 95 to get a close-up look at the cursor.



CHARACTER 95
THE CURSOR

A character with more character might be interesting. Here is how we made our cursor. You might try something different.



Print A Character

There are more character tiles than keys. You can't type all the possible character tiles. There is a special command to print a character by number. The PRINTCHAR command PC prints a character by its number. Try this:

```
PC 65
```

or

```
PRINTCHAR 65
```

The computer types the letter A, which is character 65.

The PRINTCHAR command PC types a character at the current cursor position, then moves to the next position. You can print any combination of letters or other character tiles. Try this:

```
PC 66 PC 85 PC 71 (ENTER)
```

The computer prints

```
BUG_
```

The PRINTCHAR command PC allows you to type any of the 256 possible character tiles.

Character 13 Is Special

Character 13 does a very special job. The number 13 belongs to the (ENTER) key. The computer is always alert for the character number 13. When you press (ENTER) the computer goes to the beginning of the next line. When you print character 13 the computer will go to the beginning of the next line. Give this a try:

```
PC 65 PC 13 PC 13 PC 65 (ENTER)
```

You see this:

```
A
A_
```

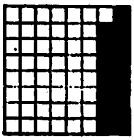
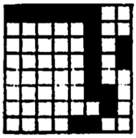
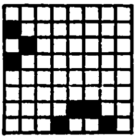
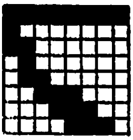
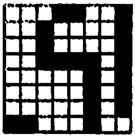
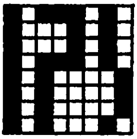
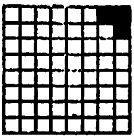
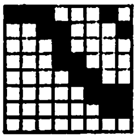
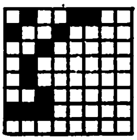
The commands have much the same effect as typing

```
A (ENTER) (ENTER) A
```

Of course, the computer knows that these are to be printed and doesn't give any error message.

Texas Surprise

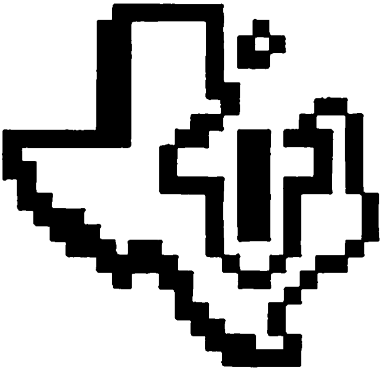
Here is a Texas Instruments jigsaw puzzle. It uses those mysterious character tiles 1, 2, 3, 4, 5, 5, 6, 7, 9. Use the MAKECHARACTER command MC to look at them. Here is what you see:

Now, let's put all the pieces of the jigsaw together. Type this procedure:

```
TO TI
PC 1 PC 2 PC 3 PC 13
PC 4 PC 5 PC 6 PC 13
PC 7 PC 8 PC 9 PC 13
END
```

When you type the command TI you see the Texas Instruments emblem.



Character Numbers

Your trusty computer can tell you the number that goes with a character. The CHARNUM command CN will do the job. Try this:

```
CN "A
```

or

```
CHARNUM "A
```

You see

```
TELL ME WHAT TO DO WITH 65
```

the CHARNUM of "A

Try this:

```
PRINT CN "Z
```

The computer prints

```
90
```

Here is a short procedure called KEYNUM that reads each character as you type it, and prints out the character number.

```
TO KEYNUM
PRINT CN RC          ☜ The character number of READCHAR
KEYNUM
END
```

Here is a procedure that does nothing. Can you see how it does nothing?

```
TO NOTHING
PC CN RC
NOTHING
END
```

The procedure NOTHING prints exactly what you type. The command CN RC is the number of the character read from the keyboard. PC CN RC prints the character with the number read from the keyboard. It prints exactly what you type.

KEYCODE

A modest change to the procedure NOTHING will change it into KEYCODE. KEYCODE changes what you type into code.

```
TO KEYCODE
PC (CN RC) + 1
KEYCODE
END
```

Now A types B, B types C and so on. (FNCT)(6) is now the (ENTER) key. (CN RC) is the character number of the character read from the keyboard. When you add 1 to it you get the next character.

- Will KEYCODE work without the parentheses? Why not?
- Add a line to KEYCODE to make the computer give a short beep each time you press a key.
- Write a procedure called DECODE that will decode the code from KEYCODE. Hint: subtract 1.

ANSWERS: PC CN RC + 1 won't work. The computer reads from the left. It finds PC. Next it expects a number. It finds CN. It expects to find a character next. RC brings a character. The computer determines the character that corresponds to PC CN RC. Next it finds + 1, but the computer can't add the character PC CN RC to a number.

How to Count

Counting is useful, but often tedious. It is something people do much better than any other creature. You can teach your computer to count. The trick is to get the next number by adding 1. Here is how to count from 0 to 2:

```
MAKE "X 0
PRINT :X
MAKE "X :X + 1          ☚ "X is 0 + 1
PRINT :X
MAKE "X :X + 1          ☚ "X is 1 + 1
PRINT :X
```

- Add two lines to make the computer print 3. No fair using 3.

You can see that there is a lot of repeating. Here is a procedure that counts from 0 to 255:

MAKE the new X equal to the old X plus 1

```
TO COUNT
MAKE "X 0
REPEAT 256 [ PRINT :X MAKE "X :X + 1 ]
END
```

When you COUNT you see the numbers 0 to 255 count down the screen:

- Change COUNT so that it counts to 999 instead of 255.
- Change COUNT so that it starts counting at 1 instead of 0.
- Change COUNT so that it TYPEs :X instead of PRINTing :X.

All the Tiles

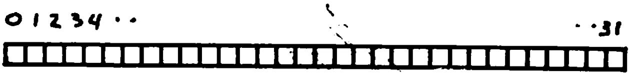
You can modify the procedure COUNT so that it prints out all the possible character tiles. Try this:

```
TO COUNT
CS
MAKE "X 0
REPEAT 255 [ PC :X MAKE "X :X + 1 ]
END
```

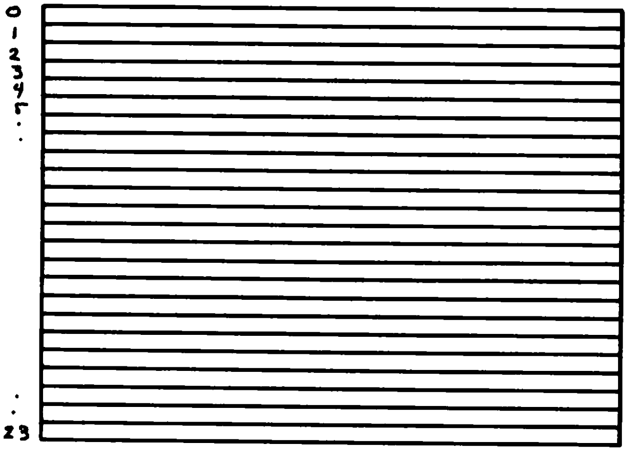
When you type COUNT you see all the characters appear on the screen. The characters from 96 to 255 are blank, unless you have created your own character tiles.

The Character Screen

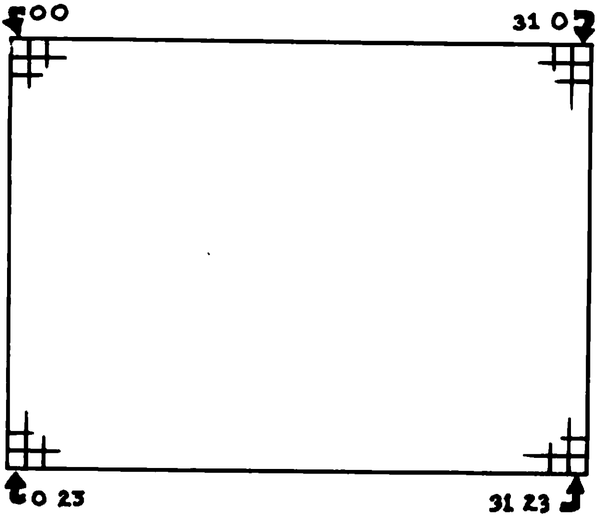
The character screen line is 32 character tiles wide.



The screen holds 24 lines.



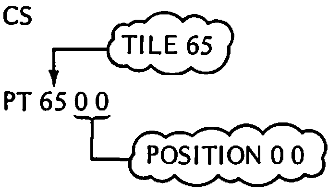
There is enough room on the screen for 768 character tiles (32 times 24 equals 768). Each of the 768 has its own two-number name. Here is a picture of the screen with some of the names shown:



The names start with 0 0 in the upper left corner and end with 31 23 in the lower right corner.

Put A Tile Anywhere

There is a special print command that lets you put a tile at any location on the screen. The PUTTILE command PT does the job. Give this a try:



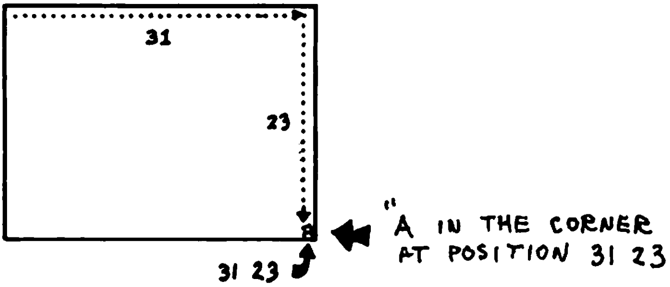
```
CS
PT 65 0 0
```

Look close. You will see character 65, the A, at position 0 0, the upper left hand corner.

- Try these:

```
PT 65 1 0
PT 65 31 0
PT 65 0 23
PT 65 31 23
```

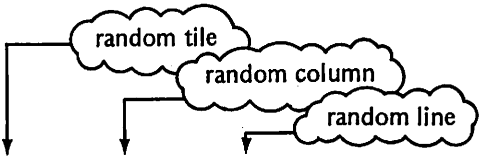
Here is what you see when you type PT 65 31 23:



- Use the PUTTILE command to put a B right in the middle of the screen.

Random Tiles

Here is a procedure that puts tiles 0 to 9 randomly in the upper left corner of the screen.



```
TO CORNER
CS
REPEAT 1000 [ PT RANDOM RANDOM RANDOM ]
END
```

- Change the REPEAT command so that it looks like this:

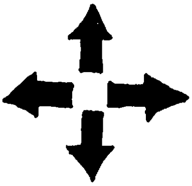
```
REPEAT 1000 [ CS WAIT 5 PT RANDOM RANDOM RANDOM ]
```

Here is a procedure that prints characters 0 to 9 at random, in the middle of the screen, at position 16 12.

```
TO WIGGLE
REPEAT 1000 [ CS PT RANDOM 16 12 ]
END
```

Paint Blocks

Here is a program to draw in colored squares on your color TV screen. You can move the cursor around the screen by pressing the arrow keys. When you press the first letter of a color name, that color is printed.



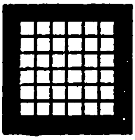
☜ Arrow keys move the cursor

W, R, O, Y, G, B, P, 1 ☜ Paint colors on the screen

The procedure PAINT is the manager of two other procedures, called SETUP and CHECK, that do all the work.

```
TO PAINT
SETUP
CHECK
END
```

The procedure SETUP clears the screen, sets the colors of the painting tiles, alerts and positions the sprite that will act as cursor and defines the starting positions. The procedure CHECK checks the keyboard, moves the sprite, and puts characters on the screen. You must make the shape that the sprite will carry. The cursor shape is on shape 9. When you MAKESHAPE 9 it should look like this:



Here is the procedure SETUP:

```
TO SETUP
CS
MAKE "X 1  ☜ Set the initial print position
MAKE "Y 1

TELL TILE 96  SC [0 15 ]  ☜ WHITE    }
TELL TILE 104 SC [0 1 ]   ☜ BLACK    }
TELL TILE 112 SC [0 6 ]   ☜ RED      }
TELL TILE 120 SC [0 9 ]   ☜ ORANGE   }
TELL TILE 128 SC [0 10 ]  ☜ YELLOW   }  Set up tiles
TELL TILE 136 SC [0 2 ]   ☜ GREEN    }
TELL TILE 144 SC [0 4 ]   ☜ BLUE     }
TELL TILE 152 SC [0 13 ]  ☜ PURPLE   }
TELL TILE 160 SC [0 7 ]   ☜ CLEAR    }

TELL 1 CARRY 9 SC 1
SXY (-120) 89
END

TO CHECK
MAKE "Z RC  ☜ "Z records the key

IF :Z = "S THEN MAKE "X ( :X - 1 )
IF :Z = "D THEN MAKE "X ( :X + 1 )
IF :Z = "X THEN MAKE "Y ( :Y + 1 )
IF :Z = "E THEN MAKE "Y ( :Y - 1 )

SXY ( :X * 8 - 128 ) ( ( - :Y )  * 8 + 97 )  ☜ Position sprite
```

```
IF :Z = "W THEN PT 96 :X :Y
IF :Z = "1 THEN PT 104 :X :Y
IF :Z = "R THEN PT 112 :X :Y
IF :Z = "0 THEN PT 120 :X :Y
IF :Z = "Y THEN PT 128 :X :Y       }  ☜ Check key, put tile
IF :Z = "G THEN PT 136 :X :Y
IF :Z = "B THEN PT 144 :X :Y
IF :Z = "P THEN PT 152 :X :Y
IF :Z = "C THEN PT 160 :X :Y

CHECK   ☜ Check again
```

- Two keys have not been explained. Can you figure out what the "1 and the "C keys do?

Summary of Chapter 6

This chapter was about the 256 character tiles. In this chapter you learned:

- That the MAKECHAR command MC 65 allows you to change character tile 65
- That character 32 is the blank
- That the clear screen is filled with the blank character 32
- That the (FCTN)(CLEAR) key clears the makecharacter grid
- That the TELL TILE command is used to get a tile's attention
- That the SETCOLOR command SC 6 sets the current sprite's color red
- That the keyboard characters are arranged in families of eight
- That setting the color of one member of a family sets them all
- That every tile starts on a clear background
- That the SETCOLOR command SC[2 6] sets the letter and the background of the current tile
- That the cursor is character 95
- That the PRINTCHAR command PC 65 prints character tile 65
- That the (ENTER) key is character 13
- That the CHARNUM command CN "A returns the number 65, the number of the character "A
- How to make your computer count

- That the character screen is 32 characters wide and 24 lines deep
- That the PUTTILE command PT 65 15 10 puts tile 65, an "A, at screen position 15 10
- That the sprites can move over the character screen

Self-Test—Chapter 6

1. Tell how to change the X key so that it types a square block:

   ____

   ____

2. Write the command that gets the attention of the A tile.

   ____

3. Write the command that will change the color of tile number 66 to red.

   ____

4. Which characters belong to the same color family as the letter I?

   ____

5. Write the command that will make the A key print a red A on a green background.

   ____

6. Write a command that will change the background of tiles 96 to 103 to black and the letters to white.

   ____

7. Write a procedure which prints the letter A, then flashes it on and off, once every 30 jiffies.

   ____

   ____

   ____

   ____

   ____

8. Write a procedure called COUNT that prints the numbers from 0 to 99.

   ____

   ____

   ____

   ____

9. Write a procedure called LETTERS that will print the letters A through Z, one at a time, for one second each, at the center of the screen.

____

____

____

____

____

10. Write a procedure called arrows that will cause the S key to print LEFT, the D key to print RIGHT, the E key to print UP and the X key to print DOWN.

____

____

____

____

____

____

Answers

1. Type MC 88; fill in the grid; press (FCTN)(BACK). Now, X prints a block.
2. TELL TILE 65
3. TELL TILE 65 SC :RED
4. Characters H, I, J, K, L, M, N and O are in the same color family.
5. TELL TILE 65 SC[6 2]
6. Tiles 96 to 103 are in the same color family. If you change one, you change them all. This works:

   ```
   TELL TILE 96 SC[15 1 ]
   ```

7. Here is our procedure. There are many other ways to do this problem.

   ```
   TO FLASHA
   PT 65 15 10  WAIT 30 ☜ Put A in the middle, wait
   PT 32 15 10  WAIT 30 ☜ Put blank in the middle, wait
   FLASHA
   END
   ```

```
 8. TO COUNT
      MAKE "X 0
      REPEAT 100 [PRINT :X MAKE "X :X+1 ]
      END
 9. TO LETTERS
      MAKE "X 65 ☜ Start with CHARNUM "A
      REPEAT 26 [PT :X 15 10 WAIT 60 MAKE "X :X+1 ]
      END
```

10. Here's how we did it. There are many other ways to do this problem. Yours may look quite different and still be ok.

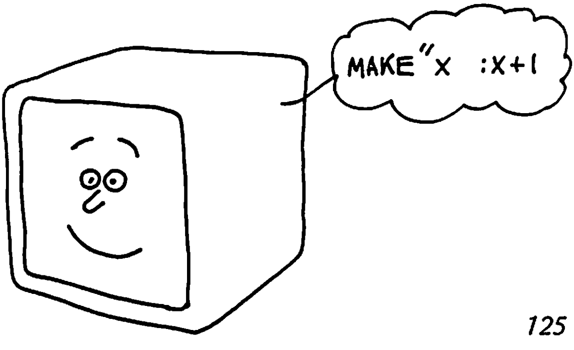
```
      TO ARROWS
      IF RC? THEN MAKE "KEY RC CHECK
      ARROWS
      END
      TO CHECK
      IF :KEY = "S THEN PRINT "LEFT
      IF :KEY = "D THEN PRINT "RIGHT
      IF :KEY = "E THEN PRINT "UP
      IF :KEY = "X THEN PRINT "DOWN
      END
```

7

What Counts

This chapter is about numbers and how LOGO deals with numbers. In this chapter you will learn:

- How to make your computer do arithmetic calculations
- How to make your computer take in numbers from the keyboard
- How to get the first number from a list of numbers
- The lowest and highest numbers that your computer can use
- How to use the sizes of numbers to control the computer
- How to write a program that makes a number guessing game
- That each sprite can use its own number in computations
- How to use the state report procedures to control the sprites
- How to simulate the flight of a cannon ball
- How to make procedures that output numbers and other objects
- How to make a computer coin tosser
- How to make procedures that take in numbers as input
- How to make a procedure that does perfect division



Number Power

You can't walk very far these days before you trip over a number. Numbers are everywhere. Computers are built to work with numbers. Your TI99 computer is a number whiz. It can add, subtract, multiply, divide and much more.

Here is a simple example that shows how the TI99 does an arithmetic problem:

`PRINT 2*3` 2*3 means multiply 2 times 3

The computer prints 6.

You can do more. Type this:

```
PRINT 2 * 3 + 7
```

The computer does multiplications first, then additions. The computer multiplies 2 times 3, then adds 7. You see:

```
13
```

- Remember that multiplications are done before additions. What will this print?

`PRINT 5 + 3 * 2` The computer does multiplication first

The computer does the multiplication 3 * 2 first, then adds 5, like this:

5 + (3 * 2) That's 5 + 6

You see the answer 11.

Parentheses ( )

You can use the parentheses marks to tell the computer to change its usual order of calculation. Try this:

`PRINT ( 5 + 3 ) * 2` That's 8 * 2

The computer does the calculation inside the parentheses first, and gives the answer 8 * 2 equals 16.

- What's the result of these calculations?

(2 + 3) * 4 __________

2 + 3 * 4 __________

2 * 4 + 3 * 5 __________

2 * (4 + 3) * 5 __________

ANSWERS: 5*4 = 20; 2+12 = 14; 8+15 = 23; 2*7*5 = 70

Divide Without Remainder

Division problems are easy for your TI99 computer. Try this:

`PRINT 6/3` ☜ 6 divided by 3

You see

```
2
```

The slash mark / means DIVIDED BY.

Here's a question about division. It has four different answers.

WHAT IS 7 DIVIDED BY 2?

ANSWER 1	7 divided by 2 is 3
ANSWER 2	7 divided by 2 is 3 with remainder 1
ANSWER 3	7 divided by 2 is 3 and 1/2
ANSWER 4	7 divided by 2 is 3.5

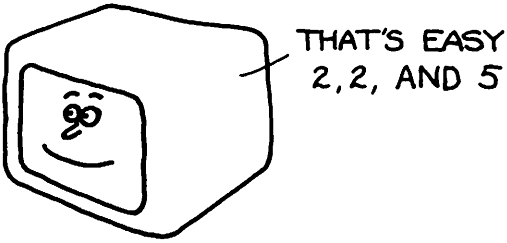
In TI LOGO, ANSWER 1 is the correct answer.

7 divided by 2 is 3 ☜ division without remainder

LOGO divides and ignores the remainder.

- Tell what answers the computer gives to these divisions:

7/3 ________
12/5 ________
26/5 ________



Use the computer to check your answer. Type this:

```
7/3
```

You see

```
TELL ME WHAT TO DO WITH 2
```

The computer knows that the value of 7/3 is 2, but it doesn't know what it should do with this answer. Let's tell it to PRINT the result. Type:

```
PRINT 7/3
```

You see:

```
2
```

Arithmetic Order

The computer does arithmetic inside parentheses first, then multiplications and divisions, then additions and subtractions. If there is a series of additions and/or subtractions to do, then the computer works from left to right. If there is a series of multiplications and/or divisions to do, then the computer works from left to right.

- Tell what the answer will be for these problems, then try them on the computer.

a. 6 + 4/2 __________
b. (6 + 4)/2 __________
c. 3*5/2 __________
d. 5/2*3 __________
e. (5/2)*3 __________
f. 5/(2*3) __________
g. 3*6 + 12*3 __________
h. 3*(6 + 12/3) __________
i. 3*((6 + 12)/3) __________
j. 13/3*3 __________

ANSWERS: a) 6+2 = 8; b) 10/2 = 5; c) 15/2 = 7; d) 2*3 = 6; e) 2*3 = 6; f) 5/6 = 0; g) 18 + 36 = 54; h) 3*(6 + 4) = 3*10 = 30; i) 3*(18/3) = 3*6 = 18; j) (13/3)*3 = 4*3 = 12.

Numbers from Lists

There will be times when you will want to tell the computer to read numbers from the keyboard. The READLINE command RL will do part of the job. As you learned in Chapter 5, the READLINE command RL reads a list of numbers or letters from the keyboard. Type this:

```
RL (ENTER)
```

The computer types the prompt >, then waits for you to type a list. Type this:

```
>27 35 44
```

You see this:

```
TELL ME WHAT TO DO WITH [27 35 44 ]
```

☜ A list of numbers

Notice that the computer put brackets around your numbers to make a list. The READLINE procedure RL returns a list. A list is not a number, but contains the number you want. The number is the first object in the list.

The First of a List

To get a number out of its list, you need the FIRST procedure. Try this:

```
FIRST [27 35 44 ]
```

You see:

```
TELL ME WHAT TO DO WITH 27
```

The FIRST procedure returns the first object inside a list, in this case, the number 27.

Here is an example that uses the FIRST and the RL procedures to get the number that you type, and then print twice the number:

```
TO TWICE
MAKE "N FIRST RL ☜ Get the first object in the list
PRINT 2 * :N ☜ Print twice the number
TWICE ☜ Go do the procedure again
END
```

When you type TWICE, you see the prompt

```
>
```

Type

```
12 (ENTER)
```

You see

```
24 ☜ 2 times 12
> ☜ Prompt for another list
```

The computer is repeating the TWICE procedure and is prompting you for a new list of numbers. Each time you type a list of numbers and press the (ENTER) key, the computer will print twice the first number in the list. Press (FCTN)(BACK) to stop the procedure.

- Change the procedure TWICE so that it prints this calculation:

  ```
  PRINT :N * :N
  ```

  Use your new procedure to find the number that prints the result 529.

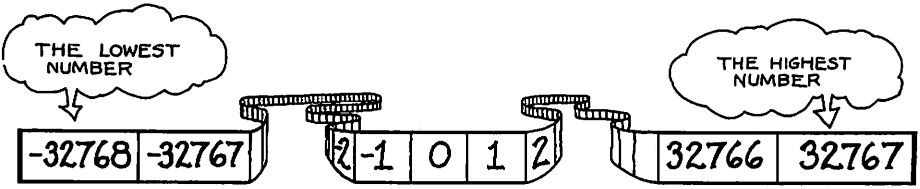
- Change the procedure TWICE so that it prints this calculation:

```
PRINT :N * :N - ( 12 * :N ) + 36
```

Use your new procedure to find the number that prints 0 as the result.

The Number Line

Your TI99 computer can count forward to 32767, and backward to -32768.



- Try printing a number higher than the highest. Try this:

PRINT 32768 ☜ higher than the highest

- Try printing a number lower than the lowest. Try this:

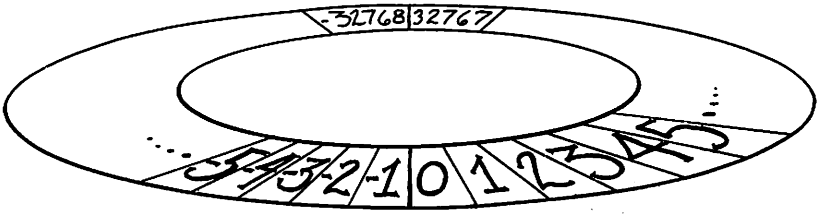
PRINT -32769 ☜ lower than the lowest

- Try the REPEAT command with a negative number. Try this:

REPEAT -1 [TYPE "*] ☜ counts backwards to 0

The Number Circle

Here is a picture that will help you understand LOGO numbers. The LOGO number line is really a circle, like this:



ANSWERS: The command PRINT 32768 prints -32768. The computer went around the number circle. The command PRINT -32769 prints 32767. The computer went around the number circle. The command REPEAT -1 [TYPE

"*] causes the computer to start its REPEAT counter at -1. The computer prints one "*, then subtracts 1 from the counter. Now the counter is at -2. Another "* is printed, and 1 is subtracted from the counter. Now the counter is -3. This process repeats as the counter goes around the number circle to -4, -5, . . ., -32768, 32767, . . ., 3, 2, 1, 0. When the counter reaches 0, the REPEAT process stops. If you have the patience to wait, you will see 65535 "*'s printed.

Size Wise

People can easily see whether one number is less than, or greater than, another number.

2 is less than 5
5 is less than 7
2 is greater than 1

LOGO has an abbreviation for the words IS LESS THAN.

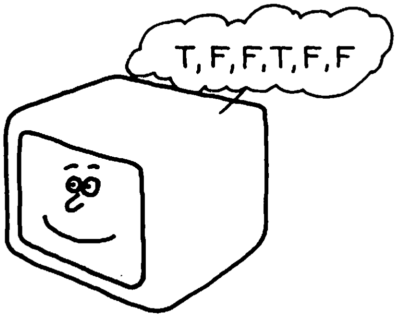
2 < 3 means 2 IS LESS THAN 3

The character < means IS LESS THAN. The small number goes on the small side, the large number goes on the large side. If you turn the LESS THAN sign < around, then you get the GREATER THAN sign >.

3 > 2 means 3 IS GREATER THAN 2

- Mark each of these TRUE or FALSE:

2 < 7 ________
7 < 2 ________
5 < 5 ________
2 > 1 ________
2 > 3 ________
12543 < 12534 ________



- Your computer can compare numbers. Type this:

```
1 < 2
```

You see

```
TELL ME WHAT TO DO WITH TRUE
```

- The computer can compare numbers to make choices. Type this:

```
IF 2 < 1 THEN PRINT [ZIP ]
```

Did you see anything? Nope. The computer does nothing, since it is false that 2 < 1. Now, try this:

```
IF 1 <2 THEN PRINT [OK ]
```

It is true that 1 < 2. You see:

```
OK
```

Here is an example that uses the LESS THAN sign to check the number you type.

```
TO PLAY
MAKE "G FIRST RL ☜ Get the first number from the list
IF :G < 100 THEN PRINT [TOO SMALL ] ☜ :G less than 100
PLAY
END
```

When you type PLAY you see the prompt >. Type a number. The computer makes "G the name of the number you type. If :G is less than 100 the computer prints TOO SMALL and prompts you again. If :G is not less than 100, then the computer just prompts you again. Type the (FCTN)(BACK) key when you are done.

Number Game

With some additions, you can change the procedure PLAY into a game. This game is really two procedures. The first, called GAME, causes the computer to choose a random number between 0 and 99. The procedure PLAY lets you try and guess the computer's number. Enter the following two procedures GAME and PLAY:

```
TO GAME
MAKE "N RANDOM + 10*RANDOM ☜ Random numbers 0 to 99
PLAY
END
```

Change the procedure PLAY to this:

```
TO PLAY
PRINT [GUESS MY NUMBER ]
MAKE "G FIRST RL ☜ The first in the list RL
IF :G < :N THEN PRINT [TOO SMALL ] ☜ Less than
IF :G > :N THEN PRINT [TOO BIG ] ☜ Greater than
IF :G = :N THEN PRINT [YOU GOT IT! ] ☜ Equal
IF :G = :N THEN STOP ☜ Stop the game
PLAY
END
```

When you type PLAY, you see:

```
GUESS MY NUMBER
```

We typed 23 as our guess. We saw:

```
TOO LOW
```

We typed 75 as our guess. We saw:

```
TOO HIGH
```

We finally typed 47 and saw:

```
YOU GOT IT!
```

- Change the procedure GAME so that it chooses random numbers from 0 to 999.
- Change the procedure GAME so that the computer chooses numbers between -55 and 54.
- Change the procedure PLAY so that it prints where the number is trapped.
- Try some ideas of your own.

EACH . . YOURNUMBER

Numbers are neat, but sprites are a delight. You can use numbers to control the sprites to get fast, colorful action on the screen. Each sprite has a number and each sprite can use its own number. This makes it easy to do complicated things easily. The EACH and the YOURNUMBER command are used together to make each current sprite use its own number. YOURNUMBER, or YN for short, returns the sprite's number. EACH tells each of the current sprites to do a list of commands which may use YN. Here is a simple procedure that does a lot. Give it a try:

```
TO POW
TELL :ALL ☜ All sprites are current
CARRY :BALL
SC :BLACK ☜ Setcolor
EACH [SH YN*24 ] ☜ Each current sprite, set heading to YN*24
SS 20 ☜ Setspeed
HOME
END
```

When you type POW, you see all the sprites explode out from the center. The sprites continue flying off in all directions.

- As the sprites fly type each of these commands:

```
SS 60 ☜ All sprites change speed
SS 120 ☜ All sprites change speed
HOME ☜ All sprites spread from home
EACH [SC YN ] ☜ Each sprite has its own color
```

- Set the background to black like this:

```
CB 1
```

Now you can see the brilliant sprites better.

- Give each sprite a different speed, like this:

```
EACH [SS YN ] (ENTER)
```

Now each sprite has a speed of its own.

- Send all the sprites home and get a surprise!

```
HOME
```

- Set the background color to CYAN 7 again, so you can see what you print, like this:

```
CB 7 ☜ Color background cyan
CS ☜ Clear screen
```

Stop the sprites:

```
SS 0
```

Set their colors clear:

```
SC 0
```

Sprite Report

Each sprite has different properties. Each has its own heading, X-coordinate, Y-coordinate, X-velocity, Y-velocity, and speed. LOGO has procedures that output information about a sprite's properties. For each procedure that sets the state of a sprite property, there is a corresponding procedure that returns a report about the state of that property.

Set State	*Report State*		
SH 90	HEADING	0 to 259	heading
SX 10	XCOR	-120 to 119	X coordinate
SY 10	YCOR	- 95 to 96	Y coordinate
SXV 10	XVEL	-127 to 127	X velocity
SYV 10	YVEL	-127 to 127	Y velocity
SS 10	SPEED	-127 to 127	speed

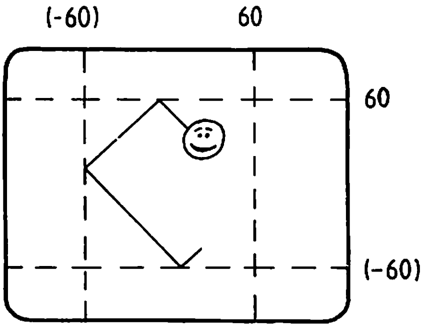
- Tell what each sequence of commands will print:

1. SH 30
   PRINT HEADING __________
2. SH 30
   RT 10
   PRINT HEADING __________
3. SX 10
   PRINT XCOR__________
4. SX 10
   SY 20
   PRINT YCOR__________
5. SXV 10
   SS 0
   PRINT XVEL __________

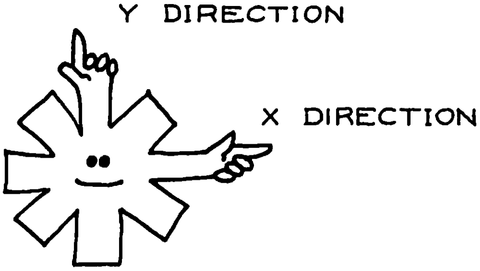
ANSWERS: 30, 40, 10, 20, 0. The last answer, 0, is a trick. The command SXV sets the speed in the X direction to 10, but the command SS 0 sets the total speed back to 0.

Bouncing Sprite

Now that you know about LOGO arithmetic, you can use numbers to control the sprites. Here is a procedure that uses numbers to make the sprite bounce off the sides of an invisible box on the screen.



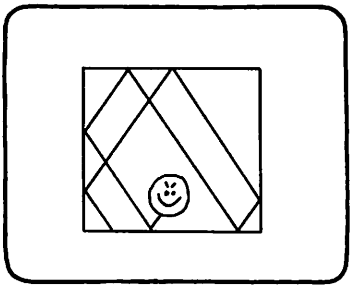
```
TO START
TELL 1 CARRY :BALL SC 1 ☜ A ball, black
SS 100 ☜ Set speed
SH RANDOM * 10 ☜ Set heading randomly
BOUNCE
END
```



```
TO BOUNCE
IF XCOR > 60 THEN SX 60 SXV (-XVEL)
IF XCOR < (-60) THEN SX (-60) SXV (-XVEL)
IF YCOR > 60 THEN SY 60 SYV (-YVEL)
IF YCOR < (-60) THEN SY (-60) SYV (-YVEL)
BOUNCE
END
```

If the coordinate gets too big, or too small, then set the coordinate back to 60, or (-60) and reverse the velocity.

When you type START the sprite begins to bound, like a berserk billiard ball, off the walls of the invisible square box.



Cannon Ball

This next procedure shows a lot about cannon balls. A cannon ball starts up fast, but then goes slower and slower. Finally it stops and begins falling backwards towards the ground. It falls slowly at first, then faster, and faster until it hits the ground and stops. You can say all this in LOGO.

```
SYV 80              the ball starts fast in the Y direction
SYV (YVEL -5)       set the Y velocity slower
SYV (YVEL -5)       set the Y velocity slower
SYV (YVEL -5)       set the Y velocity slower
```

This is repetitive. The computer is good at repetitive tasks. Here is a family of two procedures called SHOOT and SLOW. SHOOT sets up the sprite and starts it headed upward at high velocity. The second procedure is called SLOW. SLOW has the job of slowing down the sprite and checking if it is at ground level yet.

```
TO SHOOT
TELL 1 CARRY :BALL SC 1 HOME
SYV 80 ☜ Head up at velocity 80
SLOW
END

TO SLOW
IF YCOR < 0 THEN SYV 0 STOP ☜ If ground level, stop
SYV (YVEL -5) ☜ New Y velocity is YVEL -5
SLOW ☜ Go slow it more
END
```

When you type SHOOT, the ball starts up with speed 80. The speed drops to 75, then 70, eventually 0, then -5 (that's backwards), then -10, and eventually it falls back below ground level and stops.

The STOP Command

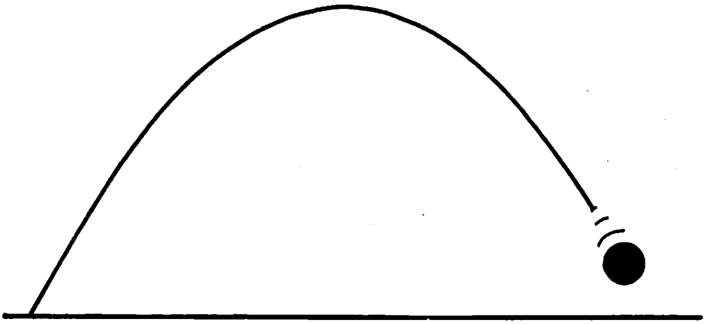
The STOP command in the last procedure allows you to stop a procedure whenever you want. It acts just like END, except that it is used to stop a procedure at places other than the END.

Trajectory

You no doubt noticed that the cannon ball went straight up and then came straight down. That is not considered good cannon ball shooting. A little forward motion will move the cannon ball elsewhere. Change the procedure SHOOT to this:

```
TO SHOOT
TELL 1 CARRY :BALL SC 1 HOME
SXV 10       ☜ This gives a little X-velocity
SYV 80
SLOW
END
```

Now, when you SHOOT, you see:



- It is easy to make your cannon ball bounce across the screen. Try this:

```
TO REBOUND
REPEAT 100 [SHOOT ]
```

Procedure Output

The RANDOM procedure yields a number as output. When you type:

```
PRINT RANDOM
```

you see some digit, like this:

```
7
```

The procedure RANDOM puts out a random number. The random number is the OUTPUT of the procedure RANDOM. You can make your own procedures output numbers and other objects. The command OUTPUT, or OP for short, sends objects out of a procedure. Here is an example of a procedure that always outputs the number 1. Give it a try.

```
TO ONE
OUTPUT 1
END
```

Now, any time the computer sees the procedure name ONE, it will replace the name with the number 1. Type this:

```
PRINT ONE
```

You see

```
1
```

Type this:

```
PRINT ONE + ONE
```

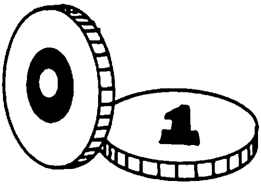
You see

```
2
```

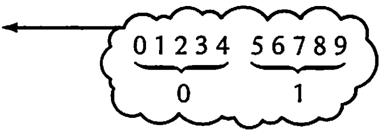
ONE is a procedure that outputs the number 1 wherever it occurs.

Coin Tosser

Here is an example of a procedure that outputs 0 or 1 at random. The procedure acts like a coin with 0 on one side and 1 on the other.



```
TO COIN
MAKE "N RANDOM    RANDOM outputs 0,1,2,3,4,5,6,7,8,9
IF :N < 5 OP 0    Output 0
IF :N > 4 OP 1    Output 1
END
```



You can use COIN like this:

```
PRINT COIN
```

You see a 0 or a 1.

Try this:

```
CS ☜ Clear screen
REPEAT 16 [TYPE COIN ]
```

We saw a line of 16 0's and 1's typed across the screen like this:

```
0010111010011101?
```

Procedure Input

A procedure can send output out and take input in. You have already used many procedures that take numbers in. Here are some procedures that you used:

```
SS 5 ☜ Set speed 5
SH 20 ☜ Set heading 20
SXY 10 30 ☜ Set X and Y coordinates to 10 and 30
```

A procedure can take in things besides numbers. The familiar procedure PRINT takes lists as input:

```
PRINT [SOME STUFF ] ☜ The list [SOME STUFF ] is input
```

You can create a procedure that takes input. Here is an example:

```
TO DOUBLE "N ☜ :N will be sent as input to the procedure
PRINT 2 * :N ☜ 2 times the input number :N
END
```

Now you can send a number to the procedure DOUBLE. Type this:

```
DOUBLE 3 ☜ 3 goes into the procedure
```

You see

```
6
```

The number 3 was taken in by the procedure. Then the procedure made "N the name of the number and printed the result 2*:N.

Square

Here is another procedure that takes input. The procedure SQUARE takes a number as input, and then OUTPUTs the product of the number with itself.

```
TO SQUARE "X ☜ "X is the name of the input number
OP :X * :X ☜ Output the square of :X
END
```

Now type

```
                 ┌─Input
                 ↓
PRINT SQUARE 3
```

You see

```
9 ☜ Output
```

Type this:

```
                 ┌─Input──────┐
                 ↓            ↓
PRINT SQUARE 3 + SQUARE 4 ☜ Print 9 + 16
```

You see

```
25 ☜ Output
```

- Find the number whose SQUARE is 2809.

Divisor

It is easy to see that 6/3 is an exact division and there is no remainder. It is easy to see that 7/3 is not exact, but has a remainder of 1. It is more difficult to see whether 1873/13 is an exact division or not. Is 1873/13 exact, or does it have a remainder?

The computer can help us decide whether 1873 is divided exactly by 13, with no remainder. First type this:

```
PRINT 1873/13
```

The computer prints

```
144
```

Well? Does that help? Is the division exact? If the division is exact, then 13 * 144 should equal 1873. The computer can help check.

```
PRINT 13 * 144
```

We are looking for 1873, but the computer prints

```
1872
```

The division of 1873 by 13 is not exact. The remainder of the division is 1873 - 1872 or 1. We can have the computer do all these computations in one line like this:

```
PRINT 1873 - ( ( 1873 / 13 ) * 13 )
```

The computer does all the calculations and prints the remainder of 1.

You can make a procedure that divides a first number :N by a second number :D. The procedure prints the quotient :Q and the remainder :R.

```
TO DIV "N "D ☜ The procedure works on "N and "D
MAKE "Q :N / :D ☜ :Q is the quotient
MAKE "R :N - ( ( :N / :D ) * :D ) ☜ :R is the remainder
TYPE :Q TYPE "+ TYPE :R TYPE" / PRINT :D
END
```

Get back to command mode and try DIV on 1873 and 13.

```
DIV 1873 13
```

The computer prints

```
144+1/13 ☜ The correct answer
```

- Try the DIV procedure on these problems:

  1001 divided by 13
  1001 divided by 17
  10001 divided by 11

Summary of Chapter 7

This chapter was about numbers and how to use numbers in procedures. In this chapter you learned:

- How to make the computer do arithmetic calculations
- That computations inside parentheses are done first
- That multiplications and divisions are done before additions and subtractions
- That 7/3 equals 2
- That TI LOGO ignores the remainder on division
- That the READLINE command RL returns a list
- That the FIRST procedure returns the first object in a list
- That the lowest TI LOGO number is -32768, and the largest is 32767
- That the TI LOGO numbers form a number circle
- That < means LESS THAN, and > means GREATER THAN
- How to use the IF . . THEN command and number comparison to control a procedure

- How to write a number guessing game
- That the EACH and the YOURNUMBER command YN allow each of the current sprites to use their own numbers in computations
- How to use the sprite report procedures HEADING, XCOR, YCOR, XVEL, YVEL and SPEED
- How to use the sprite report procedures to control the sprites
- How to STOP a procedure in the middle
- How to make a procedure OUTPUT an object
- How to make a procedure take in inputs

Self-Test–Chapter 7

1. What will this command print? PRINT 2 + 4*5 ____________
2. What will this command print? PRINT 15/4 ____________
3. Choose the kind of object that procedure RL returns:
   a. a number
   b. a word
   c. a list
   d. a procedure

   ____

4. What will this command print? PRINT FIRST [3 2 1]. ____________
5. What will this command print? IF 7<11 THEN PRINT [AAA]. ____________
6. Write a procedure called DECIMAL that outputs a random number between 0 and 99.

   ____

   ____

   ____

7. Write a procedure called REPORT that gives the heading and speed of the current sprite.

   ____

   ____

   ____

   ____

8. Write a command that will give each of the current sprites a speed equal to its own number.

   ____

9. Write a procedure called STEP that takes a number as input, then outputs 0 if the number is less than or equal to 0, and outputs 1 otherwise.

   ____

   ____

   ____

   ____

10. Write a procedure called VOLUME that takes as input the length, width, and the height of a box, then outputs the product of the three numbers.

    ____

    ____

    ____

Answers

1. The command PRINT 2 + 4*5 prints the number 22.
2. PRINT 15/4 prints the number 3.
3. The procedure RL returns a list.
4. The command PRINT FIRST [3 2 1] prints the number 3.
5. The command IF 7 < 11 THEN PRINT [AAA] prints AAA.
6. TO DECIMAL
   ```
   OP RANDOM + 10*RANDOM
   END
   ```
7. TO REPORT
   ```
   TYPE [HEADING = ] TYPE HEADING
   TYPE [SPEED = ] PRINT SPEED
   END
   ```
8. EACH [SS YN]
9. TO STEP "N
   ```
   IF :N < 0 THEN OP 0
   IF :N = 0 THEN OP 0
   IF :N > 0 THEN OP 1
   END
   ```

Here's another version that works:

```
 TO STEP "N
 IF N > 1 OP 1 ELSE OP 0
 END
```

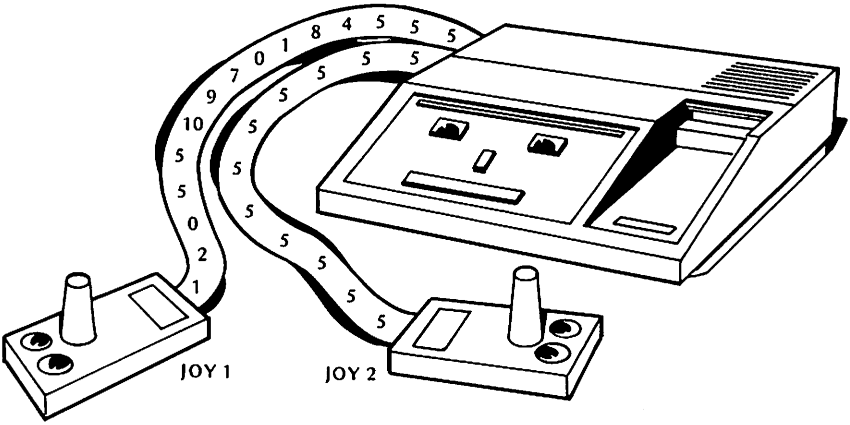
```
10. TO VOLUME "L "W "H
     OP :L * :W * :H
     END
```

8

The Joysticks

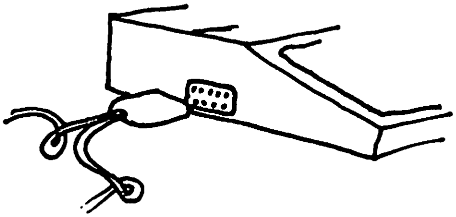
This chapter is about the joysticks. The joysticks are the easiest way to give information to the computer. In this chapter you will learn:

- About a procedure called JOY that returns joystick numbers
- How to use the joystick numbers in procedures
- How to decode the joystick number code
- How to use the joystick to control the sprites
- How to make a face whose expression is controlled by the joystick
- How to add comments using the semicolon mark (;)
- How to label a line and GO to the labeled line



The Joy of Joysticks

Your TI99 computer can do more than read the keyboard to see what you type. It can also read the position of the joystick. To get the most out of this chapter you will need to have a pair of TI joysticks.



Plug your joysticks into the socket on the left side of your computer. You won't see anything just yet. Those identical twin joysticks are called joystick 1 and joystick 2. We'll find which is which a little later.

A LOGO procedure called JOY 1 returns a number that tells the position of joystick 1. A LOGO procedure called JOY 2 returns a number that tells the position of joystick 2. Leave the joysticks pointed straight up, in center position. Type this:

```
JOY 1
```

You see:

```
TELL ME WHAT TO DO WITH 5
```

The computer reads the number from joystick 1. When the joystick is in center position it sends the number 5.

- Try this:

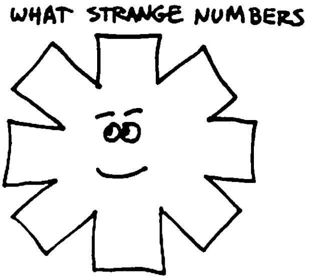
```
REPEAT 1000 [PRINT JOY 1 ]
```

You see this on the screen:

```
5
5
5
5
```

Move both joysticks. The numbers will begin to change when you move joystick 1. You see something like this:

```
4
1
5
6
5
```



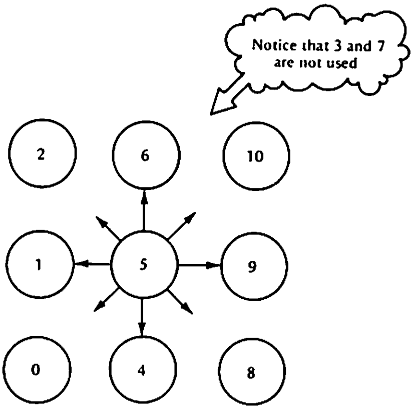
An Important Notice

THE ALPHA LOCK KEY MUST BE UP
WHEN USING THE JOYSTICKS

Move the joystick with the (ALPHA LOCK) key down. Nothing happens when you push the stick forward. Press the (ALPHA LOCK) key once, so it pops up. Now the joystick works properly at all positions.

The Joystick Numbers

Here is a picture of the numbers sent by the joystick:



- Try this:

REPEAT -1 [CB JOY 1] ☜ Repeats 65535 times

Now the joystick changes the background color. Press the (FCTN)(BACK) key when you are done.

- Try this:

```
REPEAT -1 [ IF (JOY 1) = 5 NOBEEP ELSE BEEP ]
```

The computer beeps whenever you move the joystick off center.

- Try this one:

```
REPEAT -1 [ PC JOY 1 ]
```

Each joystick position puts a different character on the screen. The characters are the Texas jigsaw pieces from Chapter 6.

- You can use both joysticks:

```
REPEAT -1 [ TYPE JOY 1 PRINT JOY 2]
```

- You can use the joystick numbers to choose a message. Here is a procedure that tells the position of the joystick.

FORWARD LEFT 2	FORWARD CENTER 6	FORWARD RIGHT 10
STOP LEFT 1	STOP CENTER 5	STOP RIGHT 9
BACK LEFT 0	BACK CENTER 4	BACK RIGHT 8

```
TO FLASHCARD
MAKE "J JOY 1
IF :J = 0 PRINT [BACK LEFT ]
IF :J = 1 PRINT [STOP LEFT ]
IF :J = 2 PRINT [FORWARD LEFT ]
IF :J = 4 PRINT [BACK CENTER ]
IF :J = 5 PRINT [STOP CENTER ]
IF :J = 6 PRINT [FORWARD CENTER ]
IF :J = 8 PRINT [BACK RIGHT ]
IF :J = 9 PRINT [STOP RIGHT ]
IF :J = 10 PRINT [FORWARD RIGHT ]
FLASHCARD
END
```

When you type FLASHCARD the procedure begins to check the joystick position. It calls the joystick number "J. The procedure prints the message that goes with the joystick number :J.

- Here is another family of words that might be controlled by the joystick: CREATE, PRESERVE, DESTROY and PAST, PRESENT, FUTURE.
- You can almost talk with the joystick. Here are 9 words to flash: WHO?, WHAT?, WHERE?, WHEN?, WHY?, HOW?, WHICH?, WHITHER?, WHENCE?.

The Secret Joystick Code

The joystick numbers are coded. Hidden in each joystick number are two numbers. The table below reveals the pattern:

2	2 = 0*4 + 2	6 = 1*4 + 2	10 = 2*4 + 2
1	1 = 0*4 + 1	5 = 1*4 + 1	9 = 2*4 + 1
0	0 = 0*4 + 0	4 = 1*4 + 0	8 = 2*4 + 0
	0	1	2

If you divide any joystick number by 4, you will get 0, 1 or 2. Division by 4 gives the column number.

2/4 = 0	6/4 = 1	10/4 = 2
1/4 = 0	5/4 = 1	9/4 = 2
0/4 = 0	4/4 = 1	8/4 = 2
0	1	2

If you divide by 4 and keep only the remainder, then you get the row number:

2	2/4 = 0 r 2	6/4 = 1 r 2	10/4 = 2 r 2
1	1/4 = 0 r 1	5/4 = 1 r 1	9/4 = 2 r 1
0	0/4 = 0 r 0	4/4 = 1 r 0	8/4 = 2 r 0

There are better numbers than 0, 1 and 2 for joystick numbers. It is most useful to have -1 mean left, 0 mean center, and 1 mean right. The next procedure does the whole job.

J1X

Here is a procedure called J1X which tells the movement of joystick 1 in the left-right (X) direction. J1X returns -1 if the stick is moved left; 0 if the stick is in the center, and 1 if the stick is moved right.

```
TO J1X
OP (JOY 1)/4-1 ☜ -1, 0, 1
END
```

- Check that J1X works. Type this:

```
REPEAT -1 [ PRINT J1X ]
```

You see -1, 0, 1 cascade down the screen as you move the joystick.

```
0
0
-1
1
```

J1Y

The next procedure returns -1, 0, and 1 as joystick 1 is positioned backwards, center or forwards.

The remainder on division by 4 gives the forward-backward movement, or Y direction movement of the stick. Remember how the computer found the remainder on division in Chapter 7? Here is an example that finds the remainder when 10 is divided by 4:

```
10 - (10/4*4)
```

This gives the remainder 2. The same method will work on JOY 1:

```
(JOY 1) - (JOY 1)/4*4
```

Here is a procedure called J1Y that calculates the remainder on division by 4, and subtracts 1 from it.

```
TO J1Y
OP (JOY 1) - (JOY 1)/4*4-1 ☜ Outputs -1, 0, 1
END
```

- Check J1Y. Try this:

```
REPEAT -1 [ PRINT J1Y ]
```

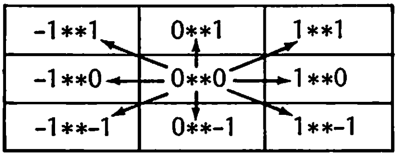
You see the numbers -1, 0, and 1 cascade down the screen as you move the joystick backward and forward.

You will use J1X and J1Y often in this chapter. Keep them in the computer's memory as you read this chapter.

- T.y this:

```
REPEAT -1 [ TYPE J1X TYPE "** PRINT J1Y ]
```

Now you can see both numbers change as you move joystick 1.



-1**1	0**1	1**1
-1**0	0**0	1**0
-1**-1	0**-1	1**-1

;A Comment About Comments

It is important to leave yourself comments and notes that remind you what your procedure does and how it works. The semicolon mark (;) signals the computer to ignore everything that follows on the line. Try this command. It does nothing.

```
;THIS IS IGNORED (ENTER)
```

LOGO ignores everything after the (;) mark.

You can put comments into procedures like this:

```
TO COMMENT
;THIS IS A COMMENT ☜ LOGO ignores this line
END
```

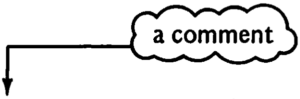
When you type COMMENT, the computer has nothing to do but END.

Comments can help you, and others, to understand your programs.

Count Your Joys

You can use the stick to change things. Here is a procedure called JCOUNT that uses the procedure J1X to increase and decrease the number "T1X. JCOUNT is really four procedures. TOTAL does the main work for

JCOUNT. TOTAL uses JOB which we will change to do various jobs. TOTAL also uses J1X which returns -1, 0 or 1 from joystick 1. Here are all the procedures you need:



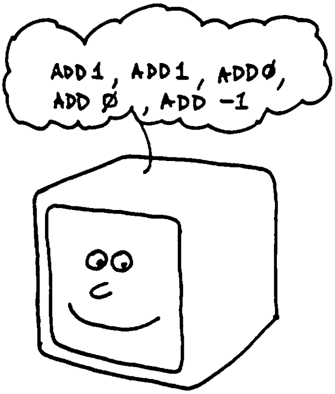
```
TO JCOUNT                 ; sets starting value of T1X
MAKE "T1X 0               ; "T1X starts at 0
TOTAL                     ; this procedure does the work
END

TO TOTAL                  ; totals J1X numbers into T1X
MAKE "T1X ;T1X + J1X      ; add J1X to :T1X
JOB                       ; this procedure will do some job
TOTAL                     ; use procedure TOTAL again
END

TO J1X                    ; returns -1, 0, 1 from joystick 1
OP (JOY 1)/4 - 1
END

TO JOB                    ; the job to be done
PRINT :T1X                ; the first job
END
```

Enter the procedures JCOUNT, TOTAL, J1X and JOB. Get back to LOGO command mode and type:

```
JCOUNT
```

You see numbers run down the screen something like this:

```
0
0
1
2    Stick to right
3    adds 1
4
4    Stick in middle
4    adds 0
3    Stick to left
2    adds -1
1
```



- Change JOB to this:

```
TO JOB
CB :T1X
END
```

Now JCOUNT makes the stick control the background color. Remember that CB 7 colors the background CYAN blue again.

- Change JOB to this:

```
TO JOB
PC :T1X
END
```

Now the joystick puts characters on the screen.

- Change JOB to this:

```
TO JOB
REPEAT :T1X [ TYPE"* ]
PRINT [ ]
END
```

Now JCOUNT makes the stick control how long a line of ***** is printed.

- Change JOB to this:

```
TO JOB
CARRY :BALL
SC 1
SH :T1X
SS 100
END
```

Now JCOUNT makes the joystick create sprites that zip off in directions determined by the joystick.

Joyful Sprite

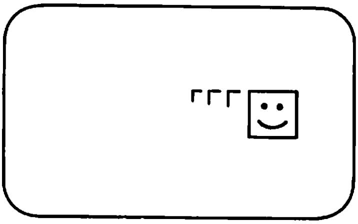
You can use the number :T1X to control sprite 1. Change JCOUNT so that it alerts sprite 1 like this:

```
TO JCOUNT
MAKE "T1X 0
TELL 1 CARRY :BOX SC 1 HOME           ; alert sprite 1
TOTAL
END
```

Now change JOB to control the sprite's position, like this:

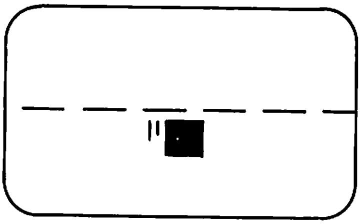
```
TO JOB
SX :T1X                   ; set the X coordinate to :T1X
PRINT :T1X                  ; print the value of :T1X
END
```

Get back to command mode. Type JCOUNT. Now the joystick moves the sprite left and right on the screen, and prints out the value of :T1X.

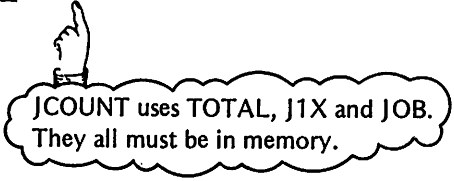


Turtle Work

This next procedure shows how the sprite, the turtle and the joystick can work together. The turtle draws a reference line, and the joystick moves the sprite along the line. This procedure represents numbers like a thermometer. The position of the sprite along the line indicates the value of a variable number.



```
TO GRAPH
CS
TELL TURTLE SH 90 SXY (-120) 0
REPEAT 60 [FD 2 PU FD 2 PD ]           ; dashed line
JCOUNT         JCOUNT must be in memory
END
```

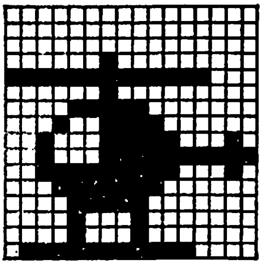

JCOUNT uses TOTAL, J1X and JOB.
They all must be in memory.

Another Direction

You can control the sprite's motion up and down by adding one extra line to each of the procedures JCOUNT, TOTAL and JOB. Here are all the procedures you need, with the changes made:

```
TO JCOUNT
MAKE "T1X 0
MAKE "T1Y 0 ☜ Add this
TELL 1 CARRY :BOX SC 1 HOME
TOTAL
END

TO TOTAL
MAKE "T1X :T1X + J1X
MAKE "T1Y :T1Y + J1Y ☜ Add this
JOB
TOTAL
END

TO JOB
SX :T1X
SY :T1Y ☜ Add this
END

TO J1X
OP (JOY 1)/4 -1
END

TO J1Y
OP (JOY 1) - (JOY 1)/4*4 -1
END
```

Now the joystick moves the sprite anywhere on the screen. Here is a more interesting shape for sprite-shape 1:

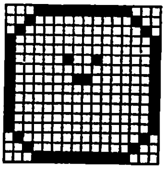


- The speed numbers can't be bigger than 127 or smaller than -127. Add two lines to TOTAL to reset big speed numbers:

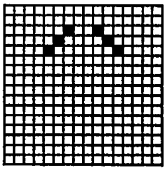
```
TO TOTAL
MAKE "T1X :T1X + J1X
MAKE "T1Y :T1Y + J1Y
IF :T1Y > 127 MAKE "T1Y 127            ; if too big, set back
IF :T1Y < (-127) MAKE "T1Y (-127)        ; if too small, set back
JOB
TOTAL
END
```

Faces Again

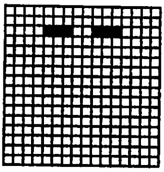
Remember the face that changes its expression in Chapter 3? Now you can use the joystick to control the expression on the face. First, you need to make the seven simple face part shapes. Here they are again:



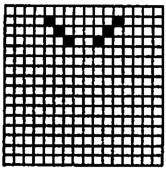
MS 10



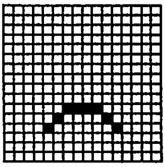
MS 11



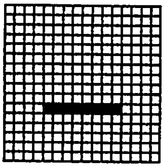
MS 12



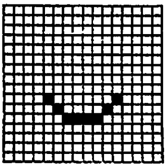
MS 13



MS 14



MS 15



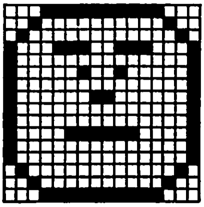
MS 16

Now, type this procedure:

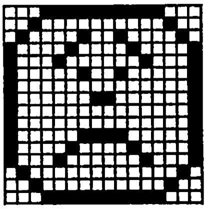
```
TO EMOTE
TELL [1 2 3 ] SC 1 HOME
TELL 1 CARRY 10
MOVE
END
```

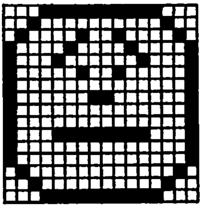
```
TO MOVE
TELL 2 CARRY 12 + J1X ☜ 11, 12, 13
TELL 3 CARRY 15 + J1Y ☜ 14, 15, 16
MOVE
END
```

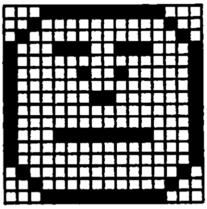
Now, when you type EMOTE a face appears on the screen:



When you move the joystick, the expression of the face changes. There are nine different faces.







Dialogue

You need two faces to have a dialogue. A modest addition will make each of the joysticks control a face. Joystick 2 needs its own procedure just like J1X and J1Y. Add these new procedures:

```
TO J2X
OP (JOY 2)/4-1
END

TO J2Y
OP (JOY 2)-(JOY 2)/4*4-1
END
```

Now change EMOTE and MOVE so they use joystick 2:

```
TO EMOTE
TELL [1 2 3 4 5 6 ] SC 1 HOME
TELL [4 5 6 ] SXY 32 0        ; move face 2 to right
TELL [1 4 ] CARRY 10 ☜ Changed to include face 2
MOVE
END

TO MOVE
TELL 2 CARRY 12 + J1X ☜ 11, 12, 13
TELL 5 CARRY 12 + J2X ☜ 11, 12, 13
TELL 3 CARRY 15 + J1Y ☜ 14, 15, 16
TELL 6 CARRY 15 + J2Y ☜ 14, 15, 16
MOVE
END
```

Now you and your face can talk to a friend and the friend's face. There are a lot of possibilities here.

- Find names for all the expressions.
- Have a dialogue with someone you usually can't talk to.
- Make up a two-face play.

GO to a Labeled Line

So far in this book we have gotten the computer to repeat a command or procedure in two different ways. First, we used the REPEAT command. Second, we had the procedure call itself at the end. There is one other, less often used method to make the computer repeat a command or list of commands. The GO command tells the computer to go to some line. Here is an example:

```
TO LOOP
TOP:                ; a line label
PRINT [INFINITE LOOP ]
GO "TOP             ; go to the labeled line
END
```

When you type LOOP the procedure starts to run. When the computer comes to TOP:, it does nothing. The colon mark (:) at the end tells the computer this is the name of the line, a line label. When the computer comes to the command GO "TOP, it goes to the line labeled TOP: and continues down the command list from that point. The computer will repeat this forever, or until you press the (FCTN)(BACK) key.

- You can stop the infinite loop in various ways. Here is one method for stopping the loop. Add this line to LOOP:, right after TOP:

```
IF RC? THEN STOP
```

Now the procedure stops if you press any key.

Here is another procedure that goes to a labeled line. This procedure continually prints the value of joystick 1.

```
TO JOYREAD
AGAIN:          ; a labeled line
PRINT JOY 1
GO "AGAIN
END
```

Experienced programmers try to use the GO command as little as possible. GO commands seem to make procedures harder to understand, and harder to fix. Here is a horrible example:

```
TO SPAGETTI           ; a horrible example
A:
GO "B
GO "A
PRINT [WHAT? ]
B: PRINT [WHERE? ]
GO "A
END
```

Can you figure out what this procedure does?

Summary of Chapter 8

Chapter 8 was about the joysticks, the joystick number code, comments and the GO to a labeled line command. In this chapter you learned:

- That JOY 1 is a procedure that returns numbers from joystick 1
- That JOY 2 returns numbers for joystick 2
- That there are nine joystick positions
- That the joysticks send the nine numbers 0, 1, 2, 4, 5, 6, 8, 9, 10
- How to write a procedure J1X that returns -1, 0 or 1 to show the X direction movement of joystick 1
- How to write a procedure J1Y that returns -1, 0 or 1 to show the Y direction movement of joystick 1
- How to write a procedure JCOUNT that makes the joysticks cause the numbers T1X and T1Y to increase and decrease
- How to control the motion of a sprite with the joystick

- How to make the joystick control the expression on a face
- That the semicolon mark (;) tells LOGO to ignore what follows
- How to use the semicolon mark to put notes and comments in procedures
- That lines can be named with a label
- That TOP: would be a typical line label
- That a line label ends with the colon mark (:)
- That the command GO "TOP tells LOGO to go to the line labeled TOP:

Self-Test—Chapter 8

1. Each box, below, represents a joystick position. Write, in each box, the joystick number that goes with the position.

2. Write a command that repeatedly types the numbers from joystick 1.

____

3. Write a procedure called ANSWERS that makes the computer print YES when you move the stick right, and print NO when you move it left.

____

____

____

____

____

____

4. Write a procedure called ASK that makes the computer print lines of asterisk marks (*). The number of asterisk marks in the line should be the number from the joystick.

____

____

____

5. What will this command print?

```
PRINT 50/4*4
```

____

6. MAKE "X 103. Now, write a LOGO command that will print the remainder of :X divided by 13. Make the computer do the work.

____

7. What will this procedure print?

```
TO BURP
URP:
PRINT [BURP ]
GO "URP
END
```

____

8. What will this procedure print?

```
TO DINK
;PRINT [MUCH ADO ABOUT NOTHING ]
PRINT [THIS ] ;PRINT [AND THIS ]
END
```

____

9. Write a procedure called SUM that continually prints the sum of the number from joystick 1 and the number from joystick 2.

____

____

____

____

10. Write a procedure called WORKING that keeps printing the word WORKING forever. Use a labeled line and a GO command.

____

____

____

____

____

Answers

1.

2	6	10
1	5	9
0	4	8

2.

```
REPEAT -1 [PRINT JOY 1 ]
```

3.

```
TO ANSWERS
  MAKE "J JOY 1
  IF :J <3 THEN PRINT [NO ]
  IF :J >7 THEN PRINT [YES ]
  ANSWERS
  END
```

4.

```
TO ASK
  REPEAT (JOY 1) [TYPE [* ] ]
```

5. The command PRINT 50/4*4 prints the number 48. The computer first divides 50 by 4 to get 12. Then 12 is multiplied by 4 to get 48.

6.

```
PRINT :X - :X/13*13
```

7. The procedure prints BURP on each line forever. The command GO "URP tells the computer to go to the line labeled URP:.

8. The procedure prints the word THIS. The semicolon mark (;) tells the computer to skip line 1. The semicolon mark in the middle of line 2 tells the computer to skip the rest of the line.

9.

```
TO SUM
  PRINT (JOY 1) + (JOY 2)
  SUM
  END
```

10.

```
TO WORKING
  TOP:
  PRINT [WORKING ]
  GO "TOP
  END
```

9

A Word About Lists

The real power of LOGO lies in the way it handles words, numbers and lists of things. This chapter reveals the secrets of LOGO list processing. We can only say a little; you will discover much on your own. In this chapter you will learn:

- How a name is different from the thing it names
- How LOGO recognizes words, numbers and lists
- About the special list editor that puts your lists in proper form
- How to turn the list editor on and off
- That the FIRST and LAST procedures return the first or last object from a word, or from a list
- That the BUTLAST and BUTFIRST procedures return all but the first, or all but the last, objects in a word or list
- That the procedure WORD puts two words together to get a new word
- That the procedures FPUT and LPUT put an object into the first or the last place in a list
- That the procedure SENTENCE puts two lists together to make a new list
- That a list of commands can be RUN
- That THING?, WORD? and NUMBER? tell the truth about a name
- How to make LOGO even more powerful by adding the procedures WHILE, DO and FOR to its vocabulary

LOGO Objects

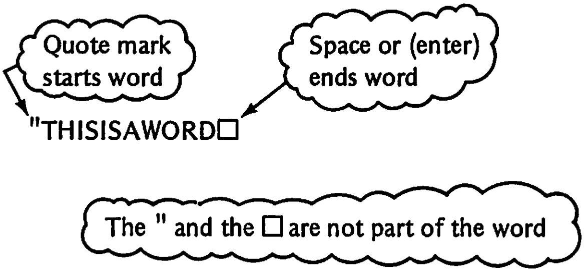
You have already met the 32 sprites, the turtle and the 256 tiles. LOGO has other objects that store data. There are three kinds of data objects in LOGO: words, numbers and lists. Much of the power of the LOGO language comes from the procedures for dealing with these data objects. There are procedures to build up data objects, and procedures to take data objects apart.

Words

LOGO words are much like ordinary words. A word is made of letters. Here are some examples of LOGO words:

"JOE, "A, "#5, "ABRACADABRA ☜ LOGO words

LOGO words always start with the quote mark ("). The quote mark (") tells the computer that a word is starting. A word ends at the first space, or (ENTER) key.


Quote mark starts word
Space or (enter) ends word
"THISISAWORD□
The " and the □ are not part of the word

- Try this:

```
PRINT "THIS ONE
```

You see:

```
THIS
TELL ME WHAT TO DO WITH ONE
```

The computer found the word "THIS and printed it. After the space, the computer found ONE. That looks to the computer like a procedure call. There is no procedure ONE, so the computer is confused.

The starting quote mark ("), and the ending space, or (ENTER), are not part of the word. Try this:

```
PRINT "GONE
```

You see:

```
GONE
```

The computer knows that you want the word GONE printed. The computer doesn't print the quote mark (").

Numbers

LOGO numbers range from -32768 to 32767. The computer recognizes numbers because they start with a minus sign (-), or a digit 0 to 9. Try this:

```
PRINT 32
```

The computer recognizes that 32 is a number and prints it.

```
32
```

Now try this:

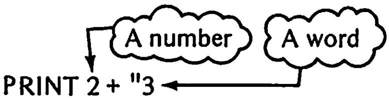
```
PRINT "32
```

The computer recognizes that "32 is a word and prints

```
32
```

The results look the same, but we shall see that they are quite different.

- Try this:



```
PRINT 2 + "3
```

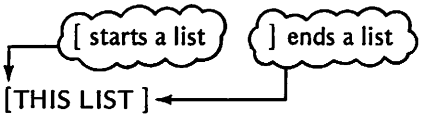
You see:

```
+ DOESN'T LIKE 3 AS INPUT
```

The computer can add a number to a number, but it doesn't know how to add the number 2 to the word denoted by "3.

Lists

Lists are the main data objects in LOGO. You can make lists of data objects. Here is an example of a list of words:



```
[THIS LIST ]
```

A LOGO list always starts with a square bracket ([) and ends with a matching square bracket (]). A list contains data objects separated by spaces. A list may contain words, numbers, lists, or even lists of lists of lists. There are lots of possibilities. Here are some examples of lists:

```
[ ]              ☜ The empty list
[A ]
[FISH FOWL ]
[[A ] [FISH FOWL ] 32 ]
[[[A ] B ] C ]
```

The computer keeps track of the brackets and can determine which bracket is the end of the list. The number of left brackets must equal the number of right brackets.

Naming Names

You never get the person mixed up with the person's name, do you? Computers are not as clever as people. Computers need to be told when you mean the name of a thing, and when you mean the thing to which the name refers. Here is an example:

First, use the MAKE procedure to link the name "BUBBLES with the list [BLUB BLUB].

```
MAKE "BUBBLES [BLUB BLUB ]
```

Now, we can talk about the name "BUBBLES, or we can talk about the list :BUBBLES.

"BUBBLES is a word that serves as a name

:BUBBLES is the thing to which the name refers

- Here is a little quiz. First link some names with some things, like this:

  ```
  MAKE "A [B ]
  MAKE "B 3
  MAKE "C "C
  ```

  Now tell what thing is meant:

  a. :A
  b. :B
  c. :C

ANSWERS: [B], 3, "C ("C is the name of itself.)

Reading A List

The READLINE procedure, RL for short, could as well be called the READ-LIST procedure. The RL procedure returns a list that you type at the keyboard. You used the RL procedure earlier in the book. Try this to remind yourself how it works:

```
RL (ENTER)
```

The computer stops and prompts you with a (>) mark. We typed:

```
WORKING HARD
```

The computer typed:

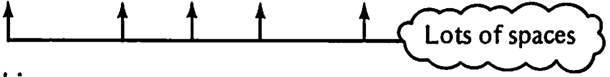
```
TELL ME WHAT TO DO WITH [WORKING HARD ]
```

Notice that the computer accepted what we typed, and stored it in a list.

The List Editor

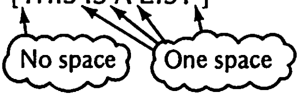
When you type a list, the computer may change the list a bit. Type this:

```
[     THIS    IS    A    LIST     ]
```



You see this:

```
TELL ME WHAT TO DO WITH [THIS IS A LIST ]
```



The computer edits your list and puts it into standard form. Here is how the computer edits your lists:

- Take out all spaces at the beginning of the list.
- Add one space at the end of the list.
- Replace any row of spaces with one space.

This is a handy feature of LOGO. You don't need to deal with all those extra spaces. The computer does it for you.

The list editor also treats the arithmetic marks +, –, *, /, <, >, = differently than other characters. The list editor always puts a space on both sides of these characters. Try this:

```
[A+B*C ]
```

No spaces

You see:

Spaces

```
TELL ME WHAT TO DO WITH [A + B * C ]
```

The computer has broken the word "A+B*C into 5 words "A, "+, "B, "* and "C.

Space Toggle (')

The single quote mark (') has a very special effect on the computer. It tells the computer to stop checking for spaces. The first (') turns the space editor off, the second (') turns it back on again.

If you want to keep the word just the way you typed it, then you can use the single quote mark to stop the editor.

- Type this:

  Single quote

  ```
  PRINT ['A+B' ]
  ```

  ☞ Turn off the list editor

  You see:

  ```
  A+B
  ```

  The single quote mark (') turned off the space editor. No spaces were added to the word. You can also use the single quote mark to put spaces in words. Try this:

  Single quote stops the editor

  ```
  PRINT "'THIS IS ONE WORD'
  ```

  You see

  ```
  THIS IS ONE WORD
  ```

  The single quote mark (') turned off the space editor. The word doesn't end until the (ENTER) key is pressed. The spaces stay in the word.

- Tell what these will print:

  ```
  PRINT "THIS' IS A 'WORD
  PRINT " ' '
  PRINT "'A □□□B'
  PRINT [ THIS 'THIS AND THAT' MORE ]
  ```

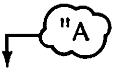
ANSWERS: THIS IS A WORD, a space, A ☐ ☐☐B, THIS THIS AND THAT MORE (This last list really has only three words in it. 'THIS AND THAT' is a single word.)

First and Last

LOGO has two procedures that let you get letters from words and objects from lists. The procedure FIRST, or F for short, gets the first letter from a word, or the first object from a list. Give it a try:

```
F "ABC
```

You see:



```
TELL ME WHAT TO DO WITH A
```

The computer got the first letter "A from the word "ABC.

The FIRST procedure also gets the first object from a list. Try this:

```
F [AA BB ]
```

You see:

```
TELL ME WHAT TO DO WITH AA
```

The computer got the first word "AA from the list [AA BB].

The procedure LAST gets the last letter from a word, or the last object from a list. Give this a try:

```
LAST "ABC
```

You see:

```
TELL ME WHAT TO DO WITH C
```

The computer got the last letter "C from the word "ABC.

The procedure LAST also gets the last object from a list. Try this:

```
LAST [AA BB ]
```

You see:

```
TELL ME WHAT TO DO WITH BB
```

The computer got the last word "BB from the list [AA BB].

BUTFIRST and BUTLAST

The procedure BUTFIRST, or BF for short, gets all but the first letter of a word, or all but the first objects from a list. Try this:

```
BF "ABC
```

You see:

```
TELL ME WHAT TO DO WITH "BC
```

The computer got all but the first letter of the word "ABC.

The procedure BUTFIRST gets all but the first objects from a list. BUTLAST returns a list containing the objects. Try this:

```
BF [AA BB CC ]
```

You see:

```
TELL ME WHAT TO DO WITH [BB CC ]
```

The procedure BF returned a list containing all but the first objects in the list.

The procedure BUTLAST does just what you expect. BUTLAST returns all but the last letter in a word, or all but the last objects in a list. Try this:

```
BL "ABC
```

You see:

```
TELL ME WHAT TO DO WITH "AB
```

The computer returned a word containing all but the last letters of the word "ABC.

The procedure BUTLAST returns a list containing all but the last objects from a list. Try this:

```
BL [AA BB CC ]
```

You see:

```
TELL ME WHAT TO DO WITH [AA BB ]
```

The procedure BL returned all but the last word in the list.

- Tell what each of these will return:

F "BLIMP __________

LAST "PLUNK __________

BF "LUNK __________

BL "MONK __________

F [HI HO] __________

LAST [HEE HAW] __________

BF [2 B OR NOT] ________

BL [U R A QT] ________

ANSWERS: "B, "K, "UNK, "MON, "HI, "HAW, [B OR NOT], [U R A]

Plucker

Here is a short procedure that illustrates how LOGO can take a list apart:

```
TO PLUCK
MAKE "A [THIS IS A SHORT LIST ]            ; list to be plucked
TOP:            ; a labeled line
IF :A = [ ] THEN STOP          ; check if :A is empty
PRINT :A               ; print the list
MAKE "A BF :A               ; make "A shorter
GO "TOP                ; go do it again
END
```

Get back to command mode and type PLUCK. You see:

```
THIS IS A SHORT LIST
IS A SHORT LIST
A SHORT LIST
SHORT LIST
LIST
```

When the list :A is finally equal to the empty list [ ], then the procedure stops.

- Change the first line of PLUCK so that "A is a line you type at the keyboard.
- Change the 5th line to MAKE "A BL :A. Now what does the procedure do?

ANSWERS: MAKE "A RL. When the fifth line is changed, the last word is dropped from list each time the line is printed.

- Here is a procedure that takes the list that you type and prints it backwards:

```
TO BACKWORDS
MAKE "A RL         ; read a line from the keyboard
TOP:            ; a labeled line
IF :A = [ ] THEN PRINT [ ] STOP         ; if done, stop
TYPE LAST :A          ; type the last word in list :A
MAKE "A BL :A            ; take the last word from :A
GO "TOP            ; keep going
END
```

- This next procedure, called GET, gets an object from a list. If you want to get the third object from the list :MYLIST, you just type

```
GET 3 :MYLIST
```

The procedure will return the third object in the list :MYLIST. Here is the procedure:

```
TO GET "N "LIST              ; :N is the number, :LIST is the list
MAKE "C 0              ; :C will keep count
TOP:            ; a labeled line
IF :LIST = [ ] THEN OP [ ] STOP           ; if empty, stop
MAKE "C :C + 1              ; increase the counter
IF :C = :N THEN OP F :LIST STOP            ; output the Nth object
MAKE "LIST BF :LIST              ; remove first object from list
GO "TOP
END
```

Now, get back to command mode and type

```
GET 2 [AA BB CC ]
```

You see:

```
TELL ME WHAT TO DO WITH BB
```

The computer got the second object in the list.

Try this:

```
MAKE "X [ONE TWO THREE ]
GET 3 :X
```

You see

```
TELL ME WHAT TO DO WITH THREE
```

The computer got the third object from the list :X.

New Words from Old

LOGO has procedures to build new words out of old words. The procedure WORD accepts two words and then joins them together to get a new, longer word. Try this:

```
WORD "THIS "THAT
```

You see:

```
TELL ME WHAT TO DO WITH THISTHAT
```

The computer put "THIS and "THAT together to get "THISTHAT.

- Tell what the result will be in each case:

a. WORD "O "FF
b. WORD "ANY "ONE
c. WORD WORD "A "B "C

ANSWERS: "OFF, "ANYONE, "ABC

Build

Here is a procedure that reads characters from the keyboard, adds them to a word, then prints the word when the asterisk mark (*) is typed.

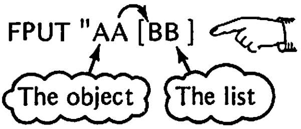
```
TO BUILD
MAKE "W"' '            ; :W starts with only a space
TOP:           ; a labeled line
MAKE "X RC             ; :X is the key pressed
IF :X = "* THEN PRINT :W STOP          ; if "*, print and stop
MAKE "W WORD :X :W          ; put :X on the front of :W
GO "TOP            ; keep going
END
```

Building Lists

LOGO has three procedures for building lists. The procedures are called FPUT, LPUT, and SENTENCE. The procedures FPUT and LPUT allow you to put new objects into the first place or the last place in a list. The procedure SENTENCE, or SE for short, lets you put two lists together to get a new list.

FPUT

FPUT means FIRST-PUT. The procedure FPUT puts an object into the first place in a list. Give this a try:



FPUT "AA [BB] put "AA first in the list

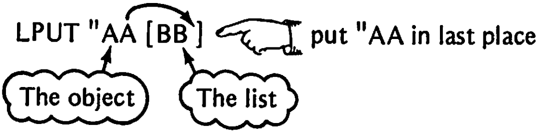
You see:

```
TELL ME WHAT TO DO WITH [AA BB ]
```

The computer put the word "AA into first place in the list [BB].

LPUT

LPUT means LAST-PUT. The procedure LPUT puts an object into the last place in a list. Give this a try:



You see:

```
TELL ME WHAT TO DO WITH [BB AA ]
```

The computer put the word "AA into the last place in the list [BB].

You can put words, numbers or lists into a list. Try this:

```
FPUT [AA ] [BB ]
```

You see:

```
TELL ME WHAT TO DO WITH [[AA ] BB ]
```

The computer put the object [AA] into the list.

- Tell what each of these returns:

```
FPUT "THIS [ONE ]
LPUT "THESE [ONE OF ]
LPUT 8 [1 2 4 ]
LPUT [3 [4 ] ] [1 [2 ] ]
```

ANSWERS: [THIS ONE], [ONE OF THESE], [1 2 4 8], [1 [2] [3 [4]]].

SENTENCE

The procedure SENTENCE, or SE for short, puts two lists together to get a new list. Try this:

```
SE [THIS ONE ] [AND THAT ]
```

You see:

```
TELL ME WHAT TO DO WITH [THIS ONE AND THAT ]
```

The computer put the two lists together to make a new list.

- What will these return?

```
SE [1 2 3] [9 8 7]
SE [RED [YELLOW] ] [ [ [ BLUE ] ] ]
SE SE [HEIGHT] [WIDTH] [LENGTH]
```

ANSWERS: [1 2 3 9 8 7], [RED [YELLOW] [[BLUE]]], [HEIGHT WIDTH LENGTH].

List Tool List

Here is a list of all the procedures introduced so far in this chapter:

F	FIRST	gets first object in a word or list
LAST	LAST	gets last object in a word or list
BF	BUTFIRST	gets list of all but the first object from a list
BL	BUTLAST	gets list of all but the last object from a list
WORD	WORD	puts two words together to get new word
FPUT	FIRST-PUT	puts an object in the first place in a list
LPUT	LAST-PUT	puts an object in the last place in a list
SE	SENTENCE	puts two lists together to get a new list

A Short Quiz

Here are some exercises to practice the procedures. Tell what each of these return:

a. WORD F "GO BF "FOOD
b. SE BF [SOME VERY] BL [NICE ICE]
c. FPUT "GREAT BF [MOB JOB]
d. SE [SOME KIND] FPUT "OF [JOB]
e. FPUT "THE LPUT "END [ ]

ANSWERS: a. WORD (F "GO) (BF "FOOD) is "GOOD, b. SE (BF [SOME VERY]) (BL [NICE ICE]) is [VERY NICE], c. [GREAT JOB], d. [SOME KIND OF JOB], e. [THE END]

Run A List

A list may contain instructions for the computer to perform.

```
[PRINT "HI ]
```

If you want to run a list of instructions, you use the RUN command. Try this:

```
RUN [PRINT "HI ]
```

You see:

```
HI
```

The computer ran the list of instructions and printed HI.

- Try this:

```
MAKE "X [PRINT "HI ]
RUN :X
```

You see:

```
HI
```

The computer ran the list :X, and printed HI.

- This next procedure, called RUN3, that accepts a list of commands and runs the list 3 times.

```
TO RUN3 "X              ; :X will be a list of commands
REPEAT 3 [RUN :X ]            ; run the list :X three times
END
```

We typed

```
RUN3 [TYPE "HI ]
```

We saw this:

```
HIHIHI?
```

TYPE keeps the prompt on the same line

The computer accepted the list [TYPE "HI] and called it "X. Then the computer ran the list :X three times.

- Try this. What do you think will happen?

```
RUN3 [RUN3 [TYPE "* ] ]
```

While Do

You can make the LOGO language even better than it is. You can add sophisticated statements that make programming faster and easier. This next example shows how to add a procedure called WHILE to LOGO's vocabulary. The WHILE command gives you a way to tell the computer to do some list of procedures while some condition remains true. The LOGO command TEST :EX will test whether the expression :EX is TRUE or FALSE.

```
TO WHILE "EX "ST          ; an expression, and a statement
TOP:
RUN SE [TEST ] :EX          ; RUN [TEST    ]  :EX goes here
IFF STOP                ; if TEST is false, stop
IFT RUN :ST                 ; if TEST is true, RUN the statement
GO "TOP
END
```

We want to do a little job. While the number :X is less than 5, we want to type :X, and then add 1 to :X. Here is how we use WHILE to do the job. Give it a try.

```
MAKE "X 0
WHILE [:X < 5 ] [TYPE :X MAKE "X :X+1 ]
```

We saw:

```
              the prompt
0 1 2 3 4?
```

- Try this command. While the key you press is not "X, the computer keeps asking for another letter, but stops when you press "X.

```
WHILE [NOT RC = "X ] [PRINT [TYPE ANOTHER KEY ] ]
```

- If you want to do something forever, you can use an expression that is always true. Try this:

```
WHILE [2 < 3 ] [BEEP WAIT 5 NOBEEP WAIT 5 ]
```

Do Until

This next procedure, called DO, is similar to the WHILE procedure. The DO procedure does a list of commands until some condition is true. The WHILE procedure tests the condition before running the list, but the DO procedure does the list of commands, then tests to see if the condition is true.

Both procedures are useful. Here is the procedure DO:

```
TO DO "ST "EX
TOP:
RUN :ST
RUN SE [TEST ] :EX
IFF GO "TOP
IFT STOP
END
```

We typed this:

Do this
Until this

```
DO [MAKE "A RC PRINT :A ] [:A = "X ]
```

The computer printed each letter we typed, until we typed an "X.

- Try this:

```
MAKE "N 10
DO [PRINT :N * :N MAKE "N :N - 1 ] [N = 0 ]
```

- Write a command to make the computer type NICE JOB until RC? is true.

____

- Write a command to type a random digit until the digit 7 occurs.

____

ANSWERS: The first two commands above will print the squares numbers from 100 down to 0. The command DO [TYPE [NICE JOB]] [RC?] will do the job. The command DO [MAKE "R RANDOM TYPE :R] [:R = 7] will work.

THING?

You can make names refer to things. It is sometimes useful to have a way to tell whether a word has been used to name something. The procedure THING? tells you whether a word stands for a thing or not. Try this:

```
THING? "QXXQ            ; this is not a name of anything
```

The computer types:

```
TELL ME WHAT TO DO WITH FALSE      ; FALSE means word unused
```

The computer found that the word "QXXQ is not the name of anything. The procedure returns "FALSE.

Now, try this:

```
MAKE "QXXQ [THING STRING ]      ; "QXXQ stands for something
THING? "QXXQ          ; check if "QXXQ stands for something
```

The computer types:

```
TELL ME WHAT TO DO WITH TRUE          ; "QXXQ does stand
                                        for something
```

- Try THING? on these words:

  "NORTH, "BLUE, "BALL

- Try THING? on a few words of your own.

THING

The procedure THING is not the same as THING?. The procedure THING returns the thing for which the word stands. Try this:

```
THING "QXXQ
```

The computer types:

```
TELL ME WHAT TO DO WITH [THING STRING ]
```

The procedure THING found that "QXXQ is the name of the string [STRING THING]. The procedure returned the string.

- Try THING on these words:

  "EAST, "RED, "ALL

- Try THING on some words of your own.

WORD? NUMBER?

You can find out if a name refers to a word, or a number with the procedures WORD? and NUMBER?. The procedure WORD? tells if a word is the name of a word. Try this:

```
WORD? "XQQX          ; not used yet, doesn't refer to a word
```

You see:

```
TELL ME WHAT TO DO WITH FALSE       ; FALSE, not name of word
```

The word "XQQX is not the name of a word. The computer returns "FALSE.

Try this:

```
NUMBER? "XQQX
```

You see:

```
TELL ME WHAT TO DO WITH FALSE     ; FALSE, not name of number
```

The word "XQQX is not the name of a number. The computer returns "FALSE.

Try this:

```
MAKE "W "AWORD
MAKE "N 35
TYPE WORD? "W
```

You see:

```
TRUE
```

Now, try this:

```
TYPE NUMBER? "N
```

You see:

```
TRUE
```

The computer found that "W names a word and that "N names a number. In each case the computer returns TRUE.

For . . To . . Do

There are many programming situations where you will want to perform some calculations for a range of numbers. For example, you might want to print the doubles of all the numbers N in the range from N = 1 to N = 5. This situation is so common that most programming languages have a special command just for such situations. This next example shows that LOGO is powerful enough to create such a command. The command will work like this: if you want to print the doubles of the numbers N for values of N from 1 to 5, then you will type

```
FOR "N [1 5 ] [TYPE 2*N ]
```

The result will be the list of numbers 2 4 6 8 10.

To make this procedure work, it will be necessary to pass the name of the index variable "N, to the procedure in such a way that it can be used in the command statement.

A Pre-Procedure Puzzle

This little quiz is to remind you how things are named and referenced.

- If you MAKE "I "INDEX, and MAKE :I 1, then what are the following?

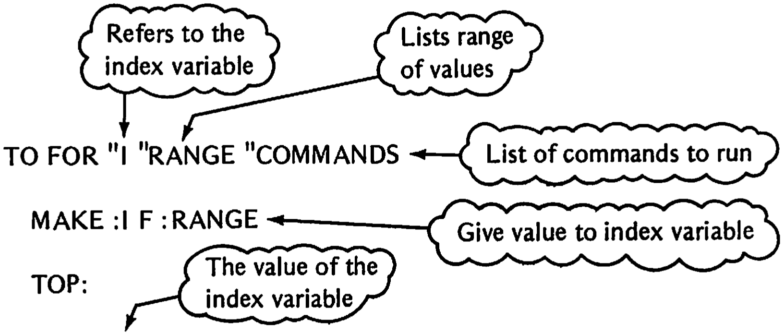
  a. :I __________

  b. :INDEX __________

  c. THING :I __________

ANSWERS: a. :I is "INDEX. b. When you MAKE :I 1, it is the same as MAKE "INDEX 1. Hence, :INDEX is 1. c. THING :I is the same as THING "INDEX. THING "INDEX is the same as :INDEX, which is 1.

Here is the procedure FOR that uses some clever naming and referring to accomplish a useful task:


Refers to the index variable
Lists range of values
List of commands to run
Give value to index variable
The value of the index variable

```
TO FOR "I "RANGE "COMMANDS
 MAKE :I F :RANGE
 TOP:
 IF (THING :I) > (LAST :RANGE) THEN STOP
 RUN :COMMANDS
 MAKE :I (THING :I) + 1
 GO "TOP

 END
```

Enter this program and give it a try on these tasks:

- Try this command:

  FOR "X [0 10] [TYPE :X TYPE ' ' PRINT :X*:X]

- Try this command:

  FOR "NUM [10 20] [PRINT (2*:X*:X + :X +1)]

Summary of Chapter 9

This chapter was about words, numbers, and lists. In this chapter you learned:

- That words always start with a quote mark (")
- That words end with the first space, or the (ENTER) key
- That numbers start with a minus sign (-), or a digit 0 to 9
- That a list starts with a square bracket ([) and ends with a matching bracket (])
- That a list may contain words, numbers or lists
- That the READLINE procedure, RL, returns a list
- That the list editor removes extra spaces from lists
- That the editor puts spaces around [,], +, -, *, /, =, <, >
- That the single quote mark (') turns the editor off and on
- That the FIRST and LAST procedures return the first or last objects from words or lists
- That BF and BL return all but the first, or all but the last, object from a word or list
- How to use the list procedures to pluck words off a list
- How to make a procedure called GET that gets an object from specified place in a list
- That WORD puts two words together to get a new word
- That FPUT and LPUT put an object into first or last place in a list
- That SENTENCE puts two lists together to get a new list
- That a list of instructions can be RUN
- How to create the useful procedures WHILE, DO and FOR
- That THING?, WORD? and NUMBER? tell to what a name refers

Self-Test—Chapter 9

1. Tell in each case below whether the thing is a word, a number, a list or a procedure:

   a. [AND] ______; b. "THIS ______; c. "[SOMETHING] ______;
   d. ["WORD] ; ______; e. -0 ______; f. "'ONE TWO THREE' ______

Tell what each of the following will print:

2. PRINT "THIS AND THAT ______________________________
3. PRINT "3 + "2 ______________________________
4. PRINT [START □□□ FINISH] ______________________________
5. ·PRINT F [FIRST AMENDMENT] ______________________________
6. PRINT LAST [LIFE LIBERTY HAPPINESS] ________________
7. PRINT BF [ONE PERSON, ONE VOTE] ________________
8. PRINT F BF [WASTE NOT, WANT NOT] ________________
9. PRINT FPUT "ONE [WAY] ______________________________
10. PRINT LPUT "ONE [WAY] ______________________________
11. PRINT SE [LOOK BEFORE] [YOU LEAP] ________________

In questions 12 to 15, assume that "X is [PRINT :X], "Y is [REPEAT].

12. PRINT THING "X ______________________________
13. RUN :X ______________________________
14. RUN SE :Y [3 [TYPE "*]] ______________________________
15. FOR [0 3] [TYPE :N TYPE"' ' PRINT :N * :N] ________________

Answers

1. a. [AND] is a list; b. "THIS is a word; c. "[SOMETHING] is a word; d. ["WORD] is a list; e. -0 is a number; f. "'ONE TWO THREE' is a word.
2. THIS

   TELL ME HOW TO AND
3. + DOESN'T LIKE "3 AS INPUT
4. START FINISH
5. FIRST
6. HAPPINESS
7. PERSON, ONE VOTE

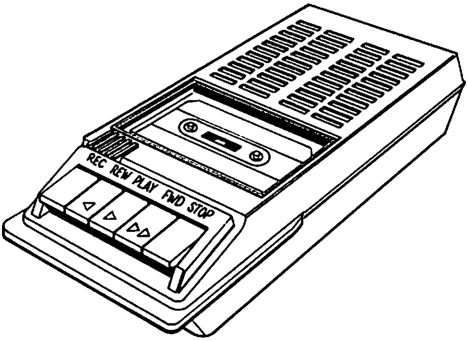
8. NOT,
9. ONE WAY
10. WAY ONE
11. LOOK BEFORE YOU LEAP
12. PRINT :X
13. RUN :X becomes RUN [PRINT :X], but :X is [PRINT :X]. The computer prints PRINT :X.
14. RUN SE :Y [3 [TYPE "*]] becomes RUN [REPEAT 3 [TYPE "*]]. The computer types ***.
15.

```
0 0
1 1
2 4
3 9
```

10

Saving and Recalling

To get the most out of your creative computer work, you will want to save your procedures, sprite-shapes and character-tiles. To save and recall your work, you will need either a cassette tape recorder, or a TI disk drive, disk drive controller and DISK MANAGER command module.



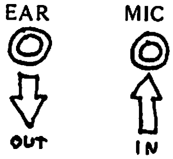
In this chapter you will learn:

- Some kinds of cassette recorders that work with your TI99/4A computer
- How to hook up the cassette recorder to your computer
- How to SAVE procedures and shapes on the cassette recorder
- How to RECALL procedures and shapes from the cassette recorder
- How to hook up the TI disk drive and controller
- How to use the DISK MANAGER module to initialize a floppy disk
- How to SAVE procedures and shapes on the TI disk drive
- How to RECALL procedures and shapes from the TI disk drive

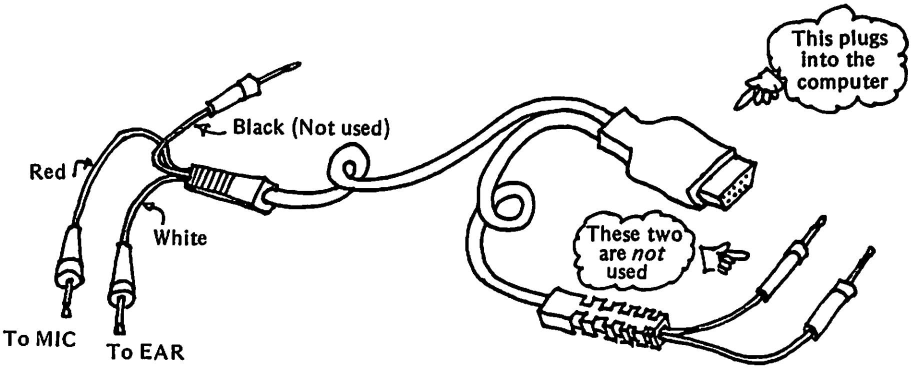
Cassette Recorders

There are many different cassette tape recorders that will work with your TI99 computer. If you already have a cassette tape recorder, there is a good chance that it will work. We'll give general directions that should work with any cassette recorder you have.

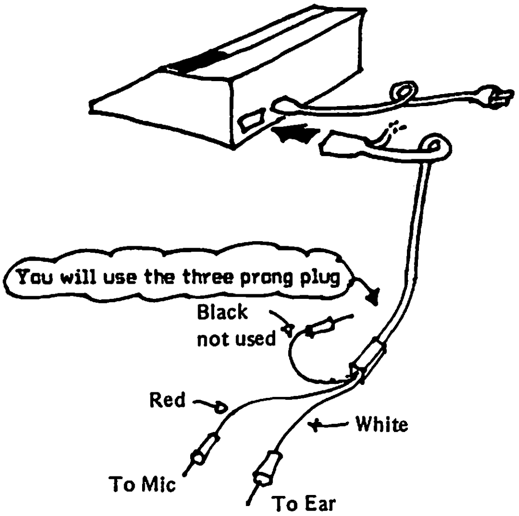
Your cassette recorder will have a hole where you can plug in a microphone, and another hole where you can plug in an earphone, or an external speaker. The microphone hole is marked MICROPHONE, or possibly MIC. The earphone hole is marked EARPHONE, or EAR, or possibly EXTERNAL, or EXT. Signals come out of the earphone hole, and signals go into the microphone hole.



A special set of wires, called the cable, came packed with your computer, or can be ordered from Texas Instruments. The cable will carry signals between the computer and the cassette recorder.



The cable has a fat plug at one end. The fat plug fits into the computer in back, on the same side as the command module, like this:



The red plug brings a signal from the computer into the cassette recorder. The RED plug goes into the MICROPHONE hole. The white plug takes a signal from the cassette recorder to the computer. The WHITE plug goes into the EARPHONE, or EXT hole.

Plug the power cord into the wall socket. There is one last important thing to do. Set the volume control indicator to middle position. This works best on most recorders, but not all. You may have to experiment later to find just the right setting for your recorder.

Cassette Tapes

Only some cassette tapes will work for computer work. Cheap or low quality cassette tapes will not give proper results. You must use the best quality tapes for recording computer data. You can buy computer data cassettes at most computer stores. You can also use high quality audio cassette tapes. Here are some that will work:

Maxell; UD type
Scotch brand; Master type
TDK brand; SA, MAVERICK, AD, or D types
Verbatim; Digital Cassette type

Short tapes are best. Computer data cassettes come in 10-minute lengths. That's a good length for most purposes. Audio cassettes come in longer lengths. Try to get a 20-minute or 30-minute tape. Longer tapes take forever to wind and rewind.

Saving Your Procedures

Turn on the memory module and the computer. Get LOGO command mode. You will need a program to save. Type this:

```
TO CHECK
PRINT "WORKING
END
```

Get back to LOGO command mode and type this:

```
SAVE
```

The screen changes and you see this menu:

```
            SAVE

PRESS    FOR
  1    PROCEDURES
  2    SHAPES AND TILES
  3    BOTH 1 AND 2

PRESS 'BACK' FOR TI LOGO
```

Press 1 to save a procedure. You see:

```
           DEVICE

PRESS    FOR
  1    CASSETTE
  2    DISKETTE
  3    THERMAL PRINTER

PRESS 'BACK' FOR TI LOGO
SAVING–PROCEDURES
```

Press 1 to save on the cassette recorder. You see, at the bottom of the screen:

```
* REWIND CASSETTE TAPE CS1
THEN PRESS ENTER
```

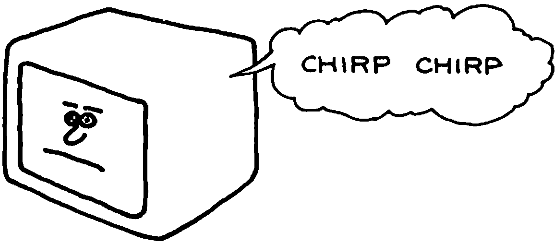
Make sure a cassette tape is in the recorder. Press the rewind key on the cassette recorder and let the tape rewind to the beginning. If your cassette player has a tape position counter, set the counter to 000. Now, press the (ENTER) key. You see two new lines appear at the bottom of the screen:

```
* PRESS CASSETTE RECORD CS1
THEN PRESS ENTER
```

This is a reminder to set the cassette to record. On most cassette recorders you must push the RECORD key and the PLAY key at the same time. The keys will lock down. The cassette recorder will start to record. Press the (ENTER) key on the computer to signal the computer to begin the SAVE procedure. You see this at the bottom of the screen:

```
* RECORDING
```

If the volume is turned up on your TV monitor, you can hear the data being sent from the computer to the cassette recorder. You hear a chirping sound.



When the recording is complete you see:

```
* PRESS CASSETTE STOP CS1
THEN PRESS ENTER
```

This is a reminder to press the STOP key on the cassette recorder. When you press (ENTER) you see:

```
* CHECK TAPE (Y OR N)?
```

Do you want to check and make sure that the recording matches the data stored in the computer's memory? Be safe. Type Y. You see:

```
* REWIND CASSETTE TAPE CS1
THEN PRESS ENTER
```

Press the REWIND key on the cassette recorder. When the tape is completely rewound, press the STOP key. Now press the (ENTER) key on the computer. You see:

```
* PRESS CASSETTE PLAY CS1
THEN PRESS ENTER
```

This is a reminder to press the cassette recorder's PLAY key so that it will play back the recorded data. The cassette recorder will begin to turn. Press the computer's (ENTER) key. The computer will start to compare the data on the tape with the data in the computer memory. You see:

```
* CHECKING
```

If the volume is up on your TV monitor, you can hear the data being played back.

If your procedure is correctly recorded, then you see this:

```
* DATA OK
PRESS CASSETTE STOP CS1
THEN PRESS ENTER
```

When you STOP the cassette and press the computer's (ENTER) key, you return to LOGO command mode. You see the familiar question mark prompt:

```
?_
```

If your procedure did not record correctly, then you will see something like this:

```
* ERROR
PRESS R TO RECORD
PRESS C TO CHECK
PRESS E TO EXIT
```

If you type R, the computer will lead you through the save procedure again. If you type C, the computer will lead you through the check procedure again. If you type E, the computer will lead you back to LOGO command mode.

Some Things That Help

- Check that everything is plugged in properly.
- Use good quality cassette tape.

- Adjust the volume control. Experiment. Turn the volume up one notch and try again. Adjust the volume down one notch and try again. The middle position is usually best, but not always.
- Turn the tone control, if any, to treble, or high.
- Shout at your computer. This doesn't make the computer work any better, but you'll feel better.

Recalling Procedures

The procedure CHECK is now stored on the cassette tape. You can recall the procedure at any time. Let's erase CHECK from the computer's memory. Type:

```
ERASE CHECK
```

The computer erases the procedure CHECK from its memory. To make sure the procedure is gone, type:

```
CHECK
```

You see:

```
TELL ME HOW TO CHECK
```

The computer doesn't have CHECK in its memory. Now, recall the procedure from the cassette tape. Type

```
RECALL
```

The screen changes, and you see the RECALL menu:

```
          RECALL

PRESS      FOR
   1    PROCEDURES
   2    SHAPES AND TILES
   3    BOTH 1 AND 2

PRESS 'BACK' FOR TI LOGO
```

Press 1 to recall a procedure. You see:

```
            DEVICE

PRESS      FOR
  1     CASSETTE
  2     DISKETTE
  3     THERMAL PRINTER

PRESS 'BACK' FOR TI LOGO
RECALLING– PROCEDURES
```

Press 1 to recall from the cassette recorder. You see, at the bottom of the screen:

```
* REWIND CASSETTE TAPE  CS1
THEN PRESS ENTER
```

Rewind the cassette tape to the position where you saved the procedure. In this case, the position is 000, the very beginning of the tape. When you press the computer's (ENTER) key you see:

```
* PRESS CASSETTE PLAY CS1
THEN PRESS ENTER
```

Press the cassette recorder's PLAY key. The recorder will start. Press the (ENTER) key to tell the computer to begin reading from the tape. You see:

```
* READING
```

When the computer is done reading the data from the cassette tape, you see:

```
* DATA OK

* PRESS CASSETTE STOP CS1
THEN PRESS (ENTER)
```

Press the cassette recorder's STOP key. When you press the computer's (ENTER) key, it returns to LOGO command mode and you see the familiar question mark prompt:

```
?_
```

Your procedure is now back in the computer's memory. Type:

```
CHECK
```

You see:

```
WORKING
```

If the computer does not read the tape successfully, then you see this:

```
* ERROR
PRESS R TO READ
PRESS C TO CHECK
PRESS E TO EXIT
```

If you press R, the computer will lead you through the recall procedure. If you press C, the computer will lead you through the checking procedure. If you press E, the computer will lead you back to LOGO command mode.

Some Things That May Help

- Check that everything is plugged in properly.
- Check that a good quality cassette tape is loaded in the recorder.
- Check that the volume setting is in the middle range.
- Check that the tone control, if any, is at high, or treble.
- Try higher and lower volume control settings for your cassette recorder. Experiment to find what setting works best.
- Say nice things to your computer. This won't help, but it may calm you.
- If it doesn't work the first time, try it again. It is probably some little thing that you will do right the next time.

Some Quick Tips

The steps for saving and recalling data using the cassette recorder appear at first to be complex. But, the steps are simple and you can speed things up by being smarter than the computer. Let's save a new procedure, and take a few shortcuts while we're at it. Type this little procedure:

```
TO SHORT
PRINT [SHORT WORKS ]
END
```

Get back to LOGO command mode and type

```
SAVE
```

Press 1 to save a procedure. Press 1 to save on cassette. The computer reminds you to prepare the cassette recorder:

```
* REWIND CASSETTE TAPE  CS1
THEN PRESS ENTER
```

Prepare the cassette recorder like this:

- Use the REWIND and FAST FORWARD, and PLAY keys to position the tape at tape position 100. This time we'll record in the middle of the tape. You might want to jot the tape position number down so you won't forget where the data is saved.
- Press down the RECORD and PLAY buttons. They will lock down, and the cassette recorder will start to record.
- Press the enter key once. The computer will prompt you to press the RECORD key, but you have already done that. Press the (ENTER) a second time and the computer begins to record your data.

After a little practice, this whole process takes only a moment.

Recalling a procedure from the cassette is done like the save. Type

```
RECALL
```

Now, type 1, and then 1 again, to save a procedure on the cassette. Rewind the cassette tape to position 100, and press down the cassette's PLAY key. Now, press the computer's (ENTER) key, then press the (ENTER) key a second time. You see

```
READING
```

Check to see that the procedure is in the computer. Type

```
PP          print procedure names
```

You should see:

```
SHORT
```

If you type SHORT, you see:

```
SHORT WORKS
```

Many Procedures

If you have many procedures stored in the computer's memory, they will all be saved together on the cassette recorder. If you only want to save one or two procedures, erase the unwanted procedures before using the SAVE command. When you recall from the cassette recorder, all the procedures that you saved together are recalled together. The procedures that you recall will not disturb procedures that are already in the computer's memory.

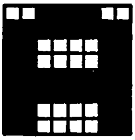
- Write a procedure called MYPRO that does some little task. Save MYPRO and SHORT at position 150 on the cassette tape. Erase both MYPRO and SHORT from the computer's memory. Now recall the procedures.

Saving Shapes

You save sprite-shapes and character-tiles in much the same way that you save procedures. You must have something to save. Make a new tile for character 65. That's the letter A. Type this:

```
MC 65          ; make character 65
```

You see the screen change. The computer is now in MAKE CHARACTER mode. We changed character-tile 65 to a fancy letter A, like this:



You might like to make your own version. When you are done, get back to LOGO command mode, and type:

```
SAVE
```

This time you wish to save a shape. Type 2, to indicate that you are saving a shape. Then, type 1 to indicate that you are saving on the cassette recorder. Rewind the cassette tape to tape position 200, and press down the RECORD and PLAY keys. The cassette begins to record. Press the computer's (ENTER) key once. Then press the (ENTER) key once again. The computer records all the 32 sprite-shapes, and all 256 character-tiles on the cassette recorder. When the recording is done, you see:

```
* DATA OK
PRESS CASSETTE STOP  CS1
THEN PRESS ENTER
```

Stop the cassette recorder and press the computer's (ENTER) key. Your character tile is recorded.

Recalling Shapes

Before recalling the character-tile that you just saved on the cassette tape, clear your computer's memory by turning the computer off. Now, turn the computer back on, and get to LOGO command mode. To recall the shape from the cassette recorder, type:

```
RECALL
```

You wish to recall a shape from the cassette recorder. Type 2, then type 1. Rewind the cassette recorder to position 200, where the shape is stored. Press down the PLAY key on the recorder. The recorder begins to play. Press the computer's (ENTER) key, then press it once again. The computer prints

```
READING
```

Watch the screen. Notice what happens to the A's on the screen. As soon as the A character is read into the computer's memory, all the A's on the screen change to the new shape. When the reading is finished, you see:

```
* DATA OK
PRESS CASSETTE STOP CS1 ☜ The new A
THEN PRESS ENTER
```

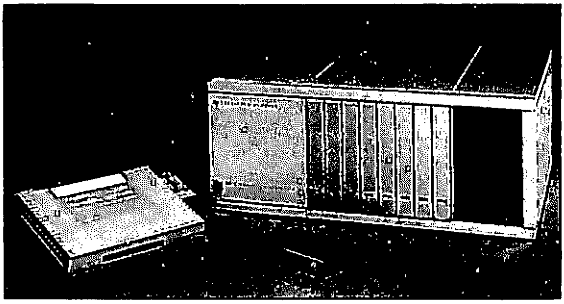
All the sprite-shapes and the character-tiles in the computer's memory are those that you saved earlier on the cassette tape.

- Create a design of your own for sprite-shape 10. Save the shape. Recall the shape. Now you are an expert.

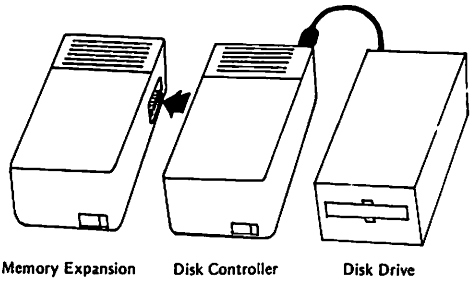
Setting Up the Disk Drive

If you have a floppy disk drive, and a disk drive controller, and the DISK MANAGER command module, then you can SAVE and RECALL easily and quickly. For this section you will also need a new blank floppy disk. The disks you need are *single-sided, single-density, soft-sectored*. Luckily, that's the usual kind.

The disk drive controller is a small computer that manages the flow of information between the disk drives and the TI 99 computer. If your system uses a TI Peripheral Expansion System, then the disk drive controller will be on a card that fits into one of the expansion slots.



If your system used the 32K memory expansion unit, then the disk drive controller will be enclosed in its own box, and will plug into the side of the memory expansion unit.

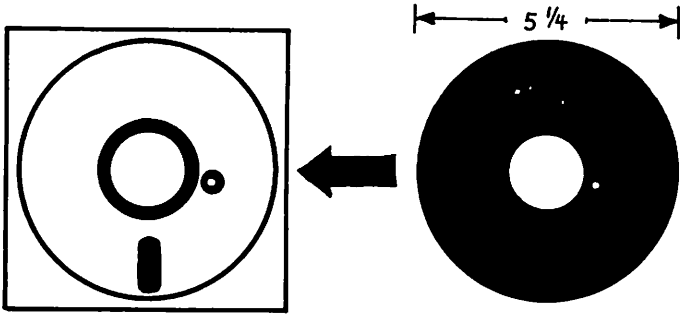


Each of the parts of the disk drive system has its own power cord. Be sure the cords are plugged into the wall socket.

You are ready to SAVE and RECALL from the disk drive.

Floppy Disks

Your disk drive stores its information on floppy disks.



A floppy disk is coated with magnetic material, similar to the material on audio cassette tapes. Your TI disk drive uses disks that are 5¼ inches wide. The disk drive records on one side of the disk in low, or single density. You can use any good floppy disk that is 5¼ inches wide. Be careful with your floppy disks. Here are some don'ts:

- Don't put your disk near any magnets, or motors or TV sets. These all have strong electric/magnetic fields that may erase your disk.
- Don't let your floppy disk get hot. Floppy disks can melt and bend.
- Don't let anything get the disk surface dirty, or greasy. Don't even touch the surface with your fingers.

This sounds like a lot of don'ts. With just a little care you will never have any trouble at all with your disks.

Initialize Your Disk

Before you can use your floppy disk, the computer must check its surface, and print magnetic reference marks on the disk. The reference marks will guide the computer later when you save and recall from the disk. The DISK MANAGER command module handles the job of initializing your disks.

The DISK MANAGER Command Module

The DISK MANAGER command module looks just like the LOGO command module, except that it says DISK MANAGER on the front. This command module tells the computer how to perform many useful tasks on the disk drive. Put in the DISK MANAGER command module. Now turn on your computer system in the usual way, peripherals first, then the computer. The disk drive may spin for a moment and its red light may flash. On the screen you see this menu:

```
DISK MANAGER
1 FILE COMMANDS
2 DISK COMMANDS
3 DISK TESTS
4 SET ALL COMMANDS FOR
SINGLE DISK PROCESSING

YOUR CHOICE?
```

The computer is asking you which of these procedures you wish to use. We will only use the first one of these many useful procedures of the disk manager. Type the number 1, and then press (ENTER). You see this:

```
DISK COMMANDS
1 CATALOG DISK
2 BACKUP DISK
3 MODIFY DISK NAME
4 INITIALIZE NEW DISK

YOUR CHOICE?
```

You are going to do choice 4, INITIALIZE NEW DISK. Press the number 4, and then press the (ENTER) key. You see this:

```
INITIALIZE NEW DISK
MASTER DISK [1-3] ?
```

Type the number 1, and press the (ENTER) key. You see

```
NEW DISKNAME?_________
```

Let's call the new disk DISK1. The name will fill in the blanks as you type and you see:

```
NEW DISKNAME?DISK1
```

Press (ENTER) and you see:

```
40 TRACKS(Y/N)?
```

Most floppy disks allow 40 tracks of information to be stored on the disk. But, some brands will only store 35. It will usually be marked on the outside label on your disk. Press the Y key (unless you have 35 track disks). The computer will give you a chance to make any last minute changes to all this information. You see:

```
SCREEN IS COMPLETE
PRESS: PROC'D, REDO, BEGIN, OR BACK
```

If the screen is not right, press the (REDO) key [that's FCTN)(8)]. Otherwise, press the (PROC'D) key [that's (FCTN)(6)]. The red disk light comes on, and you see this for a moment:

```
INITIALIZE NEW DISK
WORKING. . . .PLEASE WAIT
```

The computer will now check the surface of the disk for imperfections, and will also write magnetic reference marks on the disk. The computer will count on the screen as it progresses with its work. You see messages like this:

```
1<WORKING. . . .PLEASE WAIT
```

Finally the computer prints:

```
359<WORKING. . . .PLEASE WAIT
```

Then you see:

```
DSK1 – DISKNAME = DISK1
AVAILABLE=358 USED = 0
```

If the disk has some bad areas, some of the disk is marked USED, and will not be available.

Initialize Every Blank Disk

The initialization process you just learned must be performed on every new blank disk. The initialization procedure checks that your disk is good, marks any bad spots and writes magnetic reference marks on the disk. Your disk will not work unless it has been initialized. You will find it most convenient to initialize many disks at once and put them away for later use.

To get back to the DISK MANAGER menu, press the (BEGIN) key [that's (FCTN)(5)]. You can initialize more disks or stop. Let's see what the LOGO command module does with the new disk.

Back to The LOGO Module

We are done with the DISK MANAGER module. Take it out of the computer. Put the LOGO command module into the computer. Now we'll save and recall some procedures and shapes on the disk drives.

Saving on the Disk Drive

You will find it quick and easy to SAVE procedures and shapes on your disk drive. First, type in a procedure to save. Here's a quick one to type:

```
TO CHECK
PRINT "WORKS
END
```

To save this procedure on your disk drive type:

```
SAVE
```

The screen changes and you see this menu:

```
             SAVE

PRESS    FOR
  1    PROCEDURES
  2    SHAPES AND TILES
  3    BOTH 1 AND 2

PRESS 'BACK' FOR TI LOGO
```

Press 1 to save a procedure. You see:

```
          DEVICE

PRESS   FOR
  1   CASSETTE
  2   DISKETTE
  3   THERMAL PRINTER

PRESS 'BACK' FOR TI LOGO
SAVING– PROCEDURES
```

Press 2 to save on the floppy disk drive. The screen changes, and you see this menu:

```
TYPE FILE NAME, PRESS 'ENTER'
 OR
PRESS
'SPACE' TO REVIEW FILE NAMES
 OR
'BACK' FOR TI LOGO

SAVING– PROCEDURES
FILE NAME:________
```

File the program away under the name MYFILE1. As you type the file name MYFILE1, the bottom line changes to this:

```
FILE NAME:MYFILE1_
```

Press (ENTER), and the procedure is saved under the name MYFILE1.

Recalling from the Disk

Recalling procedures and shapes from the disk drive is quick and easy. Before recalling the procedure that you just saved, erase the procedure from the computer's memory. Type:

```
ERASE CHECK
```

To see that CHECK is really gone, type

```
CHECK
```

The computer doesn't have CHECK in its memory. You see:

```
TELL ME HOW TO CHECK
```

To recall a procedure from the disk, type:

```
RECALL
```

The screen changes and you see this menu:

```
           RECALL

PRESS    FOR
  1    PROCEDURES
  2    SHAPES AND TILES
  3    BOTH 1 AND 2

PRESS 'BACK' FOR TI LOGO
```

Press 1 to recall your procedure. You see:

```
RECALLING– PROCEDURES
```

Press 1 to recall a procedure. You see:

```
           DEVICE

PRESS    FOR
  1    CASSETTE
  2    DISKETTE
  3    THERMAL PRINTER

PRESS 'BACK' FOR TI LOGO
RECALLING– PROCEDURES
```

Press 2 to save on the floppy disk drive. The screen changes, and you see this menu:

```
TYPE FILE NAME, PRESS 'ENTER'
  OR
PRESS
'SPACE' TO REVIEW FILE NAMES
  OR
'BACK' FOR TI LOGO

RECALLING– PROCEDURES
FILE NAME:________
```

You filed the procedure away under the name MYFILE1. Type the file name MYFILE 1. The bottom line changes to this:

```
FILE NAME:MYFILE1_
```

Press (ENTER). The procedure CHECK, in MYFILE1, is recalled into the computer's memory. To see that the procedure is in memory, type

```
CHECK
```

You see:

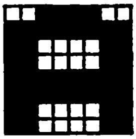
```
WORKS
```

Saving Shapes on the Disk

Sprite-shapes, and character-tiles are saved much like procedures. Make a character-tile to save. Type

```
MC 65,            That's the 'A' title
```

Change the shape of the A. Here's how our letter 'A' looks:



When your fancy 'A' is made, get back to LOGO command mode. Save your character-tile. Type:

```
SAVE
```

You see the menu as before. Type 2 to save shapes and tiles. Now, press 2 to save on the disk drive. Make up a file name. We chose the file name LETTER. When you press (ENTER), the disk drive purrs softly, and the new shapes and tiles are saved. The computer saves the current state of all the sprite-shapes and character tiles, including your fancy letter 'A'.

Recalling Shapes from the Disk

Before recalling the shape that you just saved, put the characters and shapes back to their original form. Turn off the computer, then turn it back on. Get into LOGO command mode. Now recall the shapes. Type

```
RECALL
```

You see the menu. Type 2 to recall a shape. Type 2 to recall from the disk. Type the file name LETTER, or whatever name you used. The disk drive purrs and the shapes on the file LETTER are read into the computer's memory. When the character-tile number 65 is read from the disk, all the 'A''s on the screen change.

Summary of Chapter 10

This chapter was about saving and recalling procedures and shapes on the cassette recorder and on the disk drive. In this chapter you learned:

THE CASSETTE RECORDER

- How to hook up the cassette recorder
- That the command SAVE starts the process to save information
- That you have three options: save just procedures, save just shapes, save both procedures and shapes
- That you have three device options: save on cassette, save on disk, save on the printer
- That the command RECALL starts the process to recall information
- That you have three options: recall just procedures, recall just shapes, recall both procedures and shapes
- That you have two-device options: recall from cassette, or recall from disk

THE DISK DRIVE

- How to use the DISK MANAGER command module to initialize a new disk

- How to set up the disk drive and controller
- How to save procedures and shapes on the disk drive
- How to recall procedures and shapes from the disk drive

Self-Test—Chapter 10

THE CASSETTE RECORDER

1. The cassette cable has two parts: one part has two plugs, the other part has three plugs. For the cassette recorder you use ________________
2. What is the color of the plug that brings signal into the cassette recorder? ________________
3. What is the color of the plug that takes signals out of the cassette recorder? ________________
4. Which hole, or jack, on the cassette recorder accepts signals into the computer? ________________
5. Which hole, or jack, on the cassette recorder sends signals out of the cassette recorder? ________________

THE DISK DRIVE

6. Which of the following types of floppy disk will work on your TI disk drive? a. An 8-inch, double-sided, double-density disk; b. An 8-inch, single-sided, single-density disk; c. A 5¼-inch, single-sided, single-density disk. ________________
7. How is information stored on the floppy disk? a. with a laser beam; b. by a needle in a groove; c. by magnetic variations in the disk surface.
8. What must be done to every new, blank disk? ________________
9. What command module is used to initialize a disk? ________________
10. What does the initialization procedure do to a disk? ________________

Answers

1. You use the three-prong part of the cable.
2. The red plug brings the signal to the cassette recorder.
3. The white plug takes signal from the cassette recorder to the computer.
4. The MICROPHONE hole accepts signals into the computer.

5. The EARPHONE hole sends signal from the cassette recorder to the computer.
6. A 5¼-inch, single-sided, single-density disk works in the TI disk drive.
7. Information is stored on the floppy disk in the form of magnetic variations in the surface of the disk.
8. Every new, blank disk must be initialized before it can be used.
9. The DISK MANAGER command module performs the initialization process.
10. The initialization process checks the surface of the disk for bad spots. It marks the bad spots. It also writes magnetic reference marks on the surface.

Appendix

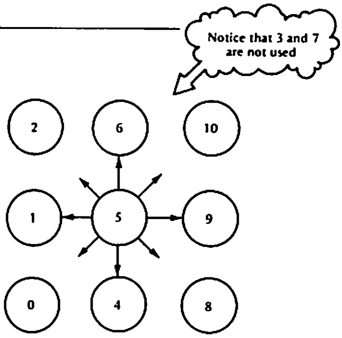
Logo Summary

LOGO Data Objects

```
WORDS          "A , "ABBA
NUMBERS        INTEGER -32768 TO 32767
LISTS          [ ], [A ], [A CATS 9 LIVES [LIST ] ]
```

Input

```
JOY 1   READ JOYSTICK 1

     PRINT JOY 1
     PRINT JOY 2
```



```
RC?  READCHAR?      RETURNS "TRUE, IF ANY KEY IS PRESSED.
                    RC? STAYS "TRUE UNTIL THE CHARACTER IS USED.

     PRINT RC?
     IF RC? THEN PRINT RC

RC   READCHAR       READS A SINGLE CHARACTER FROM KEYBOARD.

     RC             RETURNS VALUE LIKE "B
     PRINT RC       PRINTS A SINGLE CHARACTER
     MAKE "X RC

RL   READLINE       READS A LIST FROM THE KEYBOARD. (CR) TERMINATES.

     RL             RETURNS A VALUE LIKE [THIS IS A LIST ]
     PRINT RL       PRINTS A LIST LIKE   THIS IS A LIST
     MAKE "X RL
```

RECALL	ENTERS RECALL MODE. RECALL PROCEDURES AND SHAPES FROM CASSETTE OR DISK

Output

	BEEP	STARTS TONE
	NO BEEP	STOPS TONE
OP	OUTPUT	DEFINE OUTPUT OF A PROCEDURE

```
     TO BLAT
     MAKE "X "BLAT
     OP "X
     END
```

	PRINT	PRINTS LINE WITH (CR).

```
     PRINT "HI            PRINTS HI
     PRINT [HI HO]        PRINTS  HI HO
     PRINT CONTENTS       PRINTS OUTPUT OF THE PROCEDURE CONTENTS
     PRINT :X             PRINTS VALUE OF THE VARIABLE :X
```

PC	PRINTCHAR	TYPES CHARACTER BY NUMBER; CURSOR STAYS ON LINE

```
     PRINTCHAR 5
     PC 5                 PRINTS CHARACTER-TILE NUMBER 5
```

PT	PUTTILE	PUTS TILE AT COLUMN AND LINE

```
     PUTTILE 5 10 20
     PT 5 10 20           PRINTS TILE 5 IN COLUMN 10, LINE 20
```

	SAVE	ENTERS THE SAVE MODE. SAVES TO CASSETTE, DISK OR PRINTER
	TYPE	PRINT WITH NO (CR). CURSOR STAYS ON SAME LINE

```
     TYPE "HI
     TYPE [THIS AND THAT]
     TYPE CONTENTS
     TYPE :X
```

Timing

WAIT	CAUSES COMPUTER TO PAUSE FOR SPECIFIED TIME

```
     WAIT 60          WAITS FOR 60 JIFFIES, OR 1 SECOND
```

Management

BYE	EXIT LOGO
CONTENTS	RETURNS LIST OF ALL WORDS USED

```
     PRINT CONTENTS

CONTINUE       CONTINUE PAUSED PROGRAM

EDIT           ENTER EDIT MODE

     EDIT MYPRO

ERASE          ERASE PROCEDURE FROM MEMORY

     ERASE MYPRO

PA   PRINT ALL     PRINTS ALL NAMES, PRIMITIVES, PROCEDURES

PN   PRINT NAMES

PO   PRINT OUT     PRINT OUT PROCEDURE

     PO MYPRO

PP   PRINT PROCEDURES   PRINT PROCEDURE NAMES

RECALL             ENTER RECALL MODE TO RECALL PROCEDURE,TILE,SPRITE.
SAVE               ENTER SAVE MODE TO SAVE PROCEDURE, TILE, SPRINT.

TB   TRACEBACK

TO   ENTER PROCEDURE EDIT MODE
```

Definition and Assignment

```
CALL           CALL AN OBJECT BY A NAME

     CALL 5 "X
     CALL [THIS AND THAT] "BABBLE
     CALL "THIS "THAT

DE   DEFINE         DEFINES PROCEDURE BY LIST OF INSTRUCTIONS.

     DEFINE "BLAT [[ ] [PRINT "BLOT ] ]
     DEFINE "DOUBLE [ [X ][OUTPUT 2*:X ] ]

MAKE           ASSIGN NAME TO AN OBJECT

     MAKE "X 5
     MAKE "BABBLE [THIS AND THAT]
     MAKE "THIS "THAT
     MAKE "X "'ONE WORD'      ' ALLOWS SPACES IN A WORD

MC   MAKECHAR      ENTER MAKE-CHARACTER-TILE MODE.

     MAKECHAR 30     MAKE CHARACTER-TILE 30

MS   MAKESHAPE     ENTER MAKE-SPRITE-SHAPE MODE.

     MAKESHAPE 10      MAKE SPRITE-SHAPE 10
```

Procedures

```
TO              ENTER PROCEDURE-DEFINE MODE

     TO PUNT
     TYPE "PUNT
     END

     TO DOUBLE X
     OUTPUT 2*:X
     END

DEFINE          DEFINE PROCEDURE BY STRING OF COMMANDS.

     DEFINE "DOUBLE [[X ][OUTPUT 2*:X ]

ERASE           ERASE PROCEDURE

     ERASE "DOUBLE

PA   PRINT ALL PRIMITIVES, PROCEDURES, AND NAMES.

PN   PRINT ALL VARIABLE NAMES

PO   PRINT OUT SPECIFIED PROCEDURE

     PO "DOUBLE

PP   PRINT PROCEDURE NAMES.

TEXT       RETURNS PROCEDURE AS A LIST

     TEXT "DOUBLE       THIS RETURNS THE LIST
                        [[X ][OUTPUT 2*:X ] ]

     PRINT TEXT "DOUBLE      THIS PRINTS  [X ][OUTPUT 2*:X ]

OP   OUTPUT             SENDS DATA OBJECT OUT OF A PROCEDURE

     OP :X
     OP [1 2 3]
     OP "THIS

EDIT   ENTERS PROCEDURE EDITOR MODE

     EDIT DOUBLE

RUN        CAUSES COMPUTER TO PERFORM A LIST OF COMMANDS

     RUN [ PRINT "HI PRINT "HO ]

EACH [   YN  ]        PERFORMS THE LIST OF COMMANDS FOR EACH OF THE
                      CURRENT SPRITES.  USES THE CURRENT SPRITE'S
                      NUMBER YN
     EACH [PRINT YN ]             PRINTS NUMBERS OF CURRENT SPRITES

THING      INDICATES INDIRECT REFERENCE

     MAKE "X "Y
     MAKE "Y "Z
     MAKE "Z "W
     PRINT THING :X          THIS PRINTS  Z

     PRINT THING THING :X         THIS PRINTS W
```

Arithmetic Functions

DIFFERENCE ____ ____	RETURNS THE FIRST MINUS THE SECOND
5 - 3	RETURNS THE FIRST MINUS THE SECOND
PRODUCT ____ ____	RETURNS THE PRODUCT OF THE TWO NUMBERS
5 * 3	RETURNS THE PRODUCT OF THE TWO NUMBERS
QUOTIENT ____ ____	RETURNS THE QUOTIENT OF THE FIRST BY THE SECOND
5 / 3	RETURNS THE QUOTIENT OF THE FIRST BY THE SECOND
SUM ____ ____	RETURNS THE SUM OF THE TWO NUMBERS
5 + 3	RETURNS THE SUM OF THE TWO NUMBERS
RANDOM	RETURNS RANDOM 0,1,2,3,4,5,6,7,8,9

Branching and Control

IF ____ ____	IF FIRST CONDITION IS "TRUE THEN DO SECOND
IF ____ THEN ____	
IF ____ THEN ____ ELSE ____	
GO	BRANCH TO THE LINE LABELED PUNT:
PUNT:	A LABEL
GO "PUNT	BRANCH TO LABELED LINE
TEST ____	TEST IF CONDITION IS "TRUE, SET FLAG. THE FLAG IS USED BY IFT AND IFF
IFT ____	DO IF TEST FLAG IS "TRUE
IFF ____	DO IF TEST FLAG IS "FALSE

```
TEST ( :X > 0 )
IFT PRINT "YES
IFF PRINT "FALSE
```

REPEAT	REPEAT A LIST

```
REPEAT 5[PRINT "THIS]
```

"True/"False Functions

```
BOTH ____ ____        RETURNS "TRUE IF BOTH TRUE, ELSE "FALSE

     IF BOTH (:X > 1) (:X < 5 ) THEN PRINT "OK

EITHER ____ ____      RETURNS "TRUE IF EITHER IS TRUE, ELSE "FALSE

     IF EITHER ( :X > 1 ) ( :X < 5 ) THEN PRINT "OK

GREATER ____ ____     RETURNS "TRUE IF FIRST IS GREATER THAN SECOND
                      ELSE "FALSE

     IF GREATER :X 2 THEN PRINT "OK

___ > ___             RETURNS "TRUE IF FIRST IS GREATER THAN SECOND,
                      ELSE "FALSE
                      SAME AS GREATER ___ ___

     IF :X > 2 THEN PRINT "OK


IS ____ _____         RETURNS "TRUE IF FIRST NUMBER, WORD OR LIST
                      IS EQUAL TO THE SECOND, ELSE "FALSE

     IF IS :X [1 2 ] THEN PRINT "OK

___ = ____            RETURNS "TRUE IF THE FIRST NUMBER, WORD OR LIST
                      IS EQUAL TO THE SECOND, ELSE "FALSE
                      SAME AS EQUALS _____ ______

     IF :X = 2 THEN PRINT "OK

LESS ____ ____        RETURNS "TRUE IF FIRST IS LESS THAN SECOND

     IF LESS :X 0 THEN PRINT "OK

____ < ____           RETURNS "TRUE IF FIRST IS LESS THAN SECOND

     IF :X < 0 THEN PRINT "OK

RC?                   RETURNS "FALSE UNTIL ANY KEY IS PRESSED, THEN
                      RETURNS "TRUE UNTIL CHARACTER IS USED.

     IF RC? THEN PRINT RC

NOT ____              CHANGES "TRUE TO "FALSE, "FALSE TO "TRUE

     IF NOT ( :X > 0 ) THEN PRINT "OK

NUMBER?          RETURNS "TRUE, IF INPUT WORD NAMES A NUMBER

     NUMBER? "ABBA           RETURNS "TRUE, IF "ABBA NAMES A NUMBER
                             RETURNS "FALSE OTHERWISE

THING?           RETURNS "TRUE, IF INPUT WORD IS A NAME OF SOMETHING

     THING? "ABBA            RETURNS "TRUE, IF "ABBA NAMES SOMETHING
                             RETURNS "FALSE OTHERWISE

WORD?            RETURNS "TRUE, IF INPUT WORD NAMES A WORD

     WORD? "ABBA             RETURNS "TRUE, IF "ABBA NAMES A WORD
                             RETURNS "FALSE OTHERWISE
```

List Processing

```
BF   BUTFIRST      RETURNS ALL BUT THE FIRST ELEMENT.

     BF "ABC                        RETURNS THE VALUE "BC
     PRINT BF "ABC                  PRINTS  BC
     BF [THIS AND THAT ]            RETURNS THE VALUE [AND THAT ]
     PRINT BF [THIS AND THAT]       PRINTS  AND THAT

BL   BUTLAST                  RETURNS ALL BUT THE LAST ELEMENT.

     BL "ABC                        RETURNS "AB
     PRINT BL "ABC                  PRINTS  AB
     BL [THIS AND THAT ]            RETURNS [THIS AND ]
     PRINT BL [THIS AND THAT]       PRINTS  THIS AND

F    FIRST              RETURNS FIRST ELEMENT

     F "ABC                         RETURNS "A
     PRINT F "ABC                   PRINTS  A
     F [THIS AND THAT ]             RETURNS [THIS ]
     PRINT F [THIS AND THAT]        PRINTS THIS

LAST               RETURNS LAST ELEMENT

     LAST "ABC                      RETURNS "C
     PRINT LAST "ABC                PRINTS C
     LAST [THIS AND THAT ]          RETURNS [THAT ]
     PRINT LAST [THIS AND THAT]     PRINTS  THAT

FPUT      FIRSTPUT         ADDS AN OBJECT TO THE FRONT OF A LIST

     FPUT "AB [CD]                  RETURNS [AB CD ]
     PRINT "AB [CD]                 PRINTS AB CD
     FPUT [AB][CD]                  RETURNS THE VALUE [[AB ] CD ]
     PRINT FPUT [AB][CD]            PRINTS [AB ] CD

LPUT      LASTPUT          ADDS AN OBJECT TO THE END OF A LIST

     LPUT "AB [CD]                  RETURNS  [CD AB ]
     PRINT LPUT "AB [CD]            PRINTS  CD AB
     LPUT [AB][CD]                  RETURNS  [CD [AB ] ]
     PRINT LPUT [AB][CD]            PRINTS  CD [AB ]

WORD      CONCATENATES WORDS

     WORD "THIS "THAT               RETURNS  "THISTHAT
     PRINT WORD "THIS "THAT         PRINTS THISTHAT

SE   SENTENCE       CONCATENATES LISTS

     SE [THIS][THAT]                RETURNS  [THIS THAT ]
     PRINT SE [THIS ] [THAT ]       PRINTS  THIS THAT

CALL               CALLS AN OBJECT BY A NAME

     CALL "THIS "X
     CALL [THIS AND THAT] "BABBLE

CN   CHARNUM         RETURNS TILE NUMBER OF A KEYBOARD CHARACTER

     TYPE CN "A         TYPES THE NUMBER 65
```

```
DE    DEFINE                DEFINES A PROCEDURE BY A LIST
                            THE FIRST ITEM IN THE LIST IS A LIST OF
                            INPUT VARIABLES.

      DEFINE "BLAT [[   ] [PRINT "BLAT PRINT "BLOT ] ]
      DEFINE "PUNT [[X ][ PRINT :X ] ]
      DEFINE "ADD [[X Y ][ MAKE "Y SUM :X :Y ][ OP :Y ] ]


RC    READCHAR        RETURNS CHARACTER FROM KEYBOARD.

      RC              RETURNS A VALUE LIKE "B
      PRINT RC        PRINTS A CHARACTER, LIKE B

RL    READLINE        RETURNS A LIST FROM KEYBOARD. (ENTER) TERMINATES.

      READLINE           THIS HAS A VALUE LIKE [THIS IS A LIST ]
      PRINT READLINE        THIS PRINTS SOMETHING LIKE    THIS IS A LIST

REPEAT                REPEATS A LIST

      REPEAT 5[FORWARD 10 RT 30]
      REPEAT -1 [ ]             REPEATS 65536 TIMES

RUN                   RUNS A LIST OF COMMANDS.

      RUN [PRINT "HI PRINT "HO ]

TEXT                  RETURNS PROCEDURE AS A LIST.

      TEXT "BLAT        THIS HAS A VALUE LIKE
                       [[  ][PRINT "BLAT ][OP "BLOT ] ]
                       EACH ITEM IN THE LIST IS A PROCEDURE LINE

      PRINT TEXT "BLAT    THIS PRINTS SOMETHING LIKE
                       [ ][PRINT "BLAT ][OP "BLOT ]
```

Message Sending

```
TELL            OPENS A CHANNEL TO MENTIONED OBJECT.

      TELL TURTLE        ALERTS THE TURTLE
      TELL 0             ALERTS SPRITE 0
      TELL [ 0 1 2 ]     ALERTS SPRITES 0, 1, AND 2.
      TELL :ALL          ALERTS ALL 32 SPRITES 0 TO 31
      TELL :X            ALERTS THE OBJECTS MENTIONED BY VARIABLE :X
      TELL BG            ALERTS THE BACKGROUND
      TELL TILE 3        ALERTS CHARACTER TILE NUMBER 3

EACH [    YN    ]         PERFORMS THE LIST OF COMMANDS FOR EACH OF
                          THE CURRENT SPRITES.  USES THE SPRITE'S
                          NUMBER YN

      EACH [PRINT YN SC YN ]
```

Predefined Variables

```
:ALL IS  [0 1 2 ... 31 ]

:CLEAR  IS 0
:PLANE  IS 1
:TRUCK  IS 2
:ROCKET IS 3
:BALL   IS 4
:BOX    IS 5

:CLEAR  IS 0
:BLACK  IS 1
:GREEN  IS 2
:LIME   IS 3
:BLUE   IS 4
:SKY    IS 5
:RED    IS 6
:CYAN   IS 7
:RUST   IS 8
:ORANGE IS 9
:YELLOW IS 10
:LEMON  IS 11
:OLIVE  IS 12
:PURPLE IS 13
:GRAY   IS 14
:WHITE  IS 15

:NORTH  IS 0
:EAST   IS 90
:SOUTH  IS 180
:WEST   IS 270
```

Turtle Commands

```
TELL TURTLE          ALERTS TURTLE TO ACCEPT COMMANDS

DOT                  DRAWS DOT AT SPECIFIED POSITION.

     DOT 10 20       PUTS A DOT AT POSITION 10 20

ST   SHOWTURTLE      MAKES TURTLE VISIBLE.

HT   HIDETURTLE      MAKES TURTLE INVISIBLE

NOTURTLE             EXITS TURTLE MODE.

WHERE                RETURNS LIST OF X Y COORDINATES AND HEADING
                     OF THE TURTLE IN TURTLE MODE

     PRINT WHERE

BK   BACK            MOVES THE TURTLE BACKWARDS

     BK 20

FD   FORWARD         MOVES THE TURTLE FORWARDS

     FD 20

LT   LEFT TURN       TURNS THE TURTLE TO THE LEFT

     LT 90
```

RT	RIGHT TURN	TURNS THE TURTLE TO THE RIGHT
	RT 90	
PU	PEN UP	STOPS THE TURTLE FROM DRAWING
PD	PEN DOWN	STARTS THE TURTLE DRAWING
PE	PEN ERASE	CHANGES THE PEN TO AN ERASER
PR	PEN REVERSE	THE PEN REVERSES FOREGROUND AND BACKGROUND COLORS

Shape and Color Commands

SC	SETCOLOR	SETS COLOR OF CURRENT TURTLE, TILE, BACKGROUND, OR SPRITES
	SC :RED SC 3	
	SC [3 8]	SETS FOREGROUND AND BACKGROUND COLORS OF CURRENT TILE
CARRY		CARRY SHAPE COMMAND FOR CURRENT SPRITES
	CARRY 5 CARRY :TRUCK	ACCEPTS NUMBERS 0 TO 255
LOOKLIKE		SAME AS CARRY.
	LOOKLIKE 5 LOOKLIKE :TRUCK	
CB	COLORBACKGROUND	SETS BACKGROUND COLOR
	CB 1 CB :BLACK	
CS	CLEARSCREEN	FILLS THE SCREEN WITH BLANK CHARACTER (TILE 32 ERASES ALL TILES EXCEPT 32 TO 95, ERASES TURTLE DRAWING. DOES NOT EFFECT SPRITES.
MC	MAKECHARACTER	ENTERS MAKE-CHARACTER TILE MODE THE CHARACTER TILES ARE NUMBERED FROM 0 TO 255
	MC 32	MAKE CHARACTER-TILE NUMBER 32 ACCEPTS NUMBERS 0 TO 255
MS	MAKESPRITE	ENTERS MAKE-SPRITE-SHAPE MODE THE SPRITE SHAPES ARE NUMBERED FROM 0 TO 25
	MS 6	MAKE SPRITE-SHAPE NUMBER 6 ACCEPTS NUMBERS 0 TO 25

State Reports

COLOR	RETURNS COLOR NUMBER OF TURTLE PEN, OR CURRENT SPRITE
PRINT COLOR	RETURNS NUMBER 0 TO 15
HEADING	RETURNS HEADING NUMBER OF CURRENT TURTLE OR SPRITE
PRINT HEADING	RETURNS NUMBER 0 TO 359
SHAPE	RETURNS SHAPE NUMBER OF CURRENT SPRITE
PRINT SHAPE	RETURNS NUMBER 0 TO 25
SPEED	RETURNS SPEED OF ACTIVE SPRITE
PRINT SPEED	RETURNS NUMBER 0 TO 127
WHERE	RETURNS LIST OF X Y COORDINATES AND HEADING OF TURTLE IN TURTLE MODE
PRINT WHERE	
WHO	RETURNS LIST OF ACTIVE SPRITES, TURTLE, OR TILE
PRINT WHO	
XCOR	RETURNS X COORDINATE OF CURRENT TURTLE OR SPRITE
PRINT XCOR	RETURNS NUMBER -120 TO 119
YCOR	RETURNS Y COORDINATE OF CURRENT TURTLE OR SPRITE
PRINT YCOR	RETURNS NUMBER -47 TO 96
XVEL	RETURNS X VELOCITY OF CURRENT SPRITE
PRINT YVEL	RETURNS NUMBER -127 TO 127
YVEL	RETURNS Y VELOCITY OF CURRENT SPRITE
PRINT YVEL	RETURNS NUMBER -127 TO 127
YN YOURNUMBER	USED WITH EACH
EACH [PRINT YN]	RETURNS NUMBER 0 TO 31 OF ACTIVE SPRITES

Relative Motion Commands

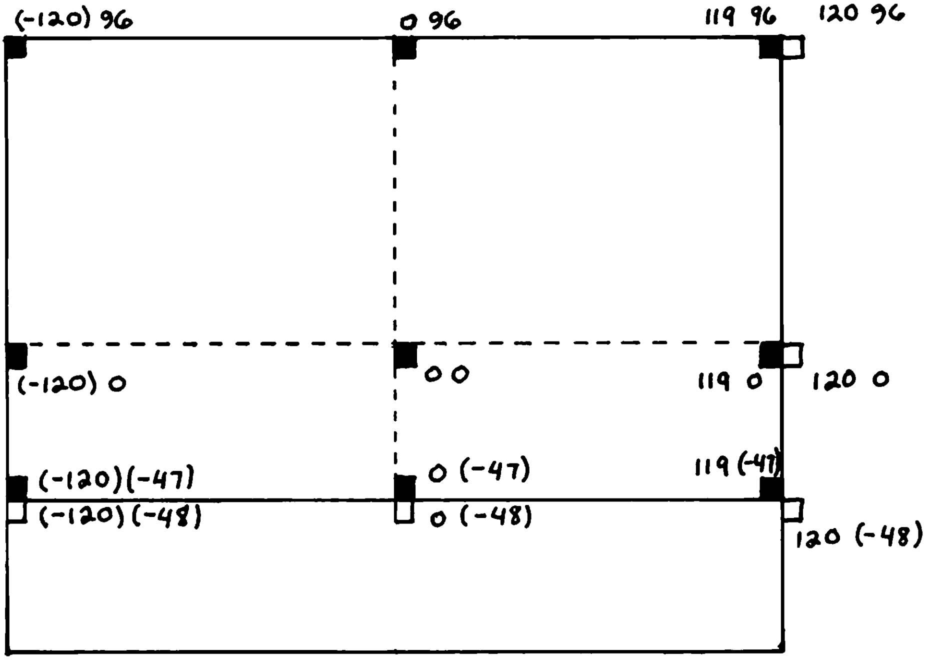
```
BK    BACK       MOVES CURRENT TURTLE OR SPRITES BACKWARD.

      BK 20

FD    FORWARD    MOVES CURRENT TURTLE OR SPRITES FORWARD.

      FD 20

LT    LEFT       TURNS CURRENT TURTLE OR SPRITES LEFT.

      LT 30          LEFT 30 DEGREES

RT    RIGHT      TURNS CURRENT TURTLE OR SPRITES RIGHT.

      RT 30          RIGHT 30 DEGREES
```

Absolute Motion Commands

```
HOME       SENDS THE CURRENT TURTLE OR SPRITES TO HOME POSITIONS
           THE TURTLE AND SPRITE HOMES ARE DIFFERENT

      TELL 1 HOME
      TELL [1 2 3] HOME
      TELL TURTLE HOME

SH    SETHEADING      SETS DIRECTION OF CURRENT TURTLE OR SPRITES

      SH 90          SET HEADING 90 DEGREES FROM NORTH

SS    SETSPEED       SETS SPEED OF CURRENT SPRITES

      SS 20          ACCEPTS NUMBERS -127 TO 127

SX    SET X COORDINATE OF CURRENT TURTLE OR SPRITES

      SX 20          ACCEPTS NUMBERS -120 TO 119

SXY   SET X AND Y COORDINATES OF CURRENT TURTLE OR SPRITES

      SXY 20 30

SY    SET Y COORDINATE OF CURRENT TURTLE OR SPRITE

      SY 30          ACCEPTS NUMBERS -47 TO 96


SXV   SET X VELOCITY OF CURRENT SPRITES

      SXV 10         ACCEPTS NUMBERS -127 TO 127

SYV   SET Y VELOCITY OF CURRENT SPRITES

      SYV 10         ACCEPTS NUMBERS -127 TO 127

FREEZE               STOPS SPRITE MOTION

THAW                 STARTS SPRITE MOTION AGAIN
```

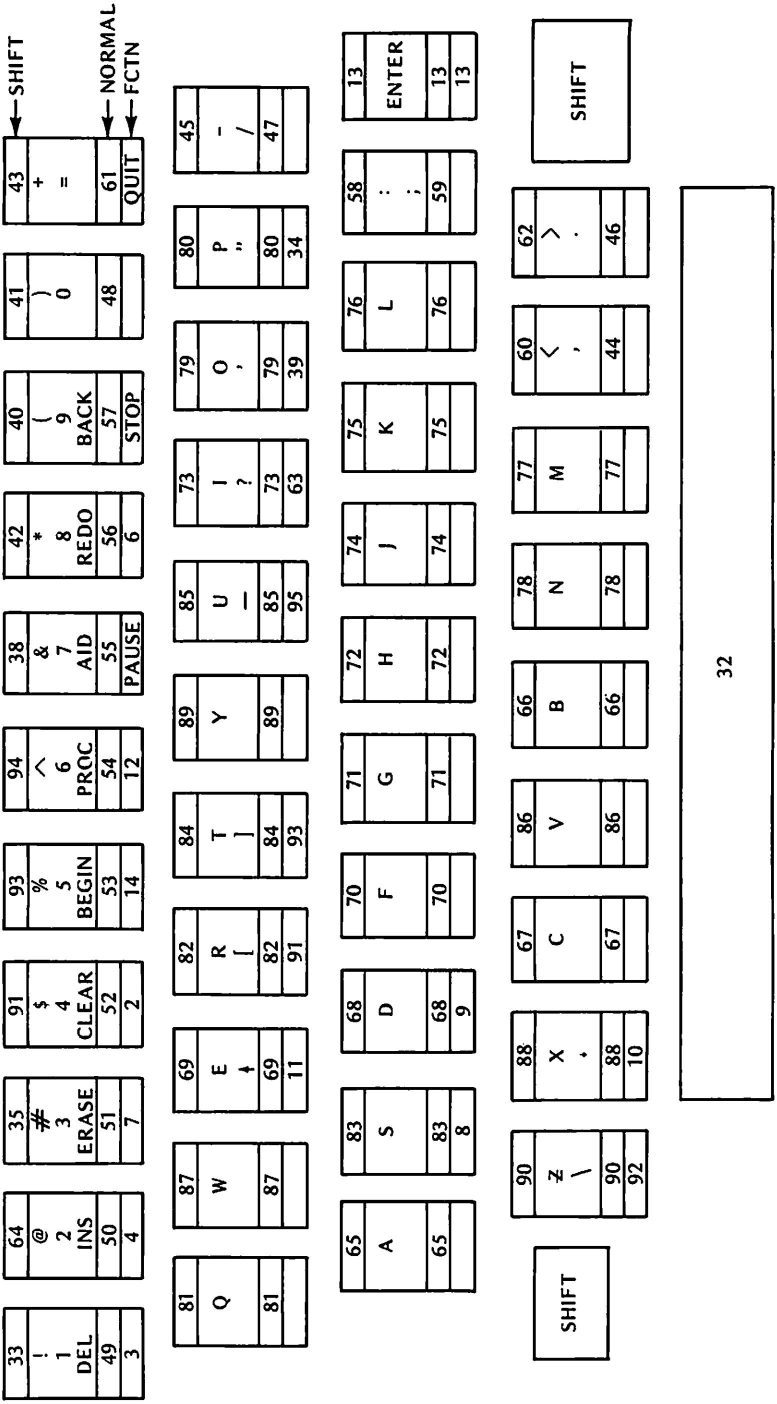
Turtle-Sprite Screen

This table shows the X and Y coordinates of various positions on the screen used by the turtle and the sprites.



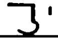
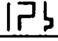
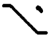
The Key Number Table

This table shows the number of the tile used by each key on the keyboard, including (SHIFT) keys, normal keys, and (FCTN) keys. The numbers are those returned by the CHARNUM procedure CN.


SHIFT
NORMAL
FCTN
33 ! 1 DEL 49 3
64 @ 2 INS 50 4
35 # 3 ERASE 51 7
91 $ 4 CLEAR 52 2
93 % 5 BEGIN 53 14
94 ^ 6 PROC 54 12
38 & 7 AID 55 PAUSE
42 * 8 REDO 56 6
40 (9 BACK 57 STOP
41) 0 48
43 + = 61 QUIT
81 Q 81
87 W 87
69 E ↑ 69 11
82 R [82 91
84 T] 84 93
89 Y 89
85 U _ 85 95
73 I ? 73 63
79 O ' 79 39
80 P " 80 34
45 - / 47
65 A 65
83 S 83 8
68 D 68 9
70 F 70
71 G 71
72 H 72
74 J 74
75 K 75
76 L 76
58 : ; 59
13 ENTER 13 13
SHIFT
90 Z \ 90 92
88 X ↓ 88 10
67 C 67
86 V 86
66 B 66
78 N 78
77 M 77
60 < , 44
62 > . 46
SHIFT
32

The Character—Tile Table

This table shows the contents of each character-tile when the computer is first turned on. Tiles 0 and 1 are special. They are continually reinitialized.

0 □	1 ■	2 	3	4	5	6 	7
8 	9	10 ©	11	12	13 (CR)	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 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7
56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G
72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	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
128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151
152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167
168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183
184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215
216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231
232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247
248	249	250	251	252	253	254	255

Index

Accessories, 3
ALL, 12-13
ALPHA LOCK key, 6
Apostrophe mark, 91
Arithmetic, 126ff
Arrow keys, 26, 46ff
ASCII numbers, 105

BACK, 27, 62
BACK key, 6, 24, 44
Background color of tiles, 108
Ball, 12
BEEP, 5
BEGIN key, 6
BF, 171
BK, 28
Blank character, 105
BOUNCE, 35
Box, 24
BUTFIRST, 171
BUTLAST, 171

CALL, 86
CARRY, 22ff
Cassette recorder, 188
  recalling procedures, 193
  recalling shapes, 197
  saving procedures, 190
  saving shapes, 197
  tapes, 189
CB, 9, 23
Character screen, 116
Character tiles, 103, 106
CHARNUM, 113
CLEAR key, 6, 8, 105
CLEARSCREEN, 8, 63
CN, 113
Coin tosser, 139
Colon mark, 10
COLORBACKGROUND, 9
Color families, 109
Colors, 9, 108
Comments, 153
Counting, 115, 125
CS, 8, 63
CTRL key, 6
Cursor, 111

DEFINE, 42
Delete, 47
DEL key, 6
DISK drive, set-up, 198ff
DISK Manager Command Module, 200

Disks, floppy, 199
  initializing, 200ff
  recalling procedures, 203
  recalling shapes, 206
  saving procedures, 202
  saving shapes, 205
Divide, 127
Do, 179
DOT, 68ff
Double quote mark, 10, 81

EACH 37, 133
East, 14
Editor, 91, 169
EMOTE, 53
END, 45
ENTER key, 6, 112
ERASE command, 55
ERASE key, 6-7
Erasing procedures, 55
Expansion box, 3

F, 173
FACE, 50ff, 158
FALSE, 97, 131
FD, 27, 62
FIRST, 129, 173
Floppy disks, 199
FNCT key, 6
FOR, 182
Form letter, 92
FORWARD 27, 62
FPUT, 175
Function key, 7

GET, 174
Go, 160
Greater than, 131

Heading, 135
HIDETURTLE, 64
HOME, 22, 70, 75
HT, 64

IF ..THEN 88, 131
Initializing disks, 200
Input, 140
Insert, 47
Interactive story, 94

JIFFY, 16
JOY, 148ff
Joystick numbers, 149ff
Joysticks, 147ff

Keyboard, TI 99/4A, 6, 222

Labeled lines, 160
LAST, 171
LEFT, 29, 60
Less than, 131
LIST EDITOR, 91, 169
LISTS, 11, 82, 165ff
LOGO module, 2-3
LOGO names, 13
LOGO numbers, 131, 167
LPUT, 176
LT, 29, 60

MAKE, 83, 84, 86
MAKECHAR, 104ff
MAKESHAPE, 25
MC, 104ff
Minimum TI equipment, 2
MS, 25
Multiplication, 126

NAMES, 11, 83, 168
NOBEEP, 5
North, 14
Numbers, in LOGO, 130, 167
NUMBER?, 181

OBJECTS, 166
OP, 138
Order in arithmetic operations, 128
OUT OF INK, 65
OUTPUT, 138

Parentheses, 126
PC, 111
PD, 66
PE, 66
Pen colors, 65
PENDOWN 66
PENERASE, 66
PENREVERSE, 67
PENUP, 66
Peripheral expansion unit, 3
PLANE 25
PN, 12
PO, 54
PP, 54
PR 67
Precedence of sprites, 34
PRINTCHAR, 111
PRINT command, 10
PRINTNAMES, 12
PRINTOUT, 54
PRINTPROCEDURES, 54
Proceed key, 6
Procedure, 41ff
Procedure editing mode, 45ff
Procedure input, 141
Procedure names, 82
Procedure using a procedure, 44
PT 117
PU, 66
PUTTILE 117

QUIT key, 6, 108
Quote mark
  double, 10, 81
  single, 91

RANDOM, 14-15
RC, 86
RC?, 97
READCHAR, 86
READLINE, 88, 128
Recalling
  procedures from disk, 203
  procedures from tape, 193
  shapes from disk, 206
  shapes from tape, 197
Remarks, 153
Reminder strip, 5-6
REPEAT, 15
RIGHT, 29, 60
RL, 88
ROCKET, 25
RT, 29, 60
RUN, 178

Saving
  procedures on disk, 198ff
  procedures on tape, 188ff
  shapes on disk, 205
  shapes on tape, 197
SC 22, 65
Screen map, 63, 68ff, 221
Semicolon mark, 153
SENTENCE 176
SETCOLOR, 22, 65, 108ff
SETHEADING 31, 61
SETSPEED, 32
SETX, 72
SETXY 72
SETY, 72
SH, 31, 61
Shapes, 25ff
SHIFT key, 6, 8
SHOWTURTLE, 64
Single quote mark, 91
South, 14
Space, 91
Speed, 134
Sprite precedence, 34
Sprite procedure,
Sprite reports, 134
Sprites, 21ff
Sprite shapes, 25ff
Square brackets, 11, 82, 165
SS, 32
ST 64
STOP, 137

Story, 94
SX, 72
SXY, 72
SY, 72

TALK 50ff
TELL, 22, 33, 35, 59, 119
TELL ME HOW TO . . . , 5
TEXAS SURPRISE 112
THING, 181
THING?, 180
Tile color, 108
Tile families, 109
Tiles, 103ff
TO, 45ff
TRUCK 24
TRUE, 97, 131
TURN 29
Turtle, 1, 59
Turtle coordinates, 72
Turtle pen colors, 65
Turtle screen, 63, 68ff, 221
TYPE, 87, 90, 93

WAIT 16
WEST, 14
WHILE, 179
Word, 81, 166
WORD, 174
WORD?, 181

X coordinate, 72
XCOR, 74, 134
XVEL, 134

Y coordinate, 72
YCOR, 74, 134
YN, 37, 133
YOURNUMBER, 37, 133
YVEL 134

Sprites, A Turtle, and TI LOGO

JIM CONLAN • DON INMAN

with DYMAX

Welcome to TI LOGO®—an amazingly friendly and fun-to-learn programming language! Use your TI99/4A® home computer to control the movements of the drawing Turtle and the 32 flying sprites. The Turtle is a funny little creature that will do anything you ask—draw squares, draw strange shapes, move in any direction you want it to. You can even make the Turtle disappear! The invisible sprites will then move the shape—a ball, a rocket, a plane, a box, for instance—up, down, to the left, to the right, or even off the screen! Whether you are a beginner or have programmed before, TI LOGO is a language you'll love. Before you know it, you'll find yourself spending hours moving your hard-working Turtle into a world where the only limit is your imagination.

Table of Contents

- Getting Started
- Sprites
- Create Your Own Commands
- Meet the Turtle
- "NAME, :NAME, NAME [NAME]
- Tile Tales
- What Counts
- The Joysticks

A Word About Lists
- Saving and Recalling

TI and TI99/4A are registered trademarks of Texas Instruments

RESTON PUBLISHING COMPANY, INC.
A Prentice-Hall Company
Reston, Virginia

0-8359-7036-1